Cloud Container Engine

User Guide

Issue	01
Date	2025-02-18

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 High-Risk Operations

During service deployment or running, you may trigger high-risk operations at different levels, causing service faults or interruption. To help you better estimate and avoid operation risks, this section introduces the consequences and solutions of high-risk operations from multiple dimensions, such as clusters, nodes, networking, load balancing, logs, and EVS disks.

Clusters and Nodes

Table 1-1 High-risk operations and solutions

Category	Operation	Impact	Solution
Master node	Modifying the security group of master nodes in a cluster **NOTE** Naming rule of a master node's security group: *Cluster name*-**cce-control**-*Random digits*	The master node may be unavailable.	Restore the security group by referring to "Creating a Cluster" and allow traffic from the security group to pass through. For details, see **How Can I Configure a Security Group Rule in a Cluster?**
	Letting the node expire or destroying the node	The master node will be unavailable.	This operation cannot be undone.
	Reinstalling the OS	Components on the master node will be deleted.	This operation cannot be undone.
	Upgrading components on the master or etcd node	The cluster may be unavailable.	Roll back to the original version.

Category	Operation	Impact	Solution
	Deleting or formatting core directory data such as **/etc/kubernetes** on the node	The master node will be unavailable.	This operation cannot be undone.
	Changing the node IP address	The master node will be unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as etcd, kube-apiserver, and docker)	The master node may be unavailable.	Restore the parameter settings to the recommended values. For details, see **Modifying Cluster Configurations**.
	Replacing the master or etcd certificate	The cluster may be unavailable.	This operation cannot be undone.
Worker node	Modifying the security group of worker nodes in a cluster **NOTE** Naming rule of a worker node's security group: *Cluster name*-**cce-node**-*Random digits*	The node may be unavailable.	Restore the security group by referring to "Creating a Cluster" and allow traffic from the security group to pass through. For details, see **How Can I Configure a Security Group Rule in a Cluster?**
	Modifying the DNS configuration (**/etc/resolv.conf**) of a node	Internal domain names cannot be accessed, which may lead to errors in functions such as add-on errors or errors in in-place node upgrade. **NOTE** If your service needs to use an on-premises DNS, configure the DNS in the workload. Do not change node's DNS address. For details, see **DNS Configuration**.	Restore the DNS configuration based on the DNS configuration of a new node.
	Deleting the node	The node will become unavailable.	This operation cannot be undone.

Category	Operation	Impact	Solution
	Reinstalling the OS	Node components are deleted, and the node becomes unavailable.	Reset the node. For details, see **Resetting a Node**.
	Upgrading the kernel or components on which the container platform depends (such as Open vSwitch, IPVLAN, Docker, and containerd)	The node may be unavailable or the network may be abnormal. **NOTE** Node running depends on the system kernel version. Do not use the **yum update** command to update or reinstall the operating system kernel of a node unless necessary. (Reinstalling the operating system kernel using the original image or other images is a risky operation.)	If the OS is EulerOS 2.2, restore the node or network connectivity by referring to **What Can I Do If the Container Network Becomes Unavailable After yum update Is Used to Upgrade the OS?** If the OS is not EulerOS 2.2, you can reset the node. For details, see **Resetting a Node**.
	Changing the node IP address	The node will become unavailable.	Change the IP address back to the original one.
	Modifying parameters of core components (such as kubelet and kube-proxy)	The node may become unavailable, and components may be insecure if security-related configurations are modified.	Restore the parameter settings to the recommended values. For details, see **Modifying Node Pool Configurations**.
	Modifying OS configuration	The node may be unavailable.	Restore the configuration items or reset the node. For details, see **Resetting a Node**.
	Deleting or modifying the **/opt/cloud/cce** and **/var/paas** directories, and deleting the data disk	The node will become unavailable.	Reset the node. For details, see **Resetting a Node**.

Category	Operation	Impact	Solution
	Modifying the node directory permission and the container directory permission, which involves the following directories: /usr/lib/systemd/system/kubelet.service /usr/lib/systemd/system/containerd-monit.service /usr/lib/systemd/system/docker-monit.service /opt/cloud/cce /var/paas /var/paas/script /var/paas/sys/log /var/paas/kubernetes /var/script/docker /var/script/kubelet /etc/containerd /etc/rc.local /etc/sudoers.d/sudoerspaas /etc/sysconfig/docker /etc/docker/daemon.json /var/lib/docker	The permissions will be abnormal.	Do not modify the permissions. Restore the permissions if they have been modified.
	Formatting or partitioning system disks, Docker disks, and kubelet disks on nodes.	The node may be unavailable.	Reset the node. For details, see **Resetting a Node**.
	Installing other software on nodes	This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.	Uninstall the software that has been installed and restore or reset the node. For details, see **Resetting a Node**.
	Modifying NetworkManager configurations	The node will become unavailable.	Reset the node. For details, see **Resetting a Node**.
	Deleting system images such as **cce-pause** from the node	Containers cannot be created and system images cannot be pulled.	Copy the image from a functional node for restoration.

Category	Operation	Impact	Solution
	Changing the flavor of a node in a node pool on the ECS console	If a node flavor is different from the flavor specified in the node pool where the node resides, the increased number of nodes in a node pool scale-out is different from the expected number.	Change the node flavor to the one specified in the node pool, or delete the node and perform a node pool scale-out again.

Network

Table 1-2 Network

Operation	Impact	Solution
Changing the value of the kernel parameter **net.ipv4.ip_forward** to **0**	The network becomes inaccessible.	Change the value to **1**.
Changing the value of the kernel parameter **net.ipv4.tcp_tw_recycle** to **1**	The NAT service becomes abnormal.	Change the value to **0**.
Changing the value of the kernel parameter **net.ipv4.tcp_tw_reuse** to **1**	The network becomes abnormal.	Change the value to **0**.
Not configuring the node security group to allow UDP packets to pass through port 53 of the container CIDR block	The DNS in the cluster cannot work properly.	Restore the security group by referring to the operations provided for **a newly created cluster** and allow traffic from the security group to pass through. For details, see **How Can I Configure a Security Group Rule in a Cluster?**
Deleting network-attachment-definitions CRD resources of default-network	The container network is disconnected, or the cluster fails to be deleted.	If the resources are deleted by mistake, use the correct configurations to create the default-network resources.

Operation	Impact	Solution
Enabling the iptables firewall	By default, the iptables firewall is disabled on CCE. Enabling the firewall can leave the network inaccessible. **NOTE** Do not enable the iptables firewall. If the iptables firewall must be enabled, check whether the rules configured in **/etc/sysconfig/iptables** and **/etc/sysconfig/ip6tables** in the test environment will affect the network.	Disable the iptables firewall and check the rules configured in **/etc/sysconfig/iptables** and **/etc/sysconfig/ip6tables**.

Containers

Table 1-3 Containers

Operation	Impact	Solution
Configuring privileged containers for a workload and directly operating the host hardware, which are prone to misoperations on the system files of the node For example, if you set the startup command to **/usr/sbin/init** and run **systemctl** in containers, the system files located in the **/lib** directory of the node may be damaged.	All mount points of the node will be unmounted. As a result, the node will be malfunctional, resulting in failed pods and affected storage add-on functions.	Do not remove the mount points in the **/lib** directory of a node. Reset the node for recovery. For details, see **Resetting a Node**.

Load Balancing

Table 1-4 Service ELB

Operation	Impact	Solution
Deleting a load balancer that has been bound to a CCE cluster on the ELB console	Accessing the target Service or ingress will fail.	Do not delete such a load balancer.
Disabling a load balancer that has been bound to a CCE cluster on the ELB console	Accessing the target Service or ingress will fail.	Do not disable such a load balancer. If a load balancer has been disabled, enable it.
Changing the private IPv4 address of a load balancer on the ELB console	• The network traffic forwarded using the private IPv4 addresses will be interrupted. • The IP addresses in the **status** field of Service or ingress YAML files will be changed.	Do not change private IPv4 addresses of load balancers. Change them back if they have been changed.
Unbinding the IPv4 EIP from a load balancer on the ELB console	After the EIP is unbound from the load balancer, the load balancer will not be able to forward Internet traffic.	Restore the EIP binding.
Creating a custom listener on the ELB console for the load balancer managed by CCE	If a load balancer is automatically created when a Service or an ingress is created, the custom listener of the load balancer cannot be deleted when the Service or ingress is deleted. In this case, the load balancer cannot be automatically deleted.	Use the listener automatically created when a Service or an ingress is created. If a custom listener is used, manually delete the target load balancer.
Deleting a listener automatically created by CCE on the ELB console	• Accessing the target Service or ingress will fail. • After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE.	Re-create or update the Service or ingress.

Operation	Impact	Solution
Modifying the basic configurations such as the name, access control, timeout, or description of a listener created by CCE on the ELB console	After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the listener is deleted.	Do not modify the basic configurations of the listener created by CCE. Restore the configurations if they have been modified.
Modifying the backend server group of a listener created by CCE on the ELB console, including adding or deleting backend servers to or from the server group	• Accessing the target Service or ingress will fail. • After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE. – Deleted backend servers will be restored. – Added backend servers will be removed.	Re-create or update the Service or ingress.
Replacing the backend server group of a listener created by CCE on the ELB console	• Accessing the target Service or ingress will fail. • After master nodes are restarted, for example, due to a cluster upgrade, all servers in the backend server group will be reset by CCE.	Re-create or update the Service or ingress.
Modifying the forwarding policy of a listener created by CCE on the ELB console, including adding or deleting forwarding rules	• Accessing the target Service or ingress will fail. • After master nodes are restarted, for example, due to a cluster upgrade, all your modifications will be reset by CCE if the forwarding rules are added using an ingress.	Do not modify the forwarding policy of such a listener. Restore the configurations if they have been modified.

Operation	Impact	Solution
Changing the ELB certificate on the ELB console for a load balancer managed by CCE	After master nodes are restarted, for example, due to a cluster upgrade, all servers in the backend server group will be reset by CCE.	Use the YAML file of the ingress to automatically manage certificates.

Logs

Table 1-5 Logs

Operation	Impact	Solution
Deleting the **/tmp/ccs-log-collector/pos** directory on the host machine	Logs are collected repeatedly.	None
Deleting the **/tmp/ccs-log-collector/buffer** directory on the host machine	Logs are lost.	None

EVS Disks

Table 1-6 EVS disks

Operation	Impact	Solution	Remarks
Manually unmounting an EVS disk on the console	An I/O error occurs when data is written into a pod.	Delete the mount path from the node and schedule the pod again.	The file in the pod records the location where files are to be collected.
Unmounting the disk mount path on the node	Pod data is written into a local disk.	Remount the corresponding path to the pod.	The buffer contains log cache files to be consumed.
Operating EVS disks on the node	Pod data is written into a local disk.	None	None

Operation	Impact	Solution	Remarks
Creating a PV with parameters that are not declared in the file For example, if the YAML file contains parameters such as **status**, **spec.claimRef**, and **annotation.everest.io/set-disk-metadata** during PV creation, the PV may be abnormal.	This operation may bypass some standard processes for creating PVs. As a result, the created PVs may become unavailable or be deleted unexpectedly.	Before such PVs are deleted, manually delete related parameters in their YAML files.	None

Add-ons

Table 1-7 Add-ons

Operation	Impact	Solution
Modifying add-on resources on the backend	The add-on becomes malfunctional or other unexpected issues occur.	Perform operations on the add-on configuration page or using open add-on management APIs.

2 Clusters

2.1 Basic Cluster Information

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications.

For developers, Kubernetes is a cluster operating system. Kubernetes provides service discovery, scaling, load balancing, self-healing, and even leader election, freeing developers from infrastructure-related configurations.

Cluster Network

A cluster network can be divided into three network types:

- Node network: IP addresses are assigned to nodes in a cluster.
- Container network: IP addresses are assigned to containers in a cluster for communication. Currently, multiple container network models are supported, and each model has its own working mechanism.
- Service network: A Service is a Kubernetes object used to access containers. Each Service has a static IP address.

When you create a cluster, select a proper CIDR block for each network. Ensure that the CIDR blocks do not conflict with each other and have sufficient available IP addresses. **You cannot change the container network model after the cluster is created.** Plan the container network model properly in advance.

You are advised to learn about the cluster network and container network models before creating a cluster. For details, see **Container Network**.

Master Nodes and Cluster Scale

When you create a cluster on CCE, you can have one or three master nodes. Three master nodes will be deployed in a cluster for HA.

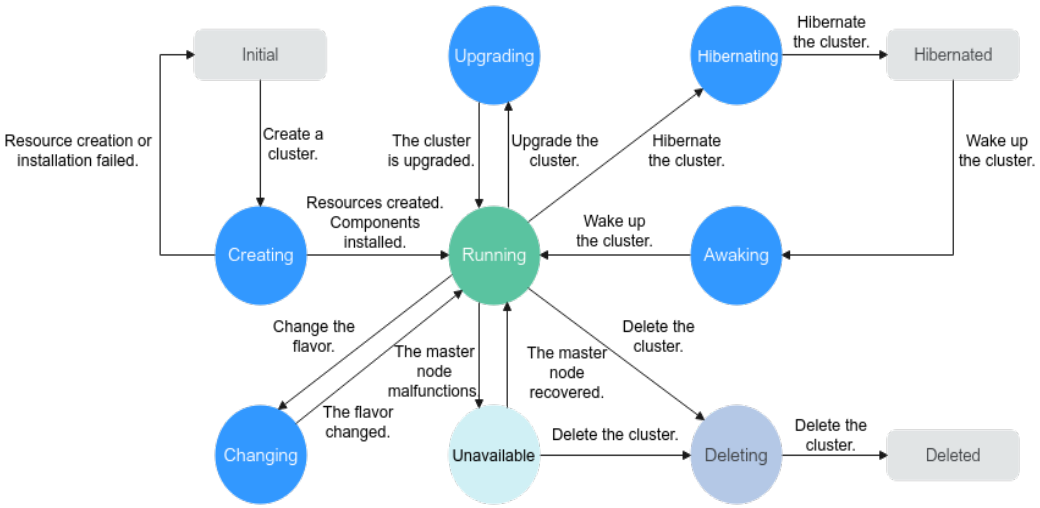
The master node specifications decide the number of nodes that can be managed by a cluster. You can select the cluster management scale, for example, 50 or 200 nodes.

Cluster Lifecycle

Table 2-1 Cluster status

Status	Description
Creating	A cluster is being created and is requesting for cloud resources.
Running	A cluster is running properly.
Hibernating	A cluster is hibernating.
Awaking	A cluster is being woken up.
Upgrading	A cluster is being upgraded.
Resizing	The cluster flavor is being changed.
Unavailable	A cluster is unavailable.
Deleting	A cluster is being deleted.

Figure 2-1 Cluster status transition



2.2 Cluster Version Release Notes

2.2.1 Kubernetes Version Release Notes

2.2.1.1 Kubernetes 1.31 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.31. This section describes the changes made in Kubernetes 1.31.

Indexes

New and Enhanced Features

- Start ordinal of StatefulSets is in the General Availability (GA) state.

  StatefulSet start ordinal is advanced to GA. By default, each pod in a StatefulSet is assigned an integer ordinal from 0. With this feature, you can configure a start ordinal for each pod. For details, see **Start ordinal**.

- Elastic indexed jobs are in the GA state.

  Elastic indexed jobs are advanced to GA. You can scale indexed Jobs up or down by modifying **.spec.completions** and **.spec.parallelism**. For details, see **Elastic Indexed Jobs**.

- Pod failure policy is in the GA state.

  Pod failure policies are advanced to GA. This feature allows you to choose how the system handles pod failures by specifying the processing mode (retry or ignore). This helps prevent unnecessary pod restarts. For details, see **Pod failure policy**.

- Pod disruption conditions are in the GA state.

  Pod disruption conditions are advanced to GA. The new **DisruptionTarget** condition indicates the reasons for pod failures, such as being preempted by a higher priority pod, cleared due to node deletion, or terminated by kubelet. When a pod is created using a job or cron job, you can use these pod disruption conditions as part of your job's **pod failure policy** to define the action when a pod is abnormal. For details, see **Pod disruption conditions**.

- Selectable fields for custom resources are in the beta state.

  Selectable fields for custom resources are moved to beta. You can specify the **selectableFields** field of a CustomResourceDefinition (CRD) to define which other fields in a custom resource may be used in field selectors. Field selectors can then be used to get only resources by filtering List, Watch, and DeleteCollection requests. For details, see **Selectable fields for custom resources**.

- Job success policies are in the beta state.

  Job success policies is moved to beta. This feature allows you to configure success policies for jobs based on the number of pods that succeeded. For details, see **Success policy**.

- **matchLabelKeys** is in the beta state.

  matchLabelKeys is moved to beta. **matchLabelKeys** and **mismatchLabelKeys** are finer fields for pod affinity or anti-affinity. They specify the keys for the labels that should or should not match with the incoming pod's labels, so that a rolling upgrade will not break affinity or anti-affinity. For details, see **matchLabelKeys**.

- ServiceAccountTokenNodeBinding is in the beta state.

  ServiceAccountTokenNodeBinding is moved to beta. You can create a service account token that is directly bound to a node. The token defines the node

information and verifies whether the node is available. The token will be valid until it expires or either the associated node is deleted. For details, see **Manually create an API token for a ServiceAccount**.

API Changes and Removals

- In Kubernetes 1.31, the **kubectl exec [POD] [COMMAND]** command cannot be executed without a **--** separator. In this case, you need to run **kubectl exec [POD] -- [COMMAND]**.
- In Kubernetes 1.31, if **caBundle** is not empty but the value is invalid or it does not define any CA certificate, the CRD does not provide services. If **caBundle** is set to a valid value, it remains unchanged if updated. Attempting direct updates results in an "invalid field value" error, ensuring uninterrupted CRD services.

Enhanced Kubernetes 1.31 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.31 and provides enhanced functions.

For details about cluster version updates, see **Patch Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.31 and other versions, see **Kubernetes v1.31 Release Notes**.

2.2.1.2 Kubernetes 1.30 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.30. This section describes the changes made in Kubernetes 1.30.

Indexes

New and Enhanced Features

- Webhook matching expression is in the GA state.

  The Webhook matching expression feature is advanced to GA. This feature enables admission webhooks to be matched based on specific conditions, providing control over the triggering conditions of the webhooks in a more precise granularity. For details, see **Dynamic Admission Control**.

- Pod scheduling readiness is in the GA state.

  The pod scheduling readiness feature is advanced to GA. With this feature, you can add custom scheduling gates to a pod and manage when to eliminate them. The pod will only be deemed ready for scheduling once all

scheduling gates have been removed. For details, see **Pod Scheduling Readiness**.

- Validating admission policies are in the GA state.

  Validating admission policies are advanced to GA. This feature allows you to declare the validating admission policies of resources using Common Expression Language (CEL). For details, see **Validating Admission Policy**.

- Horizontal pod auto scaling based on container resource metrics is in the GA state.

  The horizontal pod auto scaling feature based on container resource metrics is advanced to GA. This feature allows HPA to configure auto scaling based on the resource usage of each container within a pod, rather than just the overall resource usage of the pod. This makes it easier to set scaling thresholds for the most critical containers in a pod. For details, see **Container resource metrics**.

- The legacy ServiceAccount token cleaner is in the GA state.

  The legacy ServiceAccount token cleaner feature is advanced to GA. It runs as part of **kube-controller-manager** and checks every 24 hours to see if any auto-generated legacy ServiceAccount token has not been used in a specific amount of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**). If so, the cleaner marks those tokens as invalid and adds the **kubernetes.io/legacy-token-invalid-since** label whose value is the current date. If an invalid token is not used for a specific period of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**), the cleaner deletes it. For details, see **Legacy ServiceAccount token cleaner**.

- The minimum domain in the pod topology spread is in the GA state.

  The minimum domain feature in pod topology spread is advanced to GA. This feature allows you to configure a minimum number of domains that meet specific conditions by using the **minDomains** field in the pod configuration. If the number of domains that match the load topology constraints exceeds the **minDomains** value, this field will not affect the settings. However, if the number of domains that match the load topology constraints is less than the **minDomains** value, the global minimum value is set to 0, which represents the minimum number of matched pods in domains that meet the conditions. To prevent pods from being scheduled when topology constraints are not met, this field must be used together with **whenUnsatisfiable: DoNotSchedule**. For details, see **Spread constraint definition**.

API Changes and Removals

- kubectl removes the **prune-whitelist** parameter of the **apply** command and replaces it with **prune-allowlist**.
- SecurityContextDeny, which has been deprecated in Kubernetes 1.27, is replaced by **Pod Security admission**.

Enhanced Kubernetes 1.30 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.30 and provides enhanced functions.

For details about cluster version updates, see **Patch Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.30 and other versions, see **Kubernetes v1.30 Release Notes**.

2.2.1.3 Kubernetes 1.29 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.29. This section describes the changes made in Kubernetes 1.29.

Indexes

New and Enhanced Features

- The load balancer IP mode for Services is in the alpha state.

  The load balancer IP mode for Services is promoted to alpha. Kubernetes 1.29 adds the **ipMode** field to the Services' **status** field for configuring traffic forwarding from Services within a cluster to pods. If **ipMode** is set to **VIP**, traffic delivered to a node with the destination set to the load balancer's IP address and port will be redirected to the target node by kube-proxy. If it is set to **Proxy**, traffic delivered to a node will be sent to the load balancer and then redirected to the target node by the load balancer. This feature addresses the issue of load balancer functions being missed due to traffic bypassing it. For details, see **Load Balancer IP Mode for Services**.

- The nftables proxy mode is in the alpha state.

  The nftables proxy mode is promoted to alpha. This feature allows kube-proxy to run in nftables mode. In this mode, kube-proxy configures packet forwarding rules using the nftables API of the kernel netfilter subsystem. For details, see **nftables proxy mode**.

- Garbage collection for unused container images is in the alpha state.

  The garbage collection for unused container images is promoted to alpha. This feature allows you to specify the maximum time a local image can be unused for each node. If the time expires, the image will be garbage collected. To configure the setting, specify the **ImageMaximumGCAge** field for kubelet. For details, see **Garbage collection for unused container images**.

- **PodLifecycleSleepAction** is in the alpha state.

  PodLifecycleSleepAction is promoted to alpha. This feature introduces the sleep hook to the container lifecycle hooks. You can pause a container for a specified duration after it starts or before it is stopped by enabling this feature. For details, see **Hook handler implementations**.

- **KubeletSeparateDiskGC** is in the alpha state.

  KubeletSeparateDiskGC is promoted to alpha. With this feature enabled, container images and containers can be garbage collected even if they are on separate file systems.

- **matchLabelKeys** and **mismatchLabelKeys** are in the alpha state.

  matchLabelKeys and **mismatchLabelKeys** are promoted to alpha. With these features enabled, the **matchLabelKeys** and **mismatchLabelKeys** fields are added to the pod affinity and anti-affinity configurations. This allows for configurations of more affinity and anti-affinity policies between pods. For details, see **matchLabelKeys and mismatchLabelKeys**.

- The **clusterTrustBundle** projected volumes are in the alpha state.

  The **clusterTrustBundle** projected volumes are promoted to alpha. With this feature enabled, the **clusterTrustBundle** projected volume source injects the contents of one or more ClusterTrustBundle objects as an automatically-updating file. For details, see **clusterTrustBundle projected volumes**.

- Pulling images based on runtime classes of pods is in the alpha state.

  Pulling images based on runtime classes is promoted to alpha. With this feature enabled, the kubelet references container images by a tuple (of image name or runtime handler) rather than just the image name or digest. Your container runtime may adapt its behavior based on the selected runtime handler. Pulling images based on runtime classes will be helpful for VM based containers. For details, see **Image pull per runtime class**.

- The **PodReadyToStartContainers** condition is in the beta state.

  The **PodReadyToStartContainers** condition is promoted to beta. Kubernetes 1.29 introduces the **PodReadyToStartContainers** condition to the pods' **status** field. If it is set to **true**, the sandbox of a pod is ready and service containers can be created. This feature enables cluster administrators to gain a clearer and more comprehensive view of pod sandbox creation completion and container readiness. This enhanced visibility allows them to make better-informed decisions and troubleshoot issues more effectively. For details, see **PodReadyToStartContainers Condition Moves to Beta**.

- Two job-related features are in the beta state.
  - Pod replacement policy (beta)

    The pod replacement policy feature moves to beta. This feature ensures that a pod is replaced only when it reaches the **Failed** state, which means that **status.phase** becomes **Failed**. It does not recreate a pod when the deletion timestamp is not empty and the pod is still being deleted. This prevents two pods from occupying index and node resources concurrently.
  - Backoff limit per index (beta)

    The backoff limit per index moves to beta. By default, pod failures for indexed jobs are counted and restricted by the global limit of retries, specified by **.spec.backoffLimit**. This means that if there is a consistently failing index in a job, pods specified by the job will be restarted repeatedly until pod failures exhaust the limit. Once the limit is reached, the job is marked failed and pods for other indexes in the job may never be even started. The feature allows you to complete execution of all indexes, despite some indexes failing, and to better use the compute resources by avoiding unnecessary retries of consistently failing indexes.

- Native sidecar containers are in the beta state.

  Native sidecar containers are promoted to beta. The **restartPolicy** field is added to **initContainers**. When this field is set to **Always**, the sidecar container is enabled. The sidecar container and service container are deployed in the same pod. This cannot prolong the pod lifecycle. Sidecar containers are

commonly used in scenarios such as network proxy and log collection. For details, see **Sidecar Containers**.

- The legacy ServiceAccount token cleaner is in the beta state.

  Legacy ServiceAccount token cleaner is promoted to beta. It runs as part of **kube-controller-manager** and checks every 24 hours to see if any auto-generated legacy ServiceAccount token has not been used in a specific amount of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**). If so, the cleaner marks those tokens as invalid and adds the **kubernetes.io/legacy-token-invalid-since** label whose value is the current date. If an invalid token is not used for a specific period of time (one year by default, specified by **--legacy-service-account-token-clean-up-period**), the cleaner deletes it. For details, see **Legacy ServiceAccount token cleaner**.

- **DevicePluginCDIDevices** is in the beta state.

  DevicePluginCDIDevices moves to beta. With this feature enabled, plugin developers can use the **CDIDevices** field added to **DeviceRunContainerOptions** to pass CDI device names directly to CDI enabled runtimes.

- **PodHostIPs** is in the beta state.

  The **PodHostIPs** feature moves to beta. With this feature enabled, Kubernetes adds the **hostIPs** field to **Status** of pods and downward API to expose node IP addresses to workloads. This field specifies the dual-stack protocol version of the host IP address. The first IP address is always the same as the host IP address.

- The API Priority and Fairness feature (APF) is in the GA state.

  APF moves to GA. APF classifies and isolates requests in a more fine-grained way. It improves max-inflight limitations. It also introduces a limited amount of queuing, so that the API server does not reject any request in cases of very brief bursts. Requests are dispatched from queues using a fair queuing technique so that, for example, a poorly-behaved controller does not cause others (even at the same priority level) to become abnormal. For details, see **API Priority and Fairness**.

- **APIListChunking** is in the GA state.

  The **APIListChunking** feature moves to GA. This feature allows clients to perform pagination in List requests to avoid performance problems caused by returning too much data at a time.

- **ServiceNodePortStaticSubrange** is in the GA state.

  The **ServiceNodePortStaticSubrange** feature moves to GA. With this feature enabled, kubelet calculates the size of reserved IP addresses based on the ranges of the NodePort Services and divides node ports into static band and dynamic band. During automatic node port assignment, dynamic band is preferentially assigned, which helps avoid port conflicts during static band assignment. For details, see **ServiceNodePortStaticSubrange**.

- The phase transition timestamp of PersistentVolume (PV) is in the beta state.

  The PV phase transition timestamp moves to beta. With this feature enabled, Kubernetes adds the **lastPhaseTransitionTime** field to the **status** field of a PV to indicate the time when the PV phase changes last time. Cluster administrators are now able to track the last time a PV transitioned to a different phase, allowing for more efficient and informed resource

management. For details, see **PersistentVolume Last Phase Transition Time in Kubernetes**.

- **ReadWriteOncePod** is in the GA state.

  The **ReadWriteOncePod** feature moves to GA. With this feature enabled, you can set the access mode to **ReadWriteOncePod** in a PersistentVolumeClaim (PVC) to ensure that only one pod can modify data in the volume at a time. This can prevent data conflicts or damage. For details, see **ReadWriteOncePod**.

- **CSINodeExpandSecret** is in the GA state.

  The **CSINodeExpandSecret** feature moves to GA. This feature allows secret authentication data to be passed to a CSI driver for use when a node is added.

- The CEL-based CRD verification capability is in the GA state.

  The CEL-based CRD verification capability moves to GA. With this feature enabled, you are allowed to use the CEL to define validation rules in CRDs, which are more efficient than webhook. For details, see **CRD verification rules**.

API Changes and Removals

- The time zone of a newly created cron job cannot be configured using **TZ** or **CRON_TZ** in **.spec.schedule**. Use **.spec.timeZone** instead. Cron jobs that have been created are not affected by this change.
- The alpha API **ClusterCIDR** is removed.
- The startup parameter **--authentication-config** is added to kube-apiserver to specify the address of the **AuthenticationConfiguration** file. This startup parameter is mutually exclusive with the **--oidc-*** startup parameter.
- The API version **kubescheduler.config.k8s.io/v1beta3** of **KubeSchedulerConfiguration** is removed. Migrate **kube-scheduler** configuration files to **kubescheduler.config.k8s.io/v1**.
- The CEL expressions are added to **v1alpha1 AuthenticationConfiguration**.
- The **ServiceCIDR** type is added. It allows you to dynamically configure the IP address range used by a cluster to allocate the Service ClusterIPs.
- The startup parameters **--conntrack-udp-timeout** and **--conntrack-udp-timeout-stream** are added to **kube-proxy**. They are options for configuring the kernel parameters **nf_conntrack_udp_timeout** and **nf_conntrack_udp_timeout_stream**.
- Support for CEL expressions is added to **WebhookMatchCondition** of **v1alpha1 AuthenticationConfiguration**.
- The type of **PVC.spec.Resource** is changed from **ResourceRequirements** to **VolumeResourceRequirements**.
- **onPodConditions** in **PodFailurePolicyRule** is marked as optional.
- The API version **flowcontrol.apiserver.k8s.io/v1beta3** of **FlowSchema** and **PriorityLevelConfiguration** has been promoted to **flowcontrol.apiserver.k8s.io/v1**, and the following changes have been made:
  - **PriorityLevelConfiguration**:
    The **.spec.limited.nominalConcurrencyShares** field defaults to **30** if the field is omitted. To ensure compatibility with 1.28 API servers, specifying

an explicit **0** value is not allowed in the **v1** version in 1.29. In 1.30, explicit **0** values will be allowed in this field in the **v1** API. The **flowcontrol.apiserver.k8s.io/v1beta3** APIs are deprecated and will no longer be served in 1.32.

- The kube-proxy command line document is updated. kube-proxy does not bind any socket to the IP address specified by **--bind-address**.
- The **selectorSpread** scheduler plugin is replaced by **podTopologySpread**.
- If CSI-Node-Driver is not running, NodeStageVolume calls will be retried.
- **ValidatingAdmissionPolicy** type checking now supports CRDs. To use this feature, the **ValidatingAdmissionPolicy** feature gate must be enabled.
- The startup parameter **--nf-conntrack-tcp-be-liberal** is added to **kube-proxy**. You can configure it by setting the kernel parameter **nf_conntrack_tcp_be_liberal**.
- The startup parameter **--init-only** is added to **kube-proxy**. Setting the flag makes **kube-proxy** init container run in the privileged mode, perform its initial configuration, and then exit.
- The **fileSystem** field of container is added to the response body of CRI. It specifies the file system usage of a container. Originally, the **fileSystem** field contains only the file system of the container images.
- All built-in cloud providers are disabled by default. If you still need to use them, you can configure the **DisableCloudProviders** and **DisableKubeletCloudCredentialProvider** feature gates to disable or enable cloud providers.
- **--node-ips** can be used in kubelet to configure IPv4/IPv6 dual-stack. If **--cloud-provider** is set to **external**, you are allowed to use **--node-ips** to configure IPv4/IPv6 dual-stack for node IP addresses. To use **--node-ips**, you need to enable the **CloudDualStackNodeIPs** feature gate.

Enhanced Kubernetes 1.29 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.29 and provides enhanced functions.

For details about cluster version updates, see **Patch Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.29 and other versions, see **Kubernetes v1.29 Release Notes**.

2.2.1.4 Kubernetes 1.28 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create Kubernetes clusters 1.28. This section describes the changes made in Kubernetes 1.28.

Indexes

- **Important Notes**

- **New and Enhanced Features**
- **API Changes and Removals**
- **Feature Gate and Command Line Parameter Changes and Removals**
- **Enhanced Kubernetes 1.28 on CCE**
- **References**

Important Notes

- In Kubernetes 1.28, the scheduling framework is improved to reduce useless retries. The overall scheduling performance is enhanced. If a custom scheduler plugin is used in a cluster, you can perform the adaptation upgrade following instructions in **GitHub**.
- The Ceph FS in-tree plugin has been deprecated in Kubernetes 1.28 and will be removed in Kubernetes 1.31. (The community does not plan to support CSI migration.) Use **Ceph CSI driver** instead.
- The Ceph RBD in-tree plugin has been deprecated in Kubernetes 1.28 and will be removed in Kubernetes 1.31. (The community does not plan to support CSI migration.) Use RBD **Ceph CSI driver** instead.

New and Enhanced Features

Features in alpha stage are disabled by default, those in beta stage are enabled by default, and those in GA stage are always enabled and they cannot be disabled. The function of turning on or off the features in GA stage will be removed in later Kubernetes versions. CCE policies for new features are the same as those in the community.

- The version skew policy is expanded to three versions.

  Starting with control planes 1.28 and worker nodes 1.25, the Kubernetes skew policy expands the supported control plane and worker node skew to three versions. This enables annual minor version upgrades of nodes while staying on supported minor versions. For details, see **Version Skew Policy**.

- Retroactive Default StorageClass moves to GA.

  The retroactive default StorageClass assignment graduates to GA. This enhancement brings a significant improvement to how default StorageClasses are assigned to PersistentVolumeClaims (PVCs).

  The PV controller has been modified to automatically assign a default StorageClass to any unbound PVC with **storageClassName** not configured. Additionally, the PVC admission validation mechanism within the API server has been adjusted to allow changing values from an unset state to an actual StorageClass name. For details, see **Retroactive default StorageClass assignment**.

- Native sidecar containers are introduced.

  The native sidecar containers are available in alpha. Kubernetes 1.28 adds **restartPolicy** to Init containers. This field is available when the SidecarContainers feature gate is enabled. However, there are still some problems to be solved in the native sidecar containers. Therefore, the Kubernetes community recommends only using this feature gate in **short lived testing clusters** at the alpha phase. For details, see **Introducing native sidecar containers**.

- Mixed version proxy is introduced.

  A new mechanism (mixed version proxy) is released to improve cluster upgrade. It is an alpha feature in Kubernetes 1.28. When a cluster undergoes an upgrade, API servers of different versions in the cluster can serve different sets (groups, versions, or resources) of built-in resources. A resource request made in this scenario may be served by any of the available API servers, potentially resulting in the request ending up at an API server that may not be aware of the requested resource. As a result, the request fails. This feature can solve this problem. (Note that CCE provides hitless upgrade. Therefore, this feature is not used in CCE clusters.) For details, see **A New (alpha) Mechanism For Safer Cluster Upgrades**.

- Non-graceful node shutdown moves to GA.

  The non-graceful node shutdown is now GA in Kubernetes 1.28. When a node was shut down and that shutdown was not detected by the kubelet's Node Shutdown Manager, the StatefulSet pods that run on this node will stay in the terminated state and cannot be moved to a running node. If you have confirmed that the shutdown node is unrecoverable, you can add an **out-of-service** taint to the node. This ensures that the StatefulSet pods and VolumeAttachments on this node can be forcibly deleted and the corresponding pods will be created on a healthy node. For details, see **Non-Graceful Node Shutdown Moves to GA**.

- NodeSwap moves to beta.

  Support for NodeSwap goes to beta in Kubernetes 1.28. NodeSwap is disabled by default and can be enabled using the NodeSwap feature gate. NodeSwap allows you to configure swap memory usage for Kubernetes workloads running on Linux on a per-node basis. Note that although NodeSwap has reached beta, there are still some problems to be solved and security risks to be enhanced. For details, see **Beta Support for Using Swap on Linux**.

- Two job-related features are added.

  Two alpha features are introduced: **delayed creation of replacement pods** and **backoff limit per index**.

  - Delayed creation of replacement pods

    By default, when a pod enters the terminating state (for example, due to the preemption or eviction), Kubernetes immediately creates a replacement pod. Therefore, both pods are running concurrently.

    In Kubernetes 1.28, this feature can be enabled by turning on the JobPodReplacementPolicy feature gate. With this feature gate enabled, you can set the **podReplacementPolicy** field under **spec** of a job to **Failed**. In this way, pods would only be replaced when they reached the failed phase, and not when they are terminating. Additionally, you can check the **.status.termination** field of a job. The value of this field is the number of pods owned by the job that are currently terminating.

  - Backoff limit per index

    By default, pod failures for indexed jobs are recorded and restricted by the global limit of retries, specified by **.spec.backoffLimit**. This means that if there is a consistently failing index in a job, pods specified by the job will be restarted repeatedly until pod failures exhaust the limit. Once the limit is reached, the job is marked failed and pods for other indexes in the job may never be even started.

In Kubernetes 1.28, this feature can be enabled by turning on the JobBackoffLimitPerIndex feature gate of a cluster. With this feature gate enabled, **.spec.backoffLimitPerIndex** can be specified when an indexed job is created. Only if the failures of pods with all indexes specified in this job exceed the upper limit, pods specified by the job will not be restarted.

- Some CEL related features are improved.

  CEL related capabilities are enhanced.

  - CEL used to validate CRDs moves to beta.

    This feature has been upgraded to beta since Kubernetes 1.25. By embedding CEL expressions into CRDs, developers can solve most of the CR validation use cases without using webhooks. More CEL functions, such as support for default value and CRD conversion, will be developed in later Kubernetes versions.

  - CEL admission control graduates to beta.

    CEL admission control is customizable. With CEL expressions, you can decide whether to accept or reject requests received by kube-apiserver. CEL expressions can also serve as a substitute for admission webhooks. Kubernetes 1.28 has upgraded CEL admission control to beta and introduced new functions, such as:

    - ValidatingAdmissionPolicy can correctly handle the **authorizer** variable.
    - ValidatingAdmissionPolicy can have the **messageExpression** field checked.
    - The ValidatingAdmissionPolicy controller is added to kube-controller-manager to check the type of the CEL expression in ValidatingAdmissionPolicy and save the reason in the **status** field.
    - CEL expressions can contain a combination of one or more variables, which can be defined in ValidatingAdmissionPolicy. These variables can be used to define other variables.
    - CEL library functions can be used to parse resources specified by **resource.Quantity** in Kubernetes.

- Other features
  - The ServiceNodePortStaticSubrange feature gate moves to beta. With this feature enabled, static port range can be reserved to avoid conflicts with dynamically allocated ports. For details, see **Avoiding Collisions Assigning Ports to NodePort Services**.
  - The alpha feature ConsistentListFromCache is added to allow the API server to serve consistent lists from cache. Get and list requests can read data from the cache instead of etcd.
  - In Kubernetes 1.28, kubelet can configure the drop-in directory (alpha). This feature allows you to add support for the **--config-dir** flag to kubelet so that you can specify an insert directory that overwrites the kubelet configuration in **/etc/kubernetes/kubelet.conf**.
  - ExpandedDNSConfig moves to GA and is enabled by default. With this feature enabled, DNS configurations can be expanded.

  - The alpha feature CRDValidationRatcheting is added. This feature allows CRs with failing validations to pass if a Patch or Update request does not alter any of the invalid fields.
  - **--concurrent-cron-job-syncs** is added to kube-controller-manager to configure the number of workers for the cron job controller.

API Changes and Removals

- **NetworkPolicyStatus** is removed. There is no status attribute in a network policy.
- **annotationbatch.kubernetes.io/cronJob-scheduled-timestamp** is added to job objects to indicate the creation time of a job.
- The **podReplacementPolicy** and **terminating** fields are added to job APIs. With these fields specified, once a previously created pod is terminated in a job, the job immediately starts a new pod to replace the pod. The new fields allow you to specify whether to replace the pod immediately after the previous pod is terminated (original behavior) or replace the pod after the existing pod is completely terminated (new behavior). This is an alpha feature, and you can enable it by turning on the **JobPodReplacementPolicy** feature gate in your cluster.
- The **BackoffLimitPerIndex** field is available in a job. Pods specified by a job share a backoff mechanism. When backoff times of the job reach the limit, this job is marked as failed and resources, including indexes that are not running, are cleared up. This field allows you to configure backoff limit for a single index. For details, see **Backoff limit per index**.
- The **ServedVersions** field is added to the **StorageVersion** API. This change is introduced by mixed version proxy. The new field is used to indicate a version that can be provided by the API server.
- **SelfSubjectReview** is added to **authentication.k8s.io/v1**, and **kubectl auth whoami** goes to GA.
- **LastPhaseTransitionTime** is added to **PersistentVolume**. The new field is used to store the last time when a volume changes to a different phase.
- **resizeStatus** in **PVC.Status** is replaced by **AllocatedResourceStatus**. The new field indicates the statuses of the storage resize operation. The default value is an empty string.
- If **hostNetwork** is set to **true** and ports are specified for a pod, the **hostport** field will be automatically configured.
- StatefulSet pods have the pod index set as a pod label **statefulset.kubernetes.io/pod-index**.
- **PodHasNetwork** in the **Condition** field of pods has been renamed to **PodReadyToStartContainers**. The new field specifies that containers are ready to start after the network, volumes, and sandbox pod have been created.
- A new configuration option **delayCacheUntilActive** is added to **KubeSchedulerConfiguration**. If **delayCacheUntilActive** is set to **true**, kube-scheduler on the leader will not cache scheduling information. This reduces the memory pressure of other master nodes, but slows down the failover speed after the leader failed.
- The **namespaceParamRef** field is added to **admissionregistration.k8s.io/v1alpha1.ValidatingAdmissionPolicy**.

- The **reason** and **fieldPath** fields are added to CRD validation rules to allow you to specify reason and field path after verification failed.
- The CEL expression of ValidatingAdmissionPolicy supports namespace access via namespaceObject.
- API groups ValidatingAdmissionPolicy and ValidatingAdmissionPolicyBinding are promoted to betav1.
- A ValidatingAdmissionPolicy now has its **messageExpression** field checked against resolved types.

Feature Gate and Command Line Parameter Changes and Removals

- **–short** is removed from kubelet. Therefore, the default output of **kubectl version** is the same as that of **kubectl version –short**.
- **--volume-host-cidr-denylist** and **--volume-host-allow-local-loopback** are removed from kube-controller-manager. **--volume-host-cidr-denylist** is a comma-separated list of CIDR ranges. Volume plugins at these IP addresses are not allowed. If **--volume-host-allow-local-loopback** is set to **false**, the local loopback IP address and the CIDR ranges specified in **--volume-host-cidr-denylist** are disabled.
- **--azure-container-registry-config** is deprecated in kubelet and will be deleted in later Kubernetes versions. Use **--image-credential-provider-config** and **--image-credential-provider-bin-dir** instead.
- **--lock-object-namespace** and **--lock-object-name** are removed from kube-scheduler. Use **--leader-elect-resource-namespace** and **--leader-elect-resource-name** or **ComponentConfig** instead. (**--lock-object-namespace** is used to define the namespace of a lock object, and **--lock-object-name** is used to define the name of a lock object.)
- KMS v1 is deprecated and will only receive security updates. Use KMS v2 instead. In later Kubernetes versions, use **--feature-gates=KMSv1=true** to configure a KMS v1 provider.
- The DelegateFSGroupToCSIDriver, DevicePlugins, KubeletCredentialProviders, MixedProtocolLBService, ServiceInternalTrafficPolicy, ServiceIPStaticSubrange, and EndpointSliceTerminatingCondition feature gates are removed.

Enhanced Kubernetes 1.28 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.28 and provides enhanced functions.

For details about cluster version updates, see **Release Notes for CCE Cluster Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.28 and other versions, see **Kubernetes v1.28 Release Notes**.

2.2.1.5 Kubernetes 1.27 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create clusters of Kubernetes 1.27. This

section describes the changes made in Kubernetes 1.27 compared with Kubernetes 1.25.

Indexes

New Features

Kubernetes 1.27

- SeccompDefault is stable.

  To use SeccompDefault, add the **--seccomp-default command line flag** using kubelet on each node. If this feature is enabled, the **RuntimeDefault** profile will be used for all workloads by default, instead of the **Unconfined** (seccomp disabled) profile.

- Jobs' scheduling directives are configurable.

  This feature was introduced in Kubernetes 1.22 and is stable in Kubernetes 1.27. In most cases, you use a job to influence where the pods will run, like all in the same AZ. This feature allows scheduling directives to be modified before a job starts. You can use the **suspend** field to suspend a job. In the suspension phase, the scheduling directives (such as the node selector, node affinity, anti-affinity, and tolerations) in the job's pod template can be modified. For details, see **Mutable Scheduling Directives**.

- Downward API hugepages are stable.

  In Kubernetes 1.20, **requests.hugepages-***<pagesize>* and **limits.hugepages-***<pagesize>* were introduced to the **downward API**. Requests and limits can be configured for hugepages like other resources.

- Pod scheduling readiness moves to beta.

  After a pod is created, the Kubernetes scheduler selects an appropriate node to run the pod in the pending state. In practice, some pods may stay in the pending state for a long period due to insufficient resources. These pods may affect the running of other components like Cluster Autoscaler in the cluster. By specifying or deleting **.spec. schedulingGates** for a pod, you can control when the pod is ready for scheduling. For details, see **Pod Scheduling Readiness**.

- Accessing node logs using Kubernetes APIs is supported.

  This function is in the alpha phase. The cluster administrator can directly query node logs to help debug malfunctioning services running on the node. To use this function, ensure that the NodeLogQuery **feature gate** is enabled for that node and the kubelet configuration options **enableSystemLogHandler** and **enableSystemLogQuery** are set to **true**.

- ReadWriteOncePod access mode moves to beta.

  Kubernetes 1.22 introduced a ReadWriteOncePod access mode for PVs and PVCs. This feature has evolved into the beta phase. A volume can be mounted to a single pod in read/write mode. Use this access mode if you want to

ensure that only one pod in the cluster can read that PVC or write to it. For details, see **Access Modes**.

- The **matchLabelKeys** field in the pod topology spread constraint moves to beta.

  matchLabelKeys is a list of pod label keys. It is used to select a group of pods over which spreading will be calculated. With **matchLabelKeys**, you do not need to update **pod.spec** between different revisions. The controller or operator just needs to set different values to the same label key for different revisions. The scheduler will automatically determine the values based on **matchLabelKeys**. For details, see **Pod Topology Distribution Constraints**.

- The function of efficiently labeling SELinux volumes moves to beta.

  By default, the container runtime recursively assigns the SELinux label to all files on all pod volumes. To speed up this process, Kubernetes uses the mount option **-o context=***<label>* to immediately change the SELinux label of the volume. For details, see **Efficient SELinux volume relabeling**.

- VolumeManager reconstruction goes to beta.

  After the VolumeManager is reconstructed, if the NewVolumeManagerReconstruction **feature gate** is enabled, mounted volumes will be obtained in a more effective way during kubelet startup.

- Server side field validation and OpenAPI V3 are stable.

  OpenAPI V3 was added in Kubernetes 1.23. In Kubernetes 1.24, it moved to beta. In Kubernetes 1.27, it is stable.

- StatefulSet start ordinal moves to beta.

  Kubernetes 1.26 introduced a new, alpha-level feature for StatefulSets to control the ordinal numbering of pod replicas. Since Kubernetes 1.27, this feature moves to beta. The ordinals can start from arbitrary non-negative numbers. For details, see **Kubernetes 1.27: StatefulSet Start Ordinal Simplifies Migration**.

- **ContainerResource** metric in HorizontalPodAutoscaler moves to beta.

  Kubernetes 1.20 introduced the **ContainerResource** metric in HorizontalPodAutoscaler (HPA). In Kubernetes 1.27, this feature moves to beta, and the HPAContainerMetrics feature gate is enabled by default.

- StatefulSet PVC auto deletion moves to beta.

  Kubernetes 1.27 provides a new policy to control the lifecycle of PVCs of StatefulSets. This policy allows users to specify if the PVCs generated from the StatefulSet spec template should be automatically deleted or retained when the StatefulSet is deleted or replicas in the StatefulSet are scaled down. For details, see **PersistentVolumeClaim retention**.

- Volume group snapshots are introduced.

  Volume group snapshots are introduced as an alpha feature in Kubernetes 1.27. This feature allows users to create snapshots for multiple volumes to ensure data consistency when a fault occurs. It uses a label selector to group multiple PVCs for snapshot. This feature only supports CSI volume drivers. For details, see **Kubernetes 1.27: Introducing an API for Volume Group Snapshots**.

- **kubectl apply** pruning is more secure and efficient.

  In Kubernetes 1.5, the **--prune** flag was introduced in **kubectl apply** to delete resources that are no longer needed. This allowed **kubectl apply** to

automatically clear resources removed from the current configuration. However, the existing implementation of **--prune** has design defects that degrade its performance and lead to unexpected behaviors. In Kubernetes 1.27, **kubectl apply** provides ApplySet-based pruning, which is in the alpha phase. For details, see **Declarative Management of Kubernetes Objects Using Configuration Files**.

- Conflicts during port allocation to NodePort Service can be avoided.

  In Kubernetes 1.27, you can enable a new **feature gate** ServiceNodePortStaticSubrange to use different port allocation policies for NodePort Services. This mitigates the risk of port conflicts. This feature is in the alpha phase.

- Resizing resources assigned to pods without restarting the containers is supported.

  Kubernetes 1.27 allows users to resize CPU and memory resources assigned to pods without restarting the container. This feature is in the alpha phase. For details, see **Kubernetes 1.27: In-place Resource Resize for Kubernetes Pods (alpha)**.

- Pod startup is accelerated.

  A series of parameter adjustments like parallel image pulls and increased default API query limit for kubelet per second are made in Kubernetes 1.27 to accelerate pod startup. For details, see **Kubernetes 1.27: updates on speeding up Pod startup**.

- KMS V2 moves to beta.

  The key management KMS V2 API goes to beta. This has greatly improved the performance of the KMS encryption provider. For details, see **Using a KMS provider for data encryption**.

Kubernetes 1.26

- CRI v1alpha2 is removed.

  Kubernetes 1.26 does not support CRI v1alpha2 any longer. Use CRI v1 (containerd version must be later than or equal to 1.5.0). containerd 1.5.x or earlier is not supported by Kubernetes 1.26. Update the containerd version to 1.6.x or later before upgrading kubelet to 1.26.

  NOTE

  The containerd version used by CCE is 1.6.14, which meets the requirements. If the existing nodes do not meet the containerd version requirements, reset them to the latest version.

- Alpha API for dynamic resource allocation is added.

  In Kubernetes 1.26, **Dynamic Resource Allocation** is added to request and share resources between pods and between containers in a pod. Resources can be initialized based on parameters provided by the user. This function is still in the alpha phase. You need to enable the DynamicResourceAllocation feature gate and the **resource.k8s.io/v1alpha1** API group. You need to install drivers for specific resources to be managed. For details, see **Kubernetes 1.26: Alpha API for Dynamic Resource Allocation**.

- The non-graceful node shutdown feature goes to beta.

  In Kubernetes 1.26, the non-graceful node shutdown feature goes to beta and is enabled by default. A node shutdown can be graceful only if the kubelet's

node shutdown manager can detect the upcoming node shutdown action. For details, see **Non-graceful node shutdown handling**.

- Passing pod **fsGroup** to CSI drivers during mounting is supported.

  In Kubernetes 1.22, delegation of **fsGroup** to CSI drivers was first introduced as an alpha feature. In Kubernetes 1.25, it moved to beta. In Kubernetes 1.26, this feature enters the official release phase. For details, see **Delegating volume permission and ownership change to CSI driver**.

- Pod scheduling readiness is introduced.

  Kubernetes 1.26 introduces a new feature schedulingGates, which enables the scheduler to detect when pod scheduling can be performed. For details, see **Pod Scheduling Readiness**.

- CPU manager is officially released.

  The CPU manager is a part of kubelet. Since Kubernetes 1.10, it has moved to **beta**. The CPU manager can allocate exclusive CPUs to containers. This feature is stable in Kubernetes 1.26. For details, see **Control CPU Management Policies on the Node**.

- Kubernetes traffic engineering is advanced.

  Internal node-local traffic optimization and **EndpointSlice conditions** are upgraded to the official release version. **ProxyTerminatingEndpoints** moves to beta.

- Cross-namespace volume data sources are supported.

  This feature allows you to specify a data source that belongs to different namespaces for a PVC. This feature is in the alpha phase. For details, see **Cross namespace data sources**.

- Retroactive default StorageClass assignment moves to beta.

  In Kubernetes 1.25, an alpha feature was introduced to change the way how a default StorageClass is allocated to a PVC. After this feature is enabled, you no longer need to create a default StorageClass and then create a PVC to assign the class. Additionally, any PVCs without a StorageClass assigned can be updated later. This feature moves to beta in Kubernetes 1.26. For details, see **Retroactive default StorageClass assignment**.

- PodDisruptionBudget allows users to specify the eviction policies for unhealthy pods.

  You are allowed to specify unhealthy pod eviction policies for **PodDisruptionBudget** (PDB). This feature helps ensure node availability during node management. This feature is in the beta phase. For details, see **Unhealthy Pod Eviction Policy**.

- The number of Horizontal Pod Autoscaler (HPA) can be configured.

  kube-controller-manager allows **--concurrent-horizontal-pod-autoscaler-syncs** to configure the number of worker nodes of the pod autoscaler for horizontal scaling. For details, see **Cluster Configuration Management**.

Deprecations and Removals

Kubernetes 1.27

- In Kubernetes 1.27, the feature gates that are used for volume extension and in the GA status, including ExpandCSIVolumes,

ExpandInUsePersistentVolumes, and ExpandPersistentVolumes are removed and can no longer be referenced in the **--feature-gates** flag.

- The **--master-service-namespace** parameter is removed. This parameter specifies where to create a Service named **kubernetes** to represent the API server. This parameter was deprecated in Kubernetes 1.26 and is removed from Kubernetes 1.27.
- The ControllerManagerLeaderMigration feature gate is removed. **Leader Migration** provides a mechanism for HA clusters to safely migrate "cloud specific" controllers using a resource lock shared between kube-controller-manager and cloud-controller-manager when upgrading the replicated control plane. This feature has been enabled unconditionally since its release in Kubernetes 1.24. In Kubernetes 1.27, this feature is removed.
- The **--enable-taint-manager** parameter is removed. The feature that it supports, taint-based eviction, is enabled by default and will continue to be implicitly enabled when the flag is removed.
- The **--pod-eviction-timeout** parameter is removed from kube-controller-manager.
- The CSIMigration feature gate is removed. The **CSI migration** program allows smooth migration from the in-tree volume plug-ins to the out-of-tree CSI drivers. This feature was officially released in Kubernetes 1.16.
- The CSIInlineVolume feature gate is removed. The feature (**CSI Ephemeral Volume**) allows CSI volumes to be specified directly in the pod specification for ephemeral use cases. They can be used to inject arbitrary states, such as configuration, secrets, identity, variables, or similar information, directly inside the pod using a mounted volume. This feature graduated to GA in Kubernetes 1.25 and is removed in Kubernetes 1.27.
- The EphemeralContainers feature gate is removed. For Kubernetes 1.27, API support for ephemeral containers is unconditionally enabled.
- The LocalStorageCapacityIsolation feature gate is removed. This feature gate (**Local Ephemeral Storage Capacity Isolation**) moved to GA in Kubernetes 1.25. The feature provides support for capacity isolation of local ephemeral storage between pods, such as emptyDir volumes, so that a pod can be limited in its consumption of shared resources. kubelet will evict a pod if its consumption of local ephemeral storage exceeds the configured limit.
- The NetworkPolicyEndPort feature gate is removed. In Kubernetes 1.25, **endPort** in NetworkPolicy moved to GA. NetworkPolicy providers that support the **endPort** field can be used to specify a range of ports to apply NetworkPolicy.
- The StatefulSetMinReadySeconds feature gate is removed. For a pod that is part of a StatefulSet, Kubernetes marks the pod as read-only when the pod is available (and passes the check) at least within the period specified in **minReadySeconds**. This feature was officially released in Kubernetes 1.25. It is locked to **true** and removed from Kubernetes 1.27.
- The IdentifyPodOS feature gate is removed. If this feature is enabled, you can specify an OS for a pod. It has been stable since Kubernetes 1.25. This feature is removed from Kubernetes 1.27.
- The DaemonSetUpdateSurge feature gate is removed. In Kubernetes 1.25, this feature was stable. It was implemented to minimize DaemonSet downtime during deployment, but it is removed from Kubernetes 1.27.

- The **--container-runtime** parameter is removed. kubelet accepts a deprecated parameter **--container-runtime**, and the only valid value will be **remote** after the dockershim code is removed. This parameter was deprecated in 1.24 and later versions and is removed from Kubernetes 1.27.

Kubernetes 1.26

- HorizontalPodAutoscaler API for v2beta2 is removed.

  The autoscaling/v2beta2 API of HorizontalPodAutoscaler is no longer available in Kubernetes 1.26. For details, see **Removed APIs by release**. Use autoscaling/v2 API instead.

- The **flowcontrol.apiserver.k8s.io/v1beta1** API is removed.

  In Kubernetes 1.26 and later versions, the API of the **flowcontrol.apiserver.k8s.io/v1beta1** version for FlowSchema and PriorityLevelConfiguration is no longer served. For details, see **Removed APIs by release**. The **flowcontrol.apiserver.k8s.io/v1beta2** version is available in Kubernetes 1.23 and later versions, and the **flowcontrol.apiserver.k8s.io/v1beta3** version is available in Kubernetes 1.26 and later versions.

- The cloud service vendors' in-tree storage drivers are removed.
- The kube-proxy userspace mode is removed.

  The deprecated userspace mode is no longer supported by Linux or Windows. Linux users can use Iptables or IPVS, and Windows users can use the Kernelspace mode. Errors are returned if you use **--mode userspace**.

  – Windows winkernel kube-proxy no longer supports Windows HNS v1 APIs.

- **--prune-whitelist** flag is deprecated.

  The **--prune-whitelist** flag is **deprecated** and replaced by **--prune-allowlist** to support **Inclusive Naming Initiative**. This deprecated flag will be completely removed in later versions.

- The DynamicKubeletConfig feature gate is removed.

  The kubelet configuration of nodes can be dynamically updated through the API. The feature gate is removed from the kubelet in Kubernetes 1.24 and removed from the API server in Kubernetes 1.26. This simplifies the code and improves stability. It is recommended that you modify the kubelet configuration file instead and then restart the kubelet. For details, see **Remove DynamicKubeletConfig feature gate from the code**.

- A kube-apiserver command line parameter is removed.

  The **--master-service-namespace** parameter is deprecated. It is unused in the API Server.

- Several **kubectl run** parameters are deprecated.

  Several unused kubectl subcommands are marked as **deprecated** and will be removed in later versions. These subcommands include **--cascade**, **--filename**, **--force**, **--grace-period**, **--kustomize**, **--recursive**, **--timeout**, and **--wait**.

- Some command line parameters related to logging are removed.

  Some logging-related command line parameters are **removed**. These parameters were **deprecated** in earlier versions.

Enhanced Kubernetes 1.27 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.27 and provides enhanced functions.

For details about cluster version updates, see **Release Notes for CCE Cluster Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.27 and other versions, see the following documents:

- **Kubernetes v1.27 Release Notes**
- **Kubernetes v1.26 Release Notes**

2.2.1.6 Kubernetes 1.25 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the changes made in Kubernetes 1.25 compared with Kubernetes 1.23.

Indexes

New Features

Kubernetes 1.25

- Pod Security Admission is stable. PodSecurityPolicy is deprecated.

  PodSecurityPolicy is replaced by Pod Security Admission. For details about the migration, see **Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller**.

- The ephemeral container is stable.

  An **ephemeral container** is a container that runs temporarily in an existing pod. It is useful for troubleshooting, especially when kubectl exec cannot be used to check a container that breaks down or its image lacks a debugging tool.

- Support for cgroups v2 enters the stable phase.

  Kubernetes supports cgroups v2. cgroups v2 provides some improvements over cgroup v1. For details, see **About cgroup v2**.

- SeccompDefault moves to beta.

  To enable this feature, add the startup parameter **--seccomp-default=true** to kubelet. In this way, **seccomp** is set to **RuntimeDefault** by default, improving system security. Clusters of v1.25 no longer support **seccomp.security.alpha.kubernetes.io/pod** and **container.seccomp.security.alpha.kubernetes.io/annotation**. Replace them

with the **securityContext.seccompProfile** field in pods or containers. For details, see **Configure a Security Context for a Pod or Container**.

NOTE

After this feature is enabled, the system calls required by the application may be restricted by the runtime. Ensure that the debugging is performed in the test environment, so that application is not affected.

- The EndPort in the network policy moves to stable.

  EndPort in Network Policy is stable. This feature is incorporated in version 1.21. EndPort is added to NetworkPolicy. You can specify a port range.

- Local ephemeral storage capacity isolation is stable.

  This feature provides support for capacity isolation of local ephemeral storage between pods, such as emptyDir. If a pod's consumption of shared resources exceeds the limit, it will be evicted.

- The CRD verification expression language moves to beta.

  This makes it possible to declare how to validate custom resources using **CEL**. For details, see **Extend the Kubernetes API with CustomResourceDefinitions**.

- KMS v2 APIs are introduced.

  The KMS v2 alpha1 API is introduced to add performance, rotation, and observability improvements. This API uses AES-GCM to replace AES-CBC and uses DEK to encrypt data at rest (Kubernetes Secrets). No additional operation is required during this process. Additionally, data can be read through AES-GCM and AES-CBC. For details, see **Using a KMS provider for data encryption**.

- Pod network readiness is introduced.

  Kubernetes 1.25 introduces Alpha support for PodHasNetwork. This status is in the **status** field of the pod. For details, see **Pod network readiness**.

- The two features used for application rollout are stable.
  - In Kubernetes 1.25, **minReadySeconds** for StatefulSets is stable. It allows each pod to wait for an expected period of time to slow down the rollout of a StatefulSet. For details, see **Minimum ready seconds**.
  - In Kubernetes 1.25, **maxSurge** for DaemonSets is stable. It allows a DaemonSet workload to run multiple instances of the same pod on one node during a rollout. This minimizes DaemonSet downtime for users. DaemonSet does not allow **maxSurge** and **hostPort** to be used at the same time because two active pods cannot share the same port on the same node. For details, see **Perform a Rolling Update on a DaemonSet**.

- Alpha support for running pods with user namespaces is provided.

  This feature maps the **root** user in a pod to a non-zero ID outside the container. In this way, the container runs as the **root** user and the node runs as a regular unprivileged user. This feature is still in the internal test phase. The UserNamespacesStatelessPodsSupport gate needs to be enabled, and the container runtime must support this function. For details, see **Kubernetes 1.25: alpha support for running Pods with user namespaces**.

Kubernetes 1.24

- Dockershim is removed from kubelet.

Dockershim was marked deprecated in Kubernetes 1.20 and officially removed from kubelet in Kubernetes 1.24. If you want to use Docker container, switch to cri-dockerd or other runtimes that support CRI, such as containerd and CRI-O.

For details about how to switch from Docker to containerd, see **Migrating Nodes from Docker to containerd**.

NOTE

Check whether there are agents or applications that depend on Docker Engine. For example, if **docker ps**, **docker run**, and **docker inspect** are used, ensure that multiple runtimes are compatible and switch to the standard CRI.

- Beta APIs are disabled by default.

  The Kubernetes community found 90% cluster administrators did not care about the beta APIs and left them enabled. However, the beta features are not recommended because these APIs enabled in the production environment by default incur risks. Therefore, in 1.24 and later versions, beta APIs are disabled by default, but the existing beta APIs will retain the original settings.

- OpenAPI v3 is supported.

  In Kubernetes 1.24 and later versions, OpenAPI V3 is enabled by default.

- Storage capacity tracking is stable.

  In Kubernetes 1.24 and later versions, the CSIStorageCapacity API supports exposing the available storage capacity. This ensures that pods are scheduled to nodes with sufficient storage capacity, which reduces pod scheduling delay caused by volume creation and mounting failures. For details, see **Storage Capacity**.

- gRPC container probe moves to beta.

  In Kubernetes 1.24 and later versions, the gRPC probe goes to beta. The feature gate GRPCContainerProbe is available by default. For details about how to use this probe, see **Configure Probes**.

- LegacyServiceAccountTokenNoAutoGeneration is enabled by default.

  LegacyServiceAccountTokenNoAutoGeneration moves to beta. By default, this feature is enabled, where no secret token is automatically generated for a service account. To use a token that never expires, create a secret to hold the token. For details, see **Service account token Secrets**.

- IP address conflict is prevented.

  In Kubernetes 1.24, **an IP address pool is soft reserved for the static IP addresses of Services**. After you manually enable this function, Service IP addresses will be automatically from the IP address pool to minimize IP address conflict.

- Clusters are compiled based on Go 1.18.

  Kubernetes clusters of versions later than 1.24 are compiled based on Go 1.18. By default, the SHA-1 hash algorithm, such as SHA1WithRSA and ECDSAWithSHA1, is no longer supported for certificate signature verification. Use the certificate generated by the SHA256 algorithm instead.

- The maximum number of unavailable StatefulSet replicas is configurable.

  In Kubernetes 1.24 and later versions, the **maxUnavailable** parameter can be configured for StatefulSets so that pods can be stopped more quickly during a rolling update.

- Alpha support for non-graceful node shutdown is introduced.

  The non-graceful node shutdown is introduced as alpha in Kubernetes v1.24. A node shutdown is considered graceful only if kubelet's node shutdown manager can detect the upcoming node shutdown action. For details, see **Non-graceful node shutdown handling**.

Deprecations and Removals

Kubernetes 1.25

- The iptables chain ownership is cleared up.

  Kubernetes typically creates iptables chains to ensure data packets can be sent to the destination. These iptables chains and their names are for internal use only. These chains were never intended to be part of any Kubernetes API/ABI guarantees. For details, see **Kubernetes's IPTables Chains Are Not API**.

  In versions later than Kubernetes 1.25, Kubelet uses IPTablesCleanup to migrate the Kubernetes-generated iptables chains used by the components outside of Kubernetes in phases so that iptables chains such as KUBE-MARK-DROP, KUBE-MARK-MASQ, and KUBE-POSTROUTING will not be created in the NAT table. For more details, see **Cleaning Up IPTables Chain Ownership**.

- In-tree volume drivers from cloud service vendors are removed.

Kubernetes 1.24

- In Kubernetes 1.24 and later versions, Service.Spec.LoadBalancerIP is deprecated because it cannot be used for dual-stack protocols. Instead, use custom annotations.
- In Kubernetes 1.24 and later versions, the **--address**, **--insecure-bind-address**, **--port**, and **--insecure-port=0** parameters are removed from **kube-apiserver**.
- In Kubernetes 1.24 and later versions, startup parameters **--port=0** and **--address** are removed from **kube-controller-manager** and **kube-scheduler**.
- In Kubernetes 1.24 and later versions, **kube-apiserver --audit-log-version** and **--audit-webhook-version** support only **audit.k8s.io/v1**. In Kubernetes 1.24, **audit.k8s.io/v1[alpha|beta]1** is removed, and only **audit.k8s.io/v1** can be used.
- In Kubernetes 1.24 and later versions, the startup parameter **--network-plugin** is removed from kubelet. This Docker-specific parameter is available only when the container runtime environment is **Docker** and it is deleted with Dockershim.
- In Kubernetes 1.24 and later versions, dynamic log clearance has been discarded and removed accordingly. A log filter is introduced to the logs of all Kubernetes system components to prevent sensitive information from being leaked through logs. However, this function may block logs and therefore is discarded. For more details, see **Dynamic log sanitization** and **KEP-1753**.
- VolumeSnapshot v1beta1 CRD is discarded in Kubernetes 1.20 and removed in Kubernetes 1.24. Use VolumeSnapshot v1 instead.
- In Kubernetes 1.24 and later versions, **service annotation tolerate-unready-endpoints** discarded in Kubernetes 1.11 is replaced by **Service.spec.publishNotReadyAddresses**.

- In Kubernetes 1.24 and later versions, the **metadata.clusterName** field is discarded and will be deleted in the next version.
- In Kubernetes 1.24 and later versions, the logic for kube-proxy to listen to NodePorts is removed. If NodePorts conflict with **kernel net.ipv4.ip_local_port_range**, TCP connections may fail occasionally, which leads to a health check failure or service exception. Before the upgrade, ensure that cluster NodePorts do not conflict with **net.ipv4.ip_local_port_range** of all nodes in the cluster. For more details, see **Kubernetes PR**.

Enhanced Kubernetes 1.25 on CCE

During a version maintenance period, CCE periodically updates Kubernetes 1.25 and provides enhanced functions.

For details about cluster version updates, see **Release Notes for CCE Cluster Versions**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.25 and other versions, see the following documents:

- **Kubernetes v1.25 Release Notes**
- **Kubernetes v1.24 Release Notes**

2.2.1.7 Kubernetes 1.23 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.23.

Resource Changes and Deprecations

Kubernetes v1.23 Release Notes

- FlexVolume is deprecated. Use CSI.
- HorizontalPodAutoscaler v2 is promoted to GA, and HorizontalPodAutoscaler API v2 is gradually stable in version 1.23. The HorizontalPodAutoscaler v2beta2 API is not recommended. Use the v2 API.
- **PodSecurity** moves to beta, replacing the deprecated PodSecurityPolicy. PodSecurity is an admission controller that enforces pod security standards on pods in the namespace based on specific namespace labels that set the enforcement level. PodSecurity is enabled by default in version 1.23.

Kubernetes v1.22 Release Notes

- Ingresses no longer support networking.k8s.io/v1beta1 and extensions/v1beta1 APIs. If you use the API of an earlier version to manage ingresses, an application cannot be exposed to external services. Use networking.k8s.io/v1.
- CustomResourceDefinitions no longer support the apiextensions.k8s.io/v1beta1 API. If you use the API of an earlier version to create a CRD, the creation will fail, which affects the controller that reconciles this CRD. Use apiextensions.k8s.io/v1.

- ClusterRoles, ClusterRoleBindings, Roles, and RoleBindings no longer support the rbac.authorization.k8s.io/v1beta1 API. If you use the API of an earlier version to manage RBAC resources, application permissions control is affected and even cannot work in the cluster. Use rbac.authorization.k8s.io/v1.
- The Kubernetes release cycle is changed from four releases a year to three releases a year.
- StatefulSets support **minReadySeconds**.
- During scale-in, pods are randomly selected and deleted based on the pod UID by default (LogarithmicScaleDown). This feature enhances the randomness of the pods to be deleted and alleviates the problems caused by pod topology spread constraints. For more information, see **KEP-2185** and **issue 96748**.
- The **BoundServiceAccountTokenVolume** feature is stable, which has changed the method of mounting tokens into pods for enhanced token security of the service account. This feature is enabled by default in Kubernetes clusters of v1.21 and later versions.

References

For more details about the performance comparison and function evolution between Kubernetes 1.23 and other versions, see the following documents:

- **Kubernetes v1.23 Release Notes**
- **Kubernetes v1.22 Release Notes**

2.2.1.8 Kubernetes 1.21 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.21.

Resource Changes and Deprecations

Kubernetes v1.21 Release Notes

- CronJob is now in the stable state, and the version number changes to batch/v1.
- The immutable Secret and ConfigMap have now been upgraded to the stable state. A new immutable field is added to these objects to reject changes. The rejection protects clusters from accidental updates that may cause application outages. As these resources are immutable, kubelet does not monitor or poll for changes. This reduces the load of kube-apiserver and improves scalability and performance of your clusters. For more information, see **Immutable ConfigMaps**.
- Graceful node shutdown has been upgraded to the test state. With this update, kubelet can detect that a node is shut down and gracefully terminate the pods on the node. Prior to this update, when the node was shut down, its pod did not follow the expected termination lifecycle, which caused workload problems. Now kubelet can use systemd to detect the systems that are about to be shut down and notify the running pods to terminate them gracefully.
- For a pod with multiple containers, you can use **kubectl.kubernetes.io/** to pre-select containers.

- PodSecurityPolicy is deprecated. For details, see **https://kubernetes.io/blog/2021/04/06/podsecuritypolicy-deprecation-past-present-and-future/**.
- The **BoundServiceAccountTokenVolume** feature is in beta testing, which has changed the method of mounting tokens into pods for enhanced token security of the service account. This feature is enabled by default in Kubernetes clusters of v1.21 and later versions.

Kubernetes v1.20 Release Notes

- The API priority and fairness have reached the test state and are enabled by default. This allows kube-apiserver to classify incoming requests by priority. For more information, see **API Priority and Fairness**.
- The bug of **exec probe timeouts** is fixed. Before this bug is fixed, the exec probe does not consider the **timeoutSeconds** field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. Now, if no value is specified, the default value is used, that is, one second. If the detection time exceeds one second, the application health check may fail. During the upgrade, update the **timeoutSeconds** field for the applications that use this feature. The repair provided by the newly introduced ExecProbeTimeout feature gating enables the cluster operator to restore the previous behavior, but this behavior will be locked and removed in later versions.
- RuntimeClass enters the stable state. RuntimeClass provides a mechanism to support multiple runtimes in a cluster and expose information about the container runtime to the control plane.
- kubectl debugging has reached the test state. kubectl debugging provides support for common debugging workflows.
- dockershim was marked as deprecated in Kubernetes 1.20. Currently, you can continue to use Docker in the cluster. This change is irrelevant to the container image used by clusters. You can still use Docker to build your images. For more information, see **Dockershim Deprecation FAQ**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.21 and other versions, see the following documents:

- **Kubernetes v1.21 Release Notes**
- **Kubernetes v1.20 Release Notes**

2.2.1.9 Kubernetes 1.19 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.19.

Resource Changes and Deprecations

Kubernetes v1.19 Release Notes

- vSphere in-tree volumes can be migrated to vSphere CSI drivers. The in-tree vSphere Volume plugin is no longer used and will be deleted in later versions.
- **apiextensions.k8s.io/v1beta1** has been deprecated. You are advised to use **apiextensions.k8s.io/v1**.

- **apiregistration.k8s.io/v1beta1** has been deprecated. You are advised to use **apiregistration.k8s.io/v1**.
- **authentication.k8s.io/v1beta1** and **authorization.k8s.io/v1beta1** have been deprecated and will be removed from Kubernetes 1.22. You are advised to use **authentication.k8s.io/v1** and **authorization.k8s.io/v1**.
- **autoscaling/v2beta1** has been deprecated. You are advised to use **autoscaling/v2beta2**.
- **coordination.k8s.io/v1beta1** has been deprecated in Kubernetes 1.19 and will be removed from version 1.22. You are advised to use **coordination.k8s.io/v1**.
- kube-apiserver: The **componentstatus** API has been deprecated.
- kubeadm: The **kubeadm config view** command has been deprecated and will be deleted in later versions. Use **kubectl get cm -o yaml -n kube-system kubeadm-config** to directly obtain the kubeadm configuration.
- kubeadm: The **kubeadm alpha kubelet config enable-dynamic** command has been deprecated.
- kubeadm: The **--use-api** flag in the **kubeadm alpha certs renew** command has been deprecated.
- Kubernetes no longer supports **hyperkube** image creation.
- The **--export** flag is removed from the **kubectl get** command.
- The alpha feature **ResourceLimitsPriorityFunction** has been deleted.
- **storage.k8s.io/v1beta1** has been deprecated. You are advised to use **storage.k8s.io/v1**.

Kubernetes v1.18 Release Notes

- kube-apiserver
  - All resources in the **apps/v1beta1** and **apps/v1beta2** API versions are no longer served. You can use the **apps/v1** API version.
  - DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1** API version are no longer served. You can use the **apps/v1** API version.
  - NetworkPolicies in the **extensions/v1beta1** API version are no longer served. You can use the **networking.k8s.io/v1** API version.
  - PodSecurityPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **policy/v1beta1** API version.
- kubelet
  - **--redirect-container-streaming** is not recommended and will be deprecated in v1.20.
  - The resource measurement endpoint **/metrics/resource/v1alpha1** and all measurement standards under this endpoint have been deprecated. Use the measurement standards under the endpoint **/metrics/resource** instead:
    - scrape_error --> scrape_error
    - node_cpu_usage_seconds_total --> node_cpu_usage_seconds
    - node_memory_working_set_bytes --> node_memory_working_set_bytes

        - container_cpu_usage_seconds_total --> container_cpu_usage_seconds
        - container_memory_working_set_bytes --> container_memory_working_set_bytes
        - scrape_error --> scrape_error
    - In future releases, kubelet will no longer create the target directory **CSI NodePublishVolume** according to the CSI specifications. You may need to update the CSI driver accordingly to correctly create and process the target path.
- kube-proxy
    - You are not advised to use the **--healthz-port** and **--metrics-port** flags. Use **--healthz-bind-address** and **--metrics-bind-address** instead.
    - The **EndpointSliceProxying** function option is added to control the use of EndpointSlices in kube-proxy. This function is disabled by default.
- kubeadm
    - The **--kubelet-version** flag of **kubeadm upgrade node** has been deprecated and will be deleted in later versions.
    - The **--use-api** flag in the **kubeadm alpha certs renew** command has been deprecated.
    - kube-dns has been deprecated and will no longer be supported in future versions.
    - The ClusterStatus structure in the kubeadm-config ConfigMap has been deprecated and will be deleted in later versions.
- kubectl
    - You are not advised to use boolean and unset values for **--dry-run**. **server|client|none** is used in the new version.
    - **--server-dry-run** has been deprecated for **kubectl apply** and replaced by **--dry-run=server**.
- add-ons

  The cluster-monitoring add-on is deleted.

- kube-scheduler
    - The **scheduling_duration_seconds** metric has been deprecated.
    - The **scheduling_algorithm_predicate_evaluation_seconds** and **scheduling_algorithm_priority_evaluation_seconds** metrics are no longer used and are replaced by **framework_extension_point_duration_seconds[extension_point="Filter"]** and **framework_extension_point_duration_seconds[extension_point="Score"]**.
    - The scheduler policy AlwaysCheckAllPredictes has been deprecated.
- Other changes
    - The k8s.io/node-api component is no longer updated. Instead, you can use the **RuntimeClass** type in **k8s.io/api** and the generated clients in **k8s.io/client-go**.

  - The **client** label has been deleted from **apiserver_request_total**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.19 and other versions, see the following documents:

- **Kubernetes v1.19.0 Release Notes**
- **Kubernetes v1.18.0 Release Notes**

2.2.1.10 Kubernetes 1.17 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.17.

Resource Changes and Deprecations

- All resources in the **apps/v1beta1** and **apps/v1beta2** API versions are no longer served. Migrate to use the **apps/v1** API version.
- DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1** API version are no longer served. You can use the **apps/v1** API version.
- NetworkPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **networking.k8s.io/v1** API version.
- PodSecurityPolicies in the **extensions/v1beta1** API version are no longer served. Migrate to use the **policy/v1beta1** API version.
- Ingresses in the **extensions/v1beta1** API version will no longer be served in v1.20. Migrate to use the **networking.k8s.io/v1beta1** API version.
- PriorityClass in the **scheduling.k8s.io/v1beta1** and **scheduling.k8s.io/v1alpha1** API versions is no longer served in v1.17. Migrate to use the **scheduling.k8s.io/v1** API version.
- The **event series.state** field in the **events.k8s.io/v1beta1** API version has been deprecated and will be removed from v1.18.
- **CustomResourceDefinition** in the **apiextensions.k8s.io/v1beta1** API version has been deprecated and will no longer be served in v1.19. Use the **apiextensions.k8s.io/v1** API version.
- **MutatingWebhookConfiguration** and **ValidatingWebhookConfiguration** in the **admissionregistration.k8s.io/v1beta1** API version have been deprecated and will no longer be served in v1.19. You can use the **admissionregistration.k8s.io/v1** API version.
- The **rbac.authorization.k8s.io/v1alpha1** and **rbac.authorization.k8s.io/v1beta1** API versions have been deprecated and will no longer be served in v1.20. Use the **rbac.authorization.k8s.io/v1** API version.
- The **CSINode** object of **storage.k8s.io/v1beta1** has been deprecated and will be removed in later versions.

Other Deprecations and Removals

- **OutOfDisk node condition** is removed in favor of **DiskPressure**.
- The **scheduler.alpha.kubernetes.io/critical-pod** annotation is removed in favor of **priorityClassName**.

- **beta.kubernetes.io/os** and **beta.kubernetes.io/arch** have been deprecated in v1.14 and will be removed in v1.18.
- Do not use **--node-labels** to set labels prefixed with **kubernetes.io** and **k8s.io**. The **kubernetes.io/availablezone** label in earlier versions is removed in v1.17 and changed to **failure-domain.beta.kubernetes.io/zone**.
- The **beta.kubernetes.io/instance-type** is deprecated in favor of **node.kubernetes.io/instance-type**.
- Remove the **{kubelet_root_dir}/plugins** path.
- Remove the built-in cluster roles **system:csi-external-provisioner** and **system:csi-external-attacher**.

References

For more details about the performance comparison and function evolution between Kubernetes 1.17 and other versions, see the following documents:

- **Kubernetes v1.17.0 Release Notes**
- **Kubernetes v1.16.0 Release Notes**

2.2.1.11 Kubernetes 1.15 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.15.

To enable interoperability from one Kubernetes installation to the next, you must upgrade your Kubernetes clusters before the maintenance period ends.

Description

CCE provides full-link component optimization and upgrade for Kubernetes v1.15, which includes two minor versions v1.15.11 and v1.15.6-r1.

Resource Changes and Deprecations

- Ingress in the **extensions/v1beta1** API version has been deprecated. It will be no longer served from Kubernetes 1.19. You can use the **networking.k8s.io/v1beta1** API version.
- NetworkPolicy in the **extensions/v1beta1** API version will be officially suspended in 1.16. Migrate to use the **networking.k8s.io/v1** API version.
- PodSecurityPolicy in the **extensions/v1beta1** API version will be officially suspended in 1.16. Migrate to use the **policy/v1beta1** API version.
- DaemonSets, Deployments, and ReplicaSets in the **extensions/v1beta1**, **apps/v1beta1**, and **apps/v1beta2** API versions will not be served in 1.16. You can use the **apps/v1** API version.
- PriorityClass is upgraded to **scheduling.k8s.io/v1**, **scheduling.k8s.io/v1beta1**, and **scheduling.k8s.io/v1alpha1**. It will be deprecated in 1.17.
- The **series.state** field in the **events.k8s.io/v1beta1** Event API version has been deprecated and will be removed from 1.18.

References

Changelog from v1.13 to v1.15

- Changelog from v1.14 to v1.15:

  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.15.md
- Changelog from v1.13 to v1.14:

  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.14.md

2.2.1.12 Kubernetes 1.13 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.13.

Table 2-2 v1.13 cluster description

Kubernetes (CCE Enhanced Version)	Description
v1.13.10-r0	**Highlights**: • Arm nodes can be added to a CCE cluster. • The load balancer name is configurable. • Layer-4 load balancing supports health check, and layer-7 load balancing supports health check, allocation policy, and sticky session. • BMS nodes can be created in a CCE cluster (when the tunnel network model is used). • Ascend-accelerated nodes (powered by HiSilicon Ascend 310 AI processors) apply to scenarios such as image recognition, video processing, inference computing, and machine learning. • The docker baseSize is configurable. • Namespace affinity scheduling is supported. • User space can be partitioned in node data disks. • Cluster CPU management policies can be configured. • Nodes in a cluster can be configured across subnets (when the tunnel network mode is used).
v1.13.7-r0	**Highlights**: • Features of Kubernetes v1.13.7 are incorporated. • The network attachment definition is supported.

References

Changelog from v1.11 to v1.13

- Changelog from v1.12 to v1.13:
  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.13.md
- Changelog from v1.11 to v1.12:
  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.12.md

2.2.1.13 Kubernetes 1.11 (EOM) Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.11.

Table 2-3 v1.11 cluster description

Kubernetes (CCE Enhanced Version)	Description
v1.11.7-r2	**Highlights**: • Support for GPU V100. • Support for permission management.
v1.11.7-r0	**Highlights**: • Features of Kubernetes v1.11.7 are incorporated. • Node pools, VMs, and Kunpeng clusters can be created. • BMS nodes can be created in a CCE cluster (when the VPC network model is used), and hybrid deployment of BMSs and VMs is supported. • Support for GPU V100. • AOM notifies users when alarms are generated for container clusters of v1.11. • Access type switching is supported for Services. • Service network segments can be configured. • The number of IP addresses allocated to a node in a cluster can be customized.
v1.11.3-r2	**Highlights**: • Clusters support IPv6 dual stack. • ELB load balancing algorithms: source IP hash and sticky sessions with backend servers.
v1.11.3-r1	**Highlights**: • Perl regular expressions can be used for matching ingress URLs.

Kubernetes (CCE Enhanced Version)	Description
v1.11.3-r0	**Highlights**: • Features of Kubernetes v1.11.3 are incorporated. • Master nodes of a cluster can be deployed across multiple AZs. • CCE works with SFS Turbo to provide container storage.

References

Changelog from v1.9 to v1.11

- Changelog from v1.10 to v1.11:
  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.11.md
- Changelog from v1.9 to v1.10:
  https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.10.md

2.2.1.14 Release Notes for Kubernetes 1.9 (EOM) and Earlier Versions

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. This section describes the updates in CCE Kubernetes 1.9 and earlier versions.

Table 2-4 Description of v1.9 and earlier version clusters

Kubernetes (CCE Enhanced Version)	Description
v1.9.10-r2	**Highlights**: • ELB load balancing algorithms: source IP hash and sticky sessions with backend servers.
v1.9.10-r1	**Highlights**: • CCE works with SFS. • Enhanced ELBs can be automatically created for Services. • Transparent transmission of source IP addresses is supported for enhanced ELBs on the public network. • The maximum number of pods on a node can be set.

Kubernetes (CCE Enhanced Version)	Description
v1.9.10-r0	**Highlights**: ● Use of ELB/ingress for Kubernetes clusters; new traffic control mechanism. ● Features of Kubernetes v1.9.10 are incorporated. ● Kubernetes RBAC capability authorization is supported. **Fault rectification**: ● Occasional memory leak on nodes, which is caused by kernel cgroup bugs.
v1.9.7-r1	**Highlights**: ● The mechanism for reporting PVC and PV events is enhanced. Events can be viewed on the PVC details page. ● CCE works with a third-party authentication system. ● Physical machines that use EulerOS 2.3 can be managed. ● Data disk allocation can be user-defined. ● Elastic Volume Service (EVS) disks are supported for BMSs. ● InfiniBand NICs are supported for BMSs. ● Nodes can be created using the CM-v3 API in BMS scenarios.
v1.9.7-r0	**Highlights**: ● The Docker version of new clusters is upgraded to v17.06. ● DNS cascading is supported. ● Add-ons can be managed. ● Features of Kubernetes v1.9.7 are incorporated. ● The HTTPS of layer-7 ingress is supported. ● StatefulSets can be migrated, scheduled, updated, and upgraded.

Kubernetes (CCE Enhanced Version)	Description
v1.9.2-r3	**Highlights**: ● Cluster nodes that use CentOS 7.4 can be created or managed. ● DNAT Services are supported. ● NetworkPolicy APIs are provided. ● Multiple ports can be configured for a Kubernetes Service that uses an ELB. **Fault rectification**: ● Incomplete pod resource recycling caused by a disconnection with kube-apiserver ● Data inaccuracy during auto node scaling
v1.9.2-r2	**Highlights**: ● Custom health check ports can be configured for classic load balancers. ● Performance of classic load balancers is enhanced. ● Kubernetes Service ports can be configured for layer-4 load balancing. **Fault rectification**: ● Bugs in network add-ons, which cause deadlocks in health checks. ● A limited number of HAProxy connections in an HA cluster.
v1.9.2-r1	**Highlights**: ● Features of Kubernetes v1.9.2 are incorporated. ● Cluster nodes support CentOS 7.1. ● GPU nodes are supported and GPU resource use can be restricted. ● The web-terminal add-on is supported.
v1.7.3-r13	**Highlights**: ● The Docker version of new clusters is upgraded to v17.06. ● DNS cascading is supported. ● Add-ons can be managed. ● The mechanism for reporting PVC and PV events is enhanced. ● OBS is supported for BMS clusters.

Kubernetes (CCE Enhanced Version)	Description
v1.7.3-r12	**Highlights**: • Cluster nodes that use CentOS 7.4 can be created or managed. • DNAT Services are supported. • NetworkPolicy APIs are provided. • Multiple ports can be configured for a Kubernetes Service that uses an ELB. **Fault rectification**: • Incomplete pod resource recycling caused by a disconnection with kube-apiserver • Data inaccuracy during auto node scaling • The event aging period prompt is modified. The cluster aging period is 1 hour.
v1.7.3-r11	**Highlights**: • Custom health check ports can be configured for classic load balancers. • Performance of classic load balancers is enhanced. • Kubernetes Service ports can be configured for layer-4 load balancing. • Namespaces can be deleted. • EVS disks can be unbound. • Migration policies can be configured. **Fault rectification**: • Bugs in network add-ons, which cause deadlocks in health checks. • A limited number of HAProxy connections in an HA cluster.
v1.7.3-r10	**Highlights**: • Overlay L2 container networks are supported. • Cluster nodes can be GPU-accelerated VMs. • Cluster nodes support CentOS 7.1 and the operating system can be selected. • Windows clusters support ELBs. • CCE allows exporting data from SFS. • BMS clusters support SFS.

Kubernetes (CCE Enhanced Version)	Description
v1.7.3-r9	**Highlights**: ● Cross-AZ deployment is supported for workloads. ● Containers support OBS. ● Layer-7 load balancing is supported. ● Windows clusters support EVS. ● Device mapper in direct-lvm mode is supported in BMS scenarios.
v1.7.3-r8	**Highlights**: ● Auto scaling is supported for cluster nodes. ● Arm nodes can be managed.
v1.7.3-r7	**Highlights**: ● SUSE 12 sp2 nodes can be managed in the container clusters (in the tunnel network mode). ● Docker supports the device mapper in direct-lvm mode. ● Clusters support the dashboard add-on. ● Windows clusters can be created.
v1.7.3-r6	**Highlights**: ● Native EVS APIs are supported for clusters.
v1.7.3-r5	**Highlights**: ● HA clusters can be created. **Fault rectification**: ● Container network disconnection after a node restart
v1.7.3-r4	**Highlights**: ● Cluster performance is enhanced. ● Interconnection with ELB is allowed in BMS scenarios.
v1.7.3-r3	**Highlights**: ● Storage can be attached to kernel-based virtual machines (KVMs).
v1.7.3-r2	**Highlights**: ● SFS is supported to provide container storage. ● Custom logs can be configured for workloads. ● Graceful scaling-in is supported for workloads. **Fault rectification**: ● Expiration of Access Key ID/Secret Access Key (AK/SK) of container storage volumes.

Kubernetes (CCE Enhanced Version)	Description
v1.7.3-r1	**Highlights**: • External domain names can be resolved by kube-dns.
v1.7.3-r0	**Highlights**: • Features of Kubernetes v1.7.3 are incorporated. • Elastic Load Balance (ELB) is supported. • Storage can be attached to Xen VMs. • EVS is supported to provide container storage.

2.2.2 Patch Version Release Notes

Indexes

Version 1.31

Table 2-5 Release notes for the v1.31 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.31.1-r0	v1.31.1	CCE clusters of v1.31 are released for the first time. For more information, see **Kubernetes 1.31 Release Notes**. • LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	None

Version 1.30

Table 2-6 Release notes for the v1.30 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.30.6-r0	v1.30.6	• LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	Fixed some security issues.
v1.30.4-r4	v1.30.4	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.30.4-r2	v1.30.4	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.30.4-r0	v1.30.4	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.
v1.30.1-r2	v1.30.2	None	Enhanced system stability.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.30.1-r0	v1.30.2	CCE clusters of v1.30 are released for the first time. For more information, see **Kubernetes 1.30 Release Notes**. • When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.

Version 1.29

Table 2-7 Release notes for the v1.29 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.10-r0	v1.29.10	• LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	Fixed some security issues.
v1.29.8-r4	v1.29.8	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.29.8-r2	v1.29.8	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.8-r0	v1.29.8	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.
v1.29.4-r2	v1.29.3	None	Enhanced system stability.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.4-r0	v1.29.3	• When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.
v1.29.3-r0	v1.29.3	• RAM allows secrets to be shared between accounts. • Nodes can be migrated to a custom node pool. • FlexusX is supported.	• The default containerd image path has been included in the management of node pool configurations. • On the node pool list page, you can sort node pools according to your preferences.	Fixed some security issues.
v1.29.2-r4	v1.29.3	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.2-r2	v1.29.3	None	During cluster upgrades, the collection of container logs has been made more reliable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.29.2-r0	v1.29.3	• CCE ingresses support traffic distribution based on custom HTTP headers. • Scaling priority policies can be configured for third-party workloads. • You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters. • You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.	• An in-progress node drainage can be canceled. • When creating a node pool, you do not need to configure its billing mode. • When updating a node pool, you can change its agency name, prefix, and suffix. • Resetting a node on the console will reserve the node's Kubernetes labels and taints by default. • Both the Kubernetes service account token volume projection and the load scaling controller	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
			can be configured.	
v1.29.1-r10	v1.29.1	• CCE ingress forwarding policies can be sorted by priority. • When configuring volumeClaimTemplates for StatefulSets, you can configure the name prefixes of PVs and underlying storage. (The Everest version must be 2.4.15 or later.) • A yearly/monthly-billed node pool supports multiple node flavors. • The default node flavor can be deleted when a node pool is updated. • Nodes can be managed in a custom node pool.	• An alarm rule has been added for a CoreDNS resolution failure. • Nodes can be drained before they are unsubscribed from. • After a node pool is updated, the configuration differences of each node are displayed.	Fixed some security issues.
v1.29.1-r0	v1.29.1	CCE clusters of v1.29 are released for the first time. For more information, see **Kubernetes 1.29 Release Notes**.	None	None

Version 1.28

Table 2-8 Release notes for the v1.28 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.15-r0	v1.28.15	• LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	Fixed some security issues.
v1.28.13-r4	v1.28.13	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.28.13-r2	v1.28.13	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.13-r0	v1.28.13	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.
v1.28.8-r2	v1.28.8	None	Enhanced system stability.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.8-r0	v1.28.8	• When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.
v1.28.7-r2	v1.28.8	• RAM allows secrets to be shared between accounts. • Nodes can be migrated to a custom node pool. • FlexusX is supported.	• The default containerd image path has been included in the management of node pool configurations. • On the node pool list page, you can sort node pools according to your preferences.	Fixed some security issues.
v1.28.6-r4	v1.28.8	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.6-r2	v1.28.8	None	During cluster upgrades, the collection of container logs has been made more reliable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.6-r0	v1.28.8	• CCE ingresses support traffic distribution based on custom HTTP headers. • Scaling priority policies can be configured for third-party workloads. • You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters. • You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.	• An in-progress node drainage can be canceled. • When creating a node pool, you do not need to configure its billing mode. • When updating a node pool, you can change its agency name, prefix, and suffix. • Resetting a node on the console will reserve the node's Kubernetes labels and taints by default. • Both the Kubernetes service account token volume projection and the load scaling controller	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
			can be configured.	
v1.28.5-r0	v1.28.5	• CCE ingress forwarding policies can be sorted by priority. • When configuring volumeClaimTemplates for StatefulSets, you can configure the name prefixes of PVs and underlying storage. (The Everest version must be 2.4.15 or later.) • A yearly/monthly-billed node pool supports multiple node flavors. • The default node flavor can be deleted when a node pool is updated. • Nodes can be managed in a custom node pool.	• An alarm rule has been added for a CoreDNS resolution failure. • Nodes can be drained before they are unsubscribed from. • After a node pool is updated, the configuration differences of each node are displayed.	Fixed some security issues.
v1.28.4-r0	v1.28.5	• Docker can be selected when you create a node. • General-purpose SSD v2 EVS disks are available. • LoadBalancer ingresses support grayscale release. • LoadBalancer ingresses allow URL redirection and rewriting, as well as HTTP-to-HTTPS redirection.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.3-r0	v1.28.3	LoadBalancer Services and ingresses allow you to: • Configure SNI. • Enable HTTP/2. • Configure idle timeout, request timeout, and response timeout. • Obtain the listener port number and the number of the port requested by the client from the request header of an HTTP packet, and rewrite **X-Forwarded-Host**.	None	Fixed some security issues.
v1.28.2-r0	v1.28.3	• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress. • CCE node images support security hardening.	None	Fixed some security issues.
v1.28.1-r4	v1.28.3	None	None	Fixed CVE-2024-21626 issues.
v1.28.1-r2	v1.28.3	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.28.1-r0	v1.28.3	CCE clusters of v1.28 are released for the first time. For more information, see **Kubernetes 1.28 Release Notes**. • The prefix and suffix of a node name can be customized in node pools. • CCE Turbo clusters support container network configurations for workloads and allow you to specify pod subnets. • LoadBalancer ingresses support gRPC. • ELB private IP addresses can be specified when you create a LoadBalancer Service using YAML.	• Accelerated the startup speed for creating a large number of secure containers in a CCE Turbo cluster. • Improved the stability when secure containers are repeatedly created or deleted in a CCE Turbo cluster.	None

Version 1.27

Table 2-9 Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.16-r10	v1.27.16	• LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	Fixed some security issues.
v1.27.16-r4	v1.27.16	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.27.16-r2	v1.27.16	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.16-r0	v1.27.16	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.
v1.27.10-r2	v1.27.12	None	Enhanced system stability.	Fixed some security issues.
v1.27.10-r0	v1.27.12	• When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.9-r0	v1.27.12	• RAM allows secrets to be shared between accounts. • Nodes can be migrated to a custom node pool. • FlexusX is supported.	• The default containerd image path has been included in the management of node pool configurations. • On the node pool list page, you can sort node pools according to your preferences.	Fixed some security issues.
v1.27.8-r4	v1.27.12	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.
v1.27.8-r2	v1.27.12	None	During cluster upgrades, the collection of container logs has been made more reliable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.8-r0	v1.27.12	• CCE ingresses support traffic distribution based on custom HTTP headers. • Scaling priority policies can be configured for third-party workloads. • You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters. • You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.	• An in-progress node drainage can be canceled. • When creating a node pool, you do not need to configure its billing mode. • When updating a node pool, you can change its agency name, prefix, and suffix. • Resetting a node on the console will reserve the node's Kubernetes labels and taints by default. • Both the Kubernetes service account token volume projection and the load scaling controller can be configured.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.7-r0	v1.27.9	• CCE ingress forwarding policies can be sorted by priority. • When configuring volumeClaimTemplates for StatefulSets, you can configure the name prefixes of PVs and underlying storage. (The Everest version must be 2.4.15 or later.) • A yearly/monthly-billed node pool supports multiple node flavors. • The default node flavor can be deleted when a node pool is updated. • Nodes can be managed in a custom node pool.	• An alarm rule has been added for a CoreDNS resolution failure. • Nodes can be drained before they are unsubscribed from. • After a node pool is updated, the configuration differences of each node are displayed.	Fixed some security issues.
v1.27.6-r0	v1.27.9	• Docker can be selected when you create a node. • General-purpose SSD v2 EVS disks are available. • LoadBalancer ingresses support grayscale release. • LoadBalancer ingresses allow URL redirection and rewriting, as well as HTTP-to-HTTPS redirection.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.5-r0	v1.27.4	LoadBalancer Services and ingresses allow you to: • Configure SNI. • Enable HTTP/2. • Configure idle timeout, request timeout, and response timeout. • Obtain the listener port number and the number of the port requested by the client from the request header of an HTTP packet, and rewrite **X-Forwarded-Host**.	None	Fixed some security issues.
v1.27.4-r0	v1.27.4	• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress. • CCE node images support security hardening.	None	Fixed some security issues.
v1.27.3-r4	v1.27.4	None	None	Fixed **CVE-2024-21626** issues.
v1.27.3-r2	v1.27.4	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.3-r0	v1.27.4	• The prefix and suffix of a node name can be customized in node pools. • CCE Turbo clusters support container network configurations for workloads and allow you to specify pod subnets. • LoadBalancer ingresses support gRPC. • ELB private IP addresses can be specified when you create a LoadBalancer Service using YAML.	• Accelerated the startup speed for creating a large number of secure containers in a CCE Turbo cluster. • Improved the stability when secure containers are repeatedly created or deleted in a CCE Turbo cluster. • Added certificate verification when an ingress object is created. This prevents the ingress certificates already available on the ELB from being overwritten. • Fixed the issue of repeatedly reporting flavor sold-out events when a node pool is scaled out. • Added the logic for mutually checking the occupation of service and ingress ports, as well as the logic for checking the conflict of ingress paths in a cluster.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.27.2-r0	v1.27.2	• Volcano supports node pool affinity scheduling. • Volcano supports workload rescheduling.	None	Fixed some security issues.
v1.27.1-r10	v1.27.2	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.27.1-r0	v1.27.2	CCE clusters of v1.27 are released for the first time. For more information, see **Kubernetes 1.27 Release Notes**. • Both soft eviction and hard eviction are supported in node pool configurations. • TMS tags can be added to automatically created EVS disks to facilitate cost management.	The cluster IP address in clusters of v1.27 or later cannot be pinged due to **security hardening**.	None

Version 1.25

NOTICE

In CCE clusters of v1.25, containerd is the default runtime for nodes, except for nodes running EulerOS 2.5. In addition, clusters of v1.25 or later no longer support EulerOS 2.5.

Table 2-10 Release notes for the v1.25 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.16-r10	v1.25.16	• LoadBalancer ingresses support more advanced forwarding actions and return fixed responses. • DataPlane V2 is available for newly created CCE standard and Turbo clusters that use VPC networks. After DataPlane V2 is enabled, you can configure network policies for these clusters.	• During a cluster upgrade, you can scale out the nodes in the cluster. • You can choose multiple blocklists or trustlists to manage access to a LoadBalancer ingress.	Fixed some security issues.
v1.25.16-r4	v1.25.16	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.25.16-r2	v1.25.16	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.16-r0	v1.25.16	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.
v1.25.13-r2	v1.25.16	None	Enhanced system stability.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.13-r0	v1.25.16	• When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.
v1.25.12-r0	v1.25.16	• RAM allows secrets to be shared between accounts. • Nodes can be migrated to a custom node pool. • FlexusX is supported.	• The default containerd image path has been included in the management of node pool configurations. • On the node pool list page, you can sort node pools according to your preferences.	Fixed some security issues.
v1.25.11-r4	v1.25.16	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.11-r2	v1.25.16	None	During cluster upgrades, the collection of container logs has been made more reliable.	Fixed some security issues.
v1.25.11-r0	v1.25.16	• CCE ingresses support traffic distribution based on custom HTTP headers. • Scaling priority policies can be configured for third-party workloads. • You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters. • You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.	• An in-progress node drainage can be canceled. • When creating a node pool, you do not need to configure its billing mode. • When updating a node pool, you can change its agency name, prefix, and suffix. • Resetting a node on the console will reserve the node's Kubernetes labels and taints by default. • Both the Kubernetes service account token volume projection and the load scaling controller can be configured.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.10-r0	v1.25.16	• CCE ingress forwarding policies can be sorted by priority. • When configuring volumeClaimTemplates for StatefulSets, you can configure the name prefixes of PVs and underlying storage. (The Everest version must be 2.4.15 or later.) • A yearly/monthly-billed node pool supports multiple node flavors. • The default node flavor can be deleted when a node pool is updated. • Nodes can be managed in a custom node pool.	• An alarm rule has been added for a CoreDNS resolution failure. • Nodes can be drained before they are unsubscribed from. • After a node pool is updated, the configuration differences of each node are displayed.	Fixed some security issues.
v1.25.9-r0	v1.25.16	• General-purpose SSD v2 EVS disks are available. • LoadBalancer ingresses support grayscale release. • LoadBalancer ingresses allow URL redirection and rewriting, as well as HTTP-to-HTTPS redirection.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.8-r0	v1.25.10	LoadBalancer Services and ingresses allow you to: • Configure SNI. • Enable HTTP/2. • Configure idle timeout, request timeout, and response timeout. • Obtain the listener port number and the number of the port requested by the client from the request header of an HTTP packet, and rewrite **X-Forwarded-Host**.	None	Fixed some security issues.
v1.25.7-r0	v1.25.10	• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress. • CCE node images support security hardening.	None	Fixed some security issues.
v1.25.6-r4	v1.25.10	None	None	Fixed CVE-2024-21626 issues.
v1.25.6-r2	v1.25.10	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.6-r0	v1.25.10	• The prefix and suffix of a node name can be customized in node pools. • CCE Turbo clusters support container network configurations for workloads and allow you to specify pod subnets. • LoadBalancer ingresses support gRPC. • ELB private IP addresses can be specified when you create a LoadBalancer Service using YAML.	• Accelerated the startup speed for creating a large number of secure containers in a CCE Turbo cluster. • Improved the stability when secure containers are repeatedly created or deleted in a CCE Turbo cluster. • Fixed the issue that kubelet occasionally stops responding during startup in certain scenarios. • Fixed the issue of repeatedly reporting flavor sold-out events when a node pool is scaled out. • Fixed the issue that the state of pods is changed from **Succeed** to **Failed** when kubelet is restarted in certain scenarios.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.5-r0	v1.25.5	• Volcano supports node pool affinity scheduling. • Volcano supports workload rescheduling.	None	Fixed some security issues.
v1.25.4-r10	v1.25.5	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.25.4-r0	v1.25.5	• Both soft eviction and hard eviction are supported in node pool configurations. • TMS tags can be added to automatically created EVS disks to facilitate cost management.	None	Fixed some security issues.
v1.25.3-r10	v1.25.5	• CCE clusters support the dedicated load balancers that use elastic specifications. • The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.3-r0	v1.25.5	• CCE Turbo ENIs support fixed IP addresses. For details, see **Configuring a Static IP Address for a Pod**. • CCE Turbo ENIs support automatic allocation and binding of EIPs. For details, see **Configuring a Static IP Address for a Pod**. • Enhanced hybrid deployment of CCE Turbo clusters: The egress network bandwidth is guaranteed by network priority. For details, see **Egress Network Bandwidth Guarantee**. • CCE Turbo clusters support the association between namespaces and container CIDR blocks. For details, see **NetworkAttachmentDefinition**. • CPU Burst is supported to prevent CPU traffic limiting from affecting latency-sensitive services. For details, see **CPU Burst**.	Enhanced network stability of CCE Turbo clusters when their specifications are modified.	Fixed some security issues.
v1.25.1-r0	v1.25.5	CCE clusters of v1.25 are released for the first time. For more information, see **Kubernetes 1.25 Release Notes**.	None	None

Version 1.23

Table 2-11 Release notes for the v1.23 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.18-r16	v1.23.18	None	None	Fixed some security issues.
v1.23.18-r14	v1.23.18	None	• Improved the stability of cluster upgrades. • Optimized clusters' overload protection.	Fixed some security issues.
v1.23.18-r12	v1.23.18	None	Fixed the issue of chart upload failures in certain situations.	Fixed some security issues.
v1.23.18-r10	v1.23.18	• LoadBalancer ingresses can forward requests based on parameters such as HTTP request methods, HTTP request headers, query strings, CIDR blocks, and cookies. • LoadBalancer ingresses support cross-origin access.	• You can change a node password when updating its node pool. • A node can be attached with no data disks. • When updating a LoadBalancer ingress, you can modify the configuration of redirecting HTTP requests to HTTPS requests. • The default image address can be customized for Docker node pools.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.18-r2	v1.23.17	None	Enhanced system stability.	Fixed some security issues.
v1.23.18-r0	v1.23.17	• When deleting a cluster, CCE enables you to select which log groups to delete. • When creating a cluster, you can select bring-your-own KMS instances for secret-encrypted etcd. • When a node is created using a private image, the image password can be retained. • CCE supports GPU rendering.	CCE can handle ELB listeners on all ports.	Fixed some security issues.
v1.23.17-r0	v1.23.17	• RAM allows secrets to be shared between accounts. • Nodes can be migrated to a custom node pool. • FlexusX is supported.	• The default containerd image path has been included in the management of node pool configurations. • On the node pool list page, you can sort node pools according to your preferences.	Fixed some security issues.
v1.23.16-r4	v1.23.17	None	The stability of ELB has been improved during upgrades that span across multiple versions.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.16-r2	v1.23.17	None	During cluster upgrades, the collection of container logs has been made more reliable.	Fixed some security issues.
v1.23.16-r0	v1.23.17	• CCE ingresses support traffic distribution based on custom HTTP headers. • Scaling priority policies can be configured for third-party workloads. • You can configure a security group for a pod using annotations. This feature is only available for CCE Turbo clusters. • You can bind an existing EIP to a pod. This feature is only available for CCE Turbo clusters.	• An in-progress node drainage can be canceled. • When creating a node pool, you do not need to configure its billing mode. • When updating a node pool, you can change its agency name, prefix, and suffix. • Resetting a node on the console will reserve the node's Kubernetes labels and taints by default. • Both the Kubernetes service account token volume projection and the load scaling controller can be configured.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.15-r0	v1.23.17	• CCE ingress forwarding policies can be sorted by priority. • When configuring volumeClaimTemplates for StatefulSets, you can configure the name prefixes of PVs and underlying storage. (The Everest version must be 2.4.15 or later.) • A yearly/monthly-billed node pool supports multiple node flavors. • The default node flavor can be deleted when a node pool is updated. • Nodes can be managed in a custom node pool.	• An alarm rule has been added for a CoreDNS resolution failure. • Nodes can be drained before they are unsubscribed from. • After a node pool is updated, the configuration differences of each node are displayed.	Fixed some security issues.
v1.23.14-r0	v1.23.17	• General-purpose SSD v2 EVS disks are available. • LoadBalancer ingresses support grayscale release. • LoadBalancer ingresses allow URL redirection and rewriting, as well as HTTP-to-HTTPS redirection.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.13-r0	v1.23.17	LoadBalancer Services and ingresses allow you to: • Configure SNI. • Enable HTTP/2. • Configure idle timeout, request timeout, and response timeout. • Obtain the listener port number and the number of the port requested by the client from the request header of an HTTP packet, and rewrite **X-Forwarded-Host**.	None	Fixed some security issues.
v1.23.12-r0	v1.23.17	• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress. • CCE node images support security hardening.	None	Fixed some security issues.
v1.23.11-r4	v1.23.17	None	None	Fixed **CVE-2024-21626** issues.
v1.23.11-r2	v1.23.17	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.11-r0	v1.23.17	• The prefix and suffix of a node name can be customized in node pools. • CCE Turbo clusters support container network configurations for workloads and allow you to specify pod subnets. • LoadBalancer ingresses support gRPC. • ELB private IP addresses can be specified when you create a LoadBalancer Service using YAML.	• Accelerated the startup speed for creating a large number of secure containers in a CCE Turbo cluster. • Improved the stability when secure containers are repeatedly created or deleted in a CCE Turbo cluster. • Fixed the issue that Docker containers cannot be ended when journald exits unexpectedly. • Fixed the issue that the scheduler fails to intercept the automatic mounting of SFS 3.0 volumes during pod creation when Everest is uninstalled. • Fixed the issue that the usage of the **/var/lib/contained** disk directory is falsely high on a containerd node running EulerOS 2.9 in a v1.23 cluster.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.10-r0	v1.23.11	• Volcano supports node pool affinity scheduling. • Volcano supports workload rescheduling.	None	Fixed some security issues.
v1.23.9-r10	v1.23.11	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.23.9-r0	v1.23.11	• Both soft eviction and hard eviction are supported in node pool configurations. • TMS tags can be added to automatically created EVS disks to facilitate cost management.	None	Fixed some security issues.
v1.23.8-r10	v1.23.11	• CCE clusters support the dedicated load balancers that use elastic specifications. • The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.8-r0	v1.23.11	• CCE Turbo ENIs support fixed IP addresses. For details, see **Configuring a Static IP Address for a Pod**. • CCE Turbo ENIs support automatic allocation and binding of EIPs. For details, see **Configuring a Static IP Address for a Pod**. • Enhanced hybrid deployment of CCE Turbo clusters: The egress network bandwidth is guaranteed by network priority. For details, see **Egress Network Bandwidth Guarantee**. • CCE Turbo clusters support the association between namespaces and container CIDR blocks. For details, see **NetworkAttachment-Definition**. • CPU Burst is supported to prevent CPU traffic limiting from affecting latency-sensitive services. For details, see **CPU Burst**.	• Enhanced Docker reliability during upgrades. • Optimized node time synchronization.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.7-r20	v1.23.11	None	• Enhanced the interconnection stability between Services/ingresses and load balancers. • Enhanced the reliability of nodes with multiple data disks attached.	Fixed some security issues.
v1.23.7-r10	v1.23.11	None	• Enhanced Docker reliability during upgrades. • Enhanced the reliability of containerd upon disconnections. • Hardened security for scenarios where an error occurred during kernel parameter optimization.	Fixed some security issues.
v1.23.7-r0	v1.23.11	• Services and ingresses support dedicated load balancers that are bound with a GEIP.	• Enhanced network stability of CCE Turbo clusters when their specifications are modified. • Enhanced the network stability of nginx-ingress-controller when the cluster is upgraded. • Optimized node time synchronization.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.6-r0	v1.23.11	• TCP/UDP ports can be configured for LoadBalancer Services. • Pod readiness gate is supported.	• Enhanced the flow table reliability when the underlying network malfunctions. • Enhanced the stability of the OS with a later kernel version in restart scenarios, for example, due to unexpected power-off. • Optimized cAdvisor GPU/NPU metrics.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.5-r0	v1.23.11	• Containers support SFS 3.0 for storage. • Fault detection and isolation are supported on GPU nodes. • Security groups can be customized by cluster. • CCE Turbo clusters support ENIs pre-binding by node. • Control plane logs can be collected. • Huawei-developed Huawei Cloud EulerOS 2.0 is supported. • containerd is supported. • CCE Turbo clusters support hybrid deployment and CPU tidal affinity.	• Upgraded the etcd version of the master node to the Kubernetes version 3.5.6. • Optimized the Service access performance on EulerOS 2.8 nodes. • Optimized scheduling so that pods are evenly distributed across AZs after pods are scaled in. • Optimized the memory usage of kube-apiserver when CRDs are frequently updated.	Fixed some security issues and the following CVE vulnerabilities: • CVE-2022-3294 • CVE-2022-3162 • CVE-2022-3172 • CVE-2021-25749
v1.23.4-r10	v1.23.4	None	Optimized the memory usage of kube-apiserver when CRDs are frequently updated.	Fixed some security issues.
v1.23.4-r0	v1.23.4	Arm nodes are supported.	None	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.23.3-r0	v1.23.4	• Monitoring metrics of master nodes are made available to tenants. • Sub-ENI pre-binding is supported to speed up the startup of the sub-ENIs in CCE Turbo clusters. • Backend server weights can be configured for LoadBalancer Services. • CCE clusters support cross-cluster deployment. • CCE Turbo clusters support hybrid deployment when VM nodes are used.	Enhanced reliability when secure containers are frequently created or deleted.	Fixed some security issues.
v1.23.1-r1	v1.23.4	Node resource reservation has been optimized so that resource exhaustion can be detected to improve node stability.	Node installation compatibility has been improved.	Fixed some security issues.
v1.23.1-r0	v1.23.4	CCE clusters of v1.23 are released for the first time. For more information, see **Kubernetes 1.23 Release Notes**.	None	None

Version 1.21

Table 2-12 Release notes for the v1.21 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.15-r0	v1.21.14	• General-purpose SSD v2 EVS disks are available. • LoadBalancer ingresses support grayscale release. • LoadBalancer ingresses allow URL redirection and rewriting, as well as HTTP-to-HTTPS redirection.	The configurations of frequently used cluster parameters and node pool parameters are publicly available.	Fixed some security issues.
v1.21.14-r0	v1.21.14	A PVC can be used to dynamically create and mount an SFS Turbo subdirectory.	None	Fixed some security issues.
v1.21.13-r0	v1.21.14	• You can configure an ELB blocklist/trustlist for access control when creating a Service or ingress. • CCE node images support security hardening.	None	Fixed some security issues.
v1.21.12-r4	v1.21.14	None	None	Fixed CVE-2024-21626 issues.
v1.21.12-r2	v1.21.14	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.12-r0	v1.21.14	The prefix and suffix of a node name can be customized in node pools.	• Optimized the health check configuration to prevent keepalived from repeatedly restarting. • Optimized the security policy cache and resolved ingress retry storms. • Fixed the issue of an active/standby switchover failure due to I/O suspension on the master node where the main process of cloud-controller-manager is deployed. • Fixed the issue of incorrect pod weight due to incorrect calculation for the number of terminating pods after a weight is configured for a Service.	Fixed some security issues.
v1.21.11-r20	v1.21.14	• Volcano supports node pool affinity scheduling. • Volcano supports workload rescheduling.	None	Fixed some security issues.
v1.21.11-r10	v1.21.14	None	Optimized the events generated during node pool scaling.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.11-r0	v1.21.14	• Both soft eviction and hard eviction are supported in node pool configurations. • TMS tags can be added to automatically created EVS disks to facilitate cost management.	None	Fixed some security issues.
v1.21.10-r10	v1.21.14	• CCE clusters support the dedicated load balancers that use elastic specifications. • The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.
v1.21.10-r0	v1.21.14	• CCE Turbo ENIs support fixed IP addresses. For details, see **Configuring a Static IP Address for a Pod**. • CCE Turbo ENIs support automatic allocation and binding of EIPs. For details, see **Configuring a Static IP Address for a Pod**.	• Enhanced Docker reliability during upgrades. • Optimized node time synchronization. • Enhanced the stability of the Docker runtime for pulling images after nodes are restarted.	Fixed some security issues.
v1.21.9-r0	v1.21.14	• Services and ingresses support dedicated load balancers that are bound with a GEIP.	Enhanced network stability of CCE Turbo clusters when their specifications are modified.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.8-r0	v1.21.14	• TCP/UDP ports can be configured for LoadBalancer Services. • Pod readiness gate is supported.	• Enhanced the flow table reliability when the underlying network malfunctions. • Enhanced the stability of the OS with a later kernel version in restart scenarios, for example, due to unexpected power-off. • Optimized cAdvisor GPU/NPU metrics.	Fixed some security issues.
v1.21.7-r0	v1.21.14	• Containers support SFS 3.0 for storage. • Fault detection and isolation are supported on GPU nodes. • Security groups can be customized by cluster. • CCE Turbo clusters support ENIs pre-binding by node. • Control plane logs can be collected.	Improved the stability of LoadBalancer Services/ingresses with a large number of connections.	Fixed some security issues and the following CVE vulnerabilities: • CVE-2022-3294 • CVE-2022-3162 • CVE-2022-3172
v1.21.6-r0	v1.21.7	Arm nodes are supported.	None	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.5-r10	v1.21.7	None	• Enhanced the allocation stability of a tunnel network when the nodes on the control plane are intermittently disconnected. • Hardened for OVS vulnerabilities.	Fixed some security issues.
v1.21.5-r0	v1.21.7	None	• Enhanced the stability of container tunnel networks during cluster upgrades. • Added constraints on pod topology distribution. • Enhanced the stability in disconnecting links when a node on the cluster control plane is powered off. • Enhanced the stability when volumes are concurrently mounted for pods.	Fixed some security issues.
v1.21.4-r10	v1.21.7	None	Enhanced the stability of EulerOS 2.9 NetworkManager.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.4-r0	v1.21.7	None	• Enhanced the stability of secure containers when an ENI driver is changed. • Enhances the stability of accessing LoadBalancer Services during workload upgrade and node scaling.	Fixed some security issues.
v1.21.3-r10	v1.21.7	None	Enhanced the stability of secure containers when an ENI driver is changed.	Fixed some security issues.
v1.21.3-r0	v1.21.7	None	CCE Turbo clusters support SNAT CIDR blocks.	Fixed some security issues.
v1.21.2-r10	v1.21.7	None	Enhanced the interconnection stability between Services and load balancers.	Fixed some security issues.
v1.21.2-r0	v1.21.7	Node resource reservation has been optimized so that resource exhaustion can be detected to improve node stability.	• Improved the cluster upgrade capability and reliability. • Optimized the local storage of containers to improve stability.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.21.1-r2	v1.21.7	• Container storage supports local PVs. • EulerOS 2.9 Kunpeng servers can be managed. • Both container tunnel networks and VPC networks support wide matching of an OS kernel version.	• Optimized the node installation process to enhance the reliability of node creation. • Optimized the kernel parameters of CentOS and EulerOS 2.5 to improve the OS performance.	Fixed some security issues.
v1.21.1-r1	v1.21.7	None	Container networks support wide matching of an OS kernel version.	Fixed some security issues.
v1.21.1-r0	v1.21.7	CCE clusters of v1.21 are released for the first time. For more information, see **Kubernetes 1.21 Release Notes**.	None	None

Version 1.19

Table 2-13 Release notes for the v1.19 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r84	v1.19.16	None	None	Fixed **CVE-2024-21626** issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r82	v1.19.16	None	Fixed the issue that configuration conflicts occasionally occur when an SNI certificate is configured on an ingress with HTTP/2 enabled.	None

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r80	v1.19.16	None	• Fixed the issue that a restarted BMS node is displayed as unavailable. • Optimized the logic for clearing VIP routes so that residual VIP routes are cleared preferentially. This prevents routes from being added again after NetworkManager is restarted. • Fixed the issue that backend ELB servers may fail to add when an IP address is reused during the rolling upgrade of pods in a CCE Turbo cluster that uses dedicated load balancers. • Resolved the issue of residual cache if a LoadBalancer Service is deleted after it is added to a health check queue. • Resolved the issue of continuously increasing Docker memory usage after the physical	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
			machine or switch where the master node is located is disconnected.	
v1.19.16-r60	v1.19.16	• Volcano supports node pool affinity scheduling. • Volcano supports workload rescheduling.	None	Fixed some security issues.
v1.19.16-r50	v1.19.16	None	Optimized the events generated during node pool scaling.	Fixed some security issues.
v1.19.16-r40	v1.19.16	• Both soft eviction and hard eviction are supported in node pool configurations. • TMS tags can be added to automatically created EVS disks to facilitate cost management.	None	Fixed some security issues.
v1.19.16-r30	v1.19.16	• CCE clusters support the dedicated load balancers that use elastic specifications. • The timeout interval can be configured for a load balancer.	High-frequency parameters of kube-apiserver are configurable.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r20	v1.19.16	• CCE Turbo ENIs support fixed IP addresses. For details, see **Configuring a Static IP Address for a Pod**. • CCE Turbo ENIs support automatic allocation and binding of EIPs. For details, see **Configuring a Static IP Address for a Pod**.	• Cloud Native Network 2.0 allows you to specify subnets for a namespace. • Enhanced the stability of the Docker runtime for pulling images after nodes are restarted. • Optimized the performance of CCE Turbo clusters in allocating ENIs if not all ENIs are pre-bound.	Fixed some security issues.
v1.19.16-r10	v1.19.16	None	Enhanced the interconnection stability between Services/ingresses and load balancers.	Fixed some security issues.
v1.19.16-r7	v1.19.16	None	• Enhanced Docker reliability during upgrades. • Optimized node time synchronization. • Enhanced the reliability of CCE Turbo clusters when ENIs are pre-bound.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r6	v1.19.16	• Services and ingresses support dedicated load balancers that are bound with a GEIP.	• Enhanced the stability of containerd configured with QoS. • URL rewriting policies can be configured and modified for ingresses. • The memory usage of kube-controller-manager is optimized when CRD resources are frequently updated.	Fixed some security issues.
v1.19.16-r5	v1.19.16	• TCP/UDP ports can be configured for LoadBalancer Services. • Pod readiness gate is supported.	• Enhanced the flow table reliability when the underlying network malfunctions. • Enhanced the stability of the OS with a later kernel version in restart scenarios, for example, due to unexpected power-off.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r4	v1.19.16	• Containers support SFS 3.0 for storage. • Fault detection and isolation are supported on GPU nodes. • Security groups can be customized by cluster. • CCE Turbo clusters support ENIs pre-binding by node.	• Scheduling is optimized on taint nodes. • Enhanced the long-term running stability of containerd when cores are bound. • Improved the stability of LoadBalancer Services/ingresses with a large number of connections. • Optimized the memory usage of kube-apiserver when CRDs are frequently updated.	Fixed some security issues and the following CVE vulnerabilities: • CVE-2022-3294 • CVE-2022-3162 • CVE-2022-3172

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r3	v1.19.16	None	• The image-pull-progress-deadline startup parameter can be reserved after an upgrade. • CCE Turbo clusters support customized ENI pre-binding. • Fixed the issue of inconsistent tunnel network allocation caused by intermittent disconnection between master nodes. • Enhanced the stability of clusters running in a tunnel network when the nodes on the control plane are intermittently disconnected.	Fixed some security issues.
v1.19.16-r2	v1.19.16	None	• Enhanced the stability in disconnecting links when a node on the cluster control plane is powered off. • Enhanced the stability when volumes are concurrently mounted for pods.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.19.16-r1	v1.19.16	None	Enhanced the stability of EulerOS 2.9 NetworkManager.	Fixed some security issues.
v1.19.16-r0	v1.19.16	None	Enhanced the stability in updating LoadBalancer Services when workloads are upgraded and nodes are scaled in or out.	Fixed some security issues and the following CVE vulnerabilities: • CVE-2021-25741 • CVE-2021-25737
v1.19.10-r0	v1.19.10	CCE clusters of v1.19 are released for the first time. For more information, see Kubernetes 1.19 Release Notes.	None	None

2.3 Buying a Cluster

2.3.1 Comparison Between Cluster Types

Comparison

CCE provides different types of clusters for you to select. The following table lists the differences between them.

Category	Subcategory	CCE Standard	CCE Turbo	CCE Autopilot
Positioning	-	Standard clusters that provide highly reliable and secure containers for commercial use	Next-generation clusters designed for Cloud Native 2.0, with accelerated compute, networking, and scheduling	Serverless clusters without user nodes and billed by actual CPU and memory usage In such clusters, no node deployment, management, or security maintenance is needed.
Application scenario	-	For users who expect to use container clusters to manage applications, obtain elastic compute resources, and enable simplified management on compute, network, and storage resources	For users who have higher requirements on performance, resource utilization, and full-scenario coverage	For users whose services suffer frequent traffic surges, such as users in the online education and e-commerce sectors
Specification difference	Network model	Cloud native 1.0 networks: for scenarios where requirements on performance are not high and there are not so many containers • Tunnel network • Virtual Private Cloud (VPC) network	Cloud Native 2.0 networks: for scenarios where requirements on performance are high and there are many containers Max networking scale: 2,000 nodes	Cloud Native 2.0 networks: for scenarios where requirements on performance are high and there are many containers
	hostPort	Supported	Not supported	Not supported
	Network performance	The container network is overlaid with the VPC network, causing certain performance loss.	The VPC network and container network are flattened into one for zero performance loss.	The VPC network and container network are flattened into one for zero performance loss.

Category	Subcategory	CCE Standard	CCE Turbo	CCE Autopilot
	Network isolation	• Tunnel network model: network policies supported for communications within a cluster • VPC network model: isolation not supported	Pods can be associated with security groups for isolation. This isolation policy, based on security groups, ensures consistent security isolation both within and outside of a cluster.	Pods can be associated with security groups for isolation. This isolation policy, based on security groups, ensures consistent security isolation both within and outside of a cluster.
	Container resource isolation	cgroups are used to isolate common containers.	• VM-level isolation is supported for secure containers that run only on physical machines. • cgroups are used to isolate common containers.	VM-level isolation
	Edge infrastructure management	Not supported	Management of CloudPond edge sites	Not supported

2.3.2 Buying a CCE Standard/Turbo Cluster

On the CCE console, you can easily create Kubernetes clusters. After a cluster is created, the master node is hosted by CCE. You only need to create worker nodes. In this way, you can implement cost-effective O&M and efficient service deployment.

Precautions

- After a cluster is created, the following items cannot be changed:
  - Cluster type
  - Number of master nodes in the cluster
  - AZ of a master node

- Network configurations of the cluster, such as the VPC, subnet, Service CIDR block, IPv6 settings, and kube-proxy (**service forwarding**) settings
- Network model. For example, change **Tunnel network** to **VPC network**.

Step 1: Log In to the CCE Console

Step 1 Log in to the **CCE console**.

Step 2 On the **Clusters** page, click **Buy Cluster**.

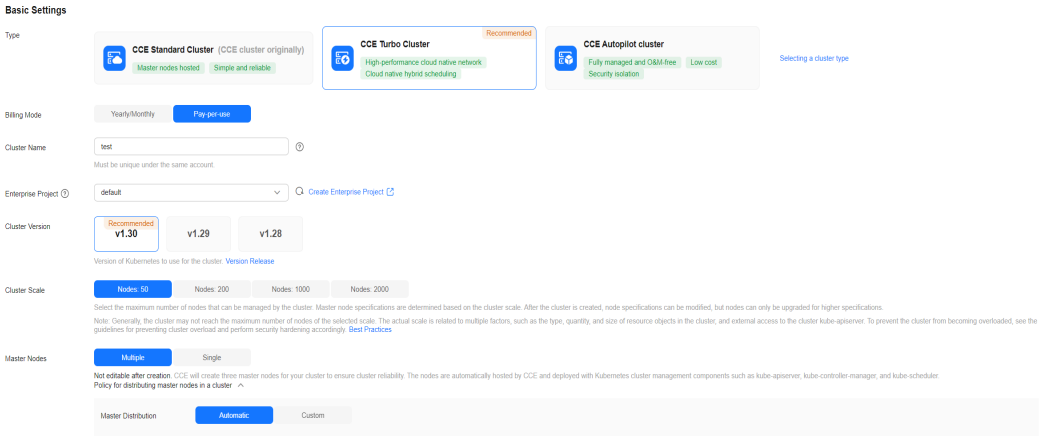
----End

Step 2: Configure the Cluster

On the **Buy Cluster** page, configure the parameters.

Basic Settings

Figure 2-2 Buying a CCE Turbo cluster



Parameter	Description
Type	Select **CCE Standard Cluster** or **CCE Turbo Cluster** as required. • CCE standard clusters provide highly reliable and secure containers for commercial use. • CCE Turbo clusters use the high-performance cloud native network. Such clusters provide cloud native hybrid scheduling, achieving higher resource utilization and wider scenario coverage. For more details, see cluster types.

Parameter	Description
Billing Mode	Select a billing mode for the cluster as required. • **Yearly/Monthly**: a prepaid billing mode. Resources will be billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. If you choose this billing mode, configure the required duration and determine whether to automatically renew the subscription. (If you purchase a monthly subscription, the automatic renewal period is one month. If you purchase a yearly subscription, the automatic renewal period is one year.) • **Pay-per-use**: a postpaid billing mode. It is suitable in scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.
Cluster Name	Enter a cluster name. Cluster names under the same account must be unique.
Enterprise Project	This parameter is available only for enterprise users who have enabled an enterprise project. After an enterprise project is selected, clusters and their security groups will be created in that project. To manage clusters and other resources like nodes, load balancers, and node security groups, you can use the Enterprise Project Management Service (EPS). For more details, see **Enterprise Management**.
Cluster Version	Select the Kubernetes version used by the cluster.
Cluster Scale	Select a cluster scale for your cluster as required. These values specify the maximum number of nodes that can be managed by the cluster. The cluster that has been created can only be scaled out. For details, see **Changing Cluster Scale**.

Parameter	Description
Master Nodes	Select the number of master nodes. The master nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler. • **3 Masters**: Three master nodes will be created for high cluster availability. • **Single**: Only one master node will be created in your cluster. **NOTE** • If more than half of the master nodes in a cluster are faulty, the cluster will not run properly. You can also select AZs for deploying the master nodes of a specific cluster. By default, AZs are allocated automatically for the master nodes. • **Automatic**: Master nodes are randomly distributed in different AZs for cluster DR. If the number of available AZs is less than the number of nodes to be created, CCE will create the nodes in the AZs with sufficient resources to preferentially ensure cluster creation. In this case, AZ-level DR may not be ensured. • **Custom**: Master nodes are deployed in specific AZs. If there is one master node in your cluster, you can select one AZ for the master node. If there are multiple master nodes in your cluster, you can select multiple AZs for the master nodes. – **AZ**: Master nodes are deployed in different AZs for cluster DR. – **Host**: Master nodes are deployed on different hosts in the same AZ for cluster DR. – **Custom**: Master nodes are deployed in the AZs you specified.

Network Settings

The network settings cover nodes, containers, and Services. For details about the cluster networking and container network models, see **Overview**.

Figure 2-3 Network settings

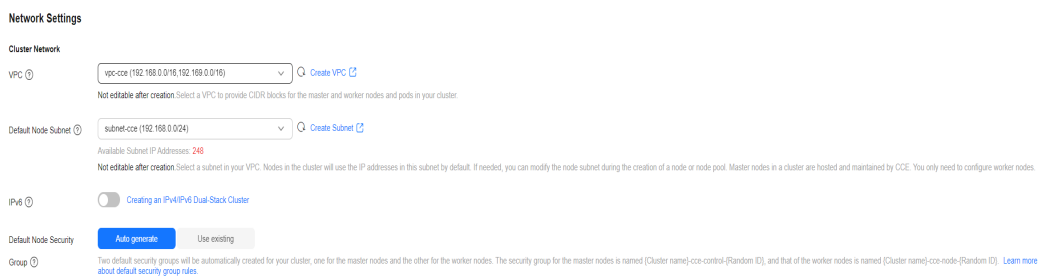


Table 2-14 Cluster network settings

Parameter	Description
VPC	Select the VPC to which the cluster belongs. If no VPC is available, click **Create VPC** to create one. The value cannot be changed after the cluster is created.
Node Subnet	Select the subnet to which the master nodes belong. If no subnet is available, click **Create Subnet** to create one. The value cannot be changed after the cluster is created.
Default Node Security Group	Select the security group automatically generated by CCE or use the existing one as the default security group of the node. **NOTICE** The default security group must allow traffic from certain ports to ensure normal communication. Otherwise, the node cannot be created. For details, see **How Can I Configure a Security Group Rule in a Cluster?**
IPv6	If enabled, cluster resources, including nodes and workloads, can be accessed through IPv6 CIDR blocks. • The IPv4/IPv6 dual stack is available for the CCE standard clusters (v1.15 and later) that use the container tunnel networks and later will be generally available for clusters of v1.23. • CCE Turbo clusters of v1.23.8-r0, v1.25.3-r0, and later versions support IPv4/IPv6 dual stack. • IPv4/IPv6 dual stack is not supported by clusters using the VPC networks. For details, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**.

Figure 2-4 Container network settings (CCE Turbo as an Example)

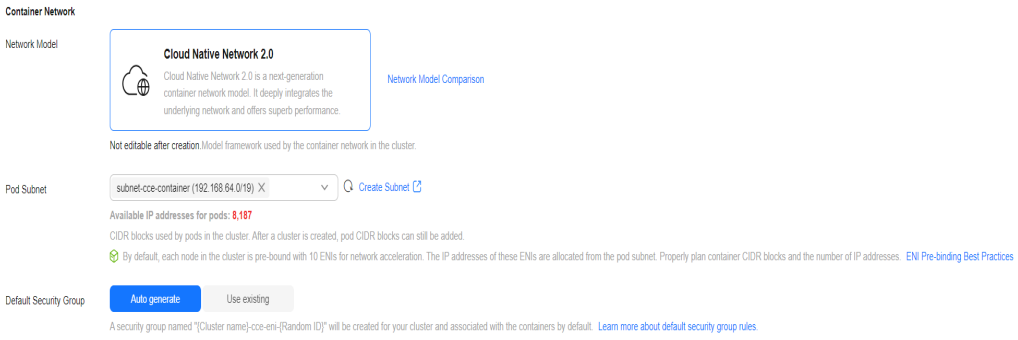


Table 2-15 Container network settings

Parameter	Description
Network Model	Select **VPC network** or **Tunnel network** for your CCE standard cluster. Select **Cloud Native Network 2.0** for your CCE Turbo cluster. For more information about their differences, see **Overview**.
DataPlane V2 (supported by CCE Turbo clusters)	DataPlane V2 uses eBPF to enable features like Service ClusterIP, NetworkPolicy, and egress bandwidth. For details, see **DataPlane V2 Network Acceleration**. This function is available for new clusters of v1.27 or later. After this function is enabled, CCE will automatically deploy the cilium-agent on every node in the cluster. Each cilium-agent will use 80 MiB of memory, and the memory usage will increase by 10 KiB whenever a new pod is added. After the cluster is created, you cannot disable this function. Additionally, the nodes can run only Huawei Cloud EulerOS 2.0, and **Guaranteed Egress Network Bandwidth** is unavailable in cloud native hybrid deployment. Therefore, enable this function only when you fully understand the constraints in **DataPlane V2 Network Acceleration**.
Network Policies (supported by CCE standard clusters using a tunnel network)	Policy-based network control for clusters. For details, see **Configuring Network Policies to Restrict Pod Access**. After this function is enabled, if the CIDR blocks of a customer's service conflict with the on-premises CIDR blocks, the link to a newly added gateway may not be established. For example, when a cluster accesses an external address through a Direct Connect connection, the external switch does not support **ip-option**. If the network policy is enabled, the network access may fail.
Container CIDR Block	Specify the CIDR block for containers, which determines the maximum number of containers allowed in the cluster. This parameter is available only for CCE standard clusters. You can add multiple container CIDR blocks to a VPC network-based cluster, even after the cluster has been created. For details, see **Adding a Container CIDR Block for a Cluster**.
Pod Subnet	Select the subnet to which the pod belongs. If no subnet is available, click **Create Subnet** to create one. This parameter is available only for CCE Turbo clusters. The pod subnet determines the maximum number of containers in a cluster. You can add pod subnets after a cluster is created.

Parameter	Description
Default Security Group	Select the security group automatically generated by CCE or use the existing one as the default security group of the containers. **NOTICE** The default security group of containers must allow access from specified ports to ensure proper communication between containers in the cluster. For details about how to configure security group ports, see **How Can I Configure a Security Group Rule in a Cluster?**

Figure 2-5 Service network settings

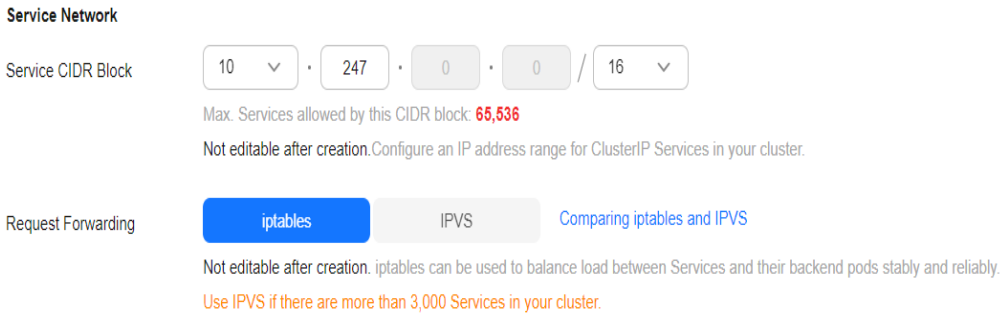
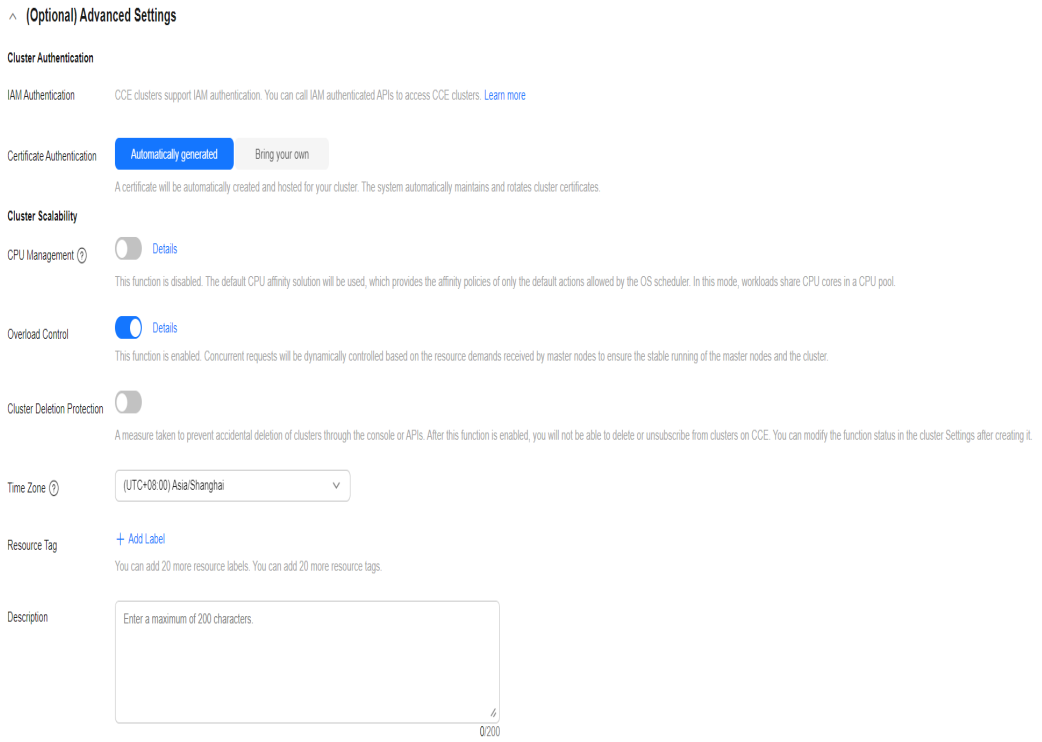


Table 2-16 Service network settings

Parameter	Description
Service CIDR Block	Configure the Service CIDR blocks for containers in the same cluster to access each other. The value determines the maximum number of Services you can create. The value cannot be changed after the cluster is created.
Request Forwarding	Select **IPVS** or **iptables** for your cluster. For details, see **Comparing iptables and IPVS**. • iptables is the traditional kube-proxy mode. This mode applies to the scenario where the number of Services is small or a large number of short connections are concurrently sent on the client. IPv6 clusters do not support iptables. • IPVS allows higher throughput and faster forwarding. This mode is suitable for large cluster scales or when there are a large number of Services.
IPv6 Service CIDR Block	Configure this parameter only when IPv6 dual stack is enabled for a CCE Turbo cluster. This configuration cannot be modified after the cluster is created.

(Optional) Advanced Settings

Figure 2-6 Advanced settings



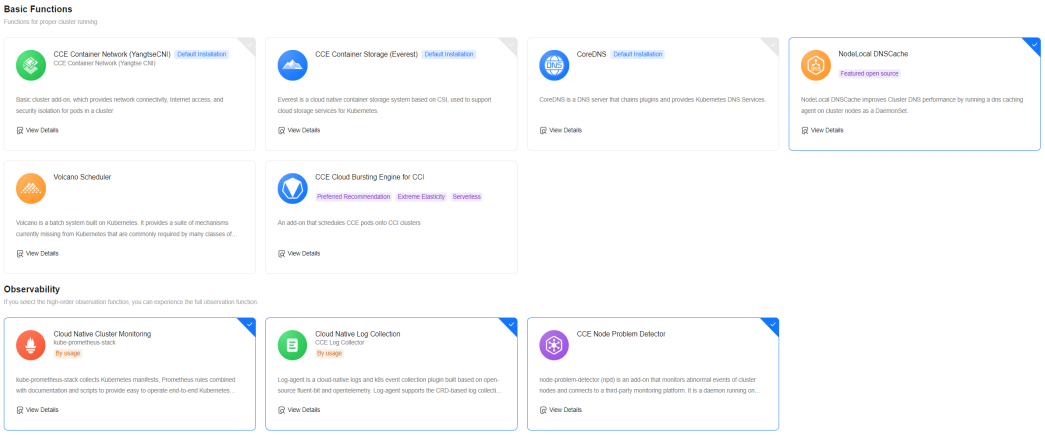
Parameter	Description
IAM Authentication	CCE clusters support IAM authentication. You can call IAM authenticated APIs to access CCE clusters.
Certificate Authentication	• If **Automatically generated** is selected, the X.509-based authentication mode will be enabled by default. X.509 is a commonly used certificate format. • If **Bring your own** is selected, the cluster can identify users based on the header in the request body for authentication. Upload your CA root certificate, client certificate, and private key. **CAUTION** • Upload a file **smaller than 1 MB**. The CA certificate and client certificate can be in **.crt** or **.cer** format. The private key of the client certificate can only be uploaded **unencrypted**. • The validity period of the client certificate must be longer than five years. • The uploaded CA root certificate is used by the authentication proxy and for configuring the kube-apiserver aggregation layer. **If any of the uploaded certificates is invalid, the cluster cannot be created.** • Starting from v1.25, Kubernetes no longer supports certificate authentication generated using the SHA1WithRSA or ECDSAWithSHA1 algorithm. The certificate authentication generated using the SHA256 algorithm is supported instead.

Parameter	Description
CPU Management	If enabled, exclusive CPU cores can be allocated to workload pods. For details, see **CPU Policy**.
Overload Control	After this function is enabled, concurrent requests will be dynamically controlled based on the resource demands received by master nodes to ensure the stable running of the master nodes and the cluster. For details, see **Enabling Overload Control for a Cluster**.
Distributed Cloud (HomeZone/ CloudPond)	If enabled, the cluster can centrally manage data center and edge computing resources. This allows you to deploy containers in proper regions based on service requirements. This function is supported by CCE Turbo clusters only, and you need to register an **edge site** beforehand. For details, see **Using Distributed Cloud Resources in a CCE Turbo Cluster**.
Cluster Deletion Protection	A measure taken to prevent accidental deletion of clusters through the console or APIs. After this function is enabled, you will not be able to delete or unsubscribe from clusters on CCE. You can modify the function status in the cluster Settings after creating it.
Time Zone	The cluster's scheduled tasks and nodes are subject to the chosen time zone.
Resource Tag	You can add resource tags to classify resources. A maximum of 20 resource tags can be added. **NOTE** If your account belongs to an **organization** and the organization has configured with CCE **tag policies**, you need to add tags to the cluster based on these policies. If a tag does not comply with the tag policies, cluster creation may fail. Contact your administrator to learn more about tag policies. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. • A tag key can have a maximum of 128 characters, including letters, digits, spaces, and special characters (-_.:=+@). It cannot start or end with a space, or start with _sys_. The key cannot be empty. • A tag value can have a maximum of 255 characters. It can only contain letters, digits, spaces, and special characters (-_.:/=+@). The value can be empty.
Description	You can enter description for the cluster. A maximum of 200 characters are allowed.

Step 3: Select Add-ons

Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation.

Figure 2-7 Selecting add-ons



Basic capabilities

Add-on Name	Description
CCE Container Network (Yangtse CNI)	This is the basic cluster add-on. It provides network connectivity, Internet access, and security isolation for pods in your cluster.
CCE Container Storage (Everest)	This add-on (**CCE Container Storage (Everest)**) is installed by default. It is a cloud native container storage system based on CSI and supports cloud storage services such as EVS.
CoreDNS	This add-on (**CoreDNS**) is installed by default. It provides DNS resolution for your cluster and can be used to access the in-cloud DNS server.
NodeLocal DNSCache	(Optional) If selected, this add-on (**NodeLocal DNSCache**) will be automatically installed. NodeLocal DNSCache improves cluster DNS performance by running a DNS caching agent on cluster nodes.
Volcano Scheduler	(Optional) After you select this option, CCE will automatically install **Volcano Scheduler** and set the default scheduler of the cluster to Volcano. This will enable you to access advanced scheduling capabilities for batch computing and high-performance computing.
CCE Cloud Bursting Engine for CCI	(Optional) After you select this option, CCE will automatically install **CCE Cloud Bursting Engine for CCI**. When there is a sudden increase in workload, the pods deployed on CCE will be dynamically created on CCI to handle the increased load.

Observability

Add-on Name	Description
Cloud Native Cluster Monitoring	(Optional) If selected, this add-on (**Cloud Native Cluster Monitoring**) will be automatically installed. Cloud Native Cluster Monitoring collects monitoring metrics for your cluster and reports the metrics to AOM. The agent mode does not support HPA based on custom Prometheus statements. If related functions are required, install this add-on manually after the cluster is created. Collecting **basic metrics** is free of charge. Collecting custom metrics is billed by AOM. For details, see **Pricing Details**. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.
Cloud Native Logging	(Optional) If selected, this add-on (**Cloud Native Log Collection**) will be automatically installed. Cloud Native Logging helps report logs to LTS. After the cluster is created, you are allowed to obtain and manage collection rules on the **Logging** page of the CCE cluster console. LTS does not charge you for creating log groups and offers a free quota for you to collect logs every month. You pay only for the log volume exceeding the quota. For details, see **Price Calculator**. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.
CCE Node Problem Detector	(Optional) If selected, this add-on (**CCE Node Problem Detector**) will be automatically installed to detect faults and isolate nodes for prompt cluster troubleshooting.

Step 4: Configure Add-ons

Click **Next: Add-on Configuration**.

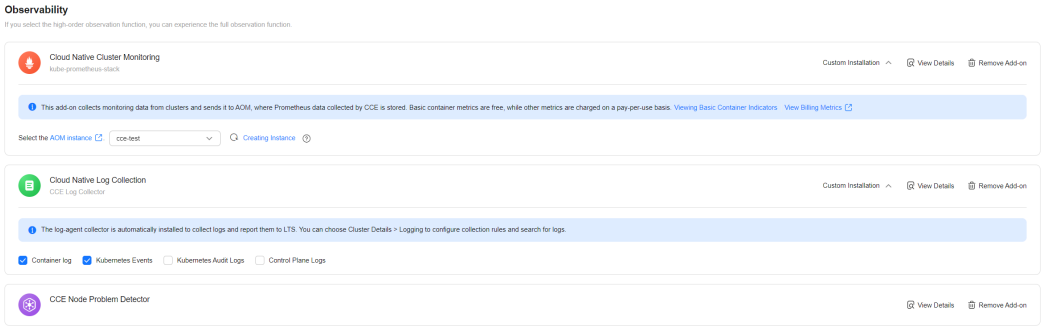
Basic capabilities

Add-on Name	Description
CCE Container Network (Yangtse CNI)	This add-on is unconfigurable.
CCE Container Storage (Everest)	This add-on is unconfigurable. After the cluster is created, choose **Add-ons** in the navigation pane of the cluster console and modify the configuration.
CoreDNS	This add-on is unconfigurable. After the cluster is created, choose **Add-ons** in the navigation pane of the cluster console and modify the configuration.

Add-on Name	Description
NodeLocal DNSCache	This add-on is unconfigurable. After the cluster is created, choose **Add-ons** in the navigation pane of the cluster console and modify the configuration.
Volcano Scheduler	This add-on is unconfigurable. After the cluster is created, choose **Add-ons** in the navigation pane of the cluster console and modify the configuration.
CCE Cloud Bursting Engine for CCI	After you specify the subnet for a CCI instance, the pods that are scheduled to CCI will consume the IP addresses within that subnet. Properly plan the CIDR blocks to prevent running out of available IP addresses. After workload pods are scheduled to CCI, they will be billed according to CCI billing requirements. For details, see **Billed Items**.

Observability

Figure 2-8 Observability settings



Add-on Name	Description
Cloud Native Cluster Monitoring	Select an AOM instance for Cloud Native Cluster Monitoring to report metrics. If no AOM instance is available, click **Creating Instance** to create one. Collecting **basic metrics** is free of charge. Collecting custom metrics is billed by AOM. For details, see **Pricing Details**. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.

Add-on Name	Description
Cloud Native Logging	Select the logs to be collected. If enabled, a log group named **k8s-log-***{clusterId}* will be automatically created, and a log stream will be created for each selected log type. • **Container log**: Standard output logs of containers are collected. The corresponding log stream is named in the format of **stdout-***{Cluster ID}*. • **Kubernetes Events**: Kubernetes logs are collected. The corresponding log stream is named in the format of **event-***{Cluster ID}*. • **Kubernetes audit log**: Audit logs of the master nodes are collected. The corresponding log stream is named in the format of **audit-***{Cluster ID}*. • **Control Plane Logs**: Logs of components like kube-apiserver, kube-controller-manage, and kube-scheduler that run on the master nodes are collected. The corresponding log streams are named in the format of **kube-apiserver-***{Cluster ID}*, **kube-controller-manage-***{Cluster ID}*, and **kube-scheduler-***{Cluster ID}*, respectively. If log collection is disabled, choose **Logging** in the navigation pane of the cluster console after the cluster is created and enable this function. LTS does not charge you for creating log groups and offers a free quota for you to collect logs every month. You pay only for the log volume exceeding the quota. For details, see **Price Calculator**. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.
CCE Node Problem Detector	This add-on is unconfigurable. After the cluster is created, choose **Add-ons** in the navigation pane of the cluster console and modify the configuration.

Step 5: Confirm the Configuration

After the parameters are specified, click **Next: Confirm configuration**. The cluster resource list is displayed. Confirm the information and click **Submit**.

It takes about 5 to 10 minutes to create a cluster. You can click **Back to Cluster List** to perform other operations on the cluster or click **Go to Cluster Events** to view the cluster details.

Related Operations

- After creating a cluster, you can use the Kubernetes command line (CLI) tool kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.
- Add nodes to the cluster. For details, see **Creating a Node**.

- Create an IPv4/IPv6 dual-stack. For details, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**.
- Connect to multiple clusters using kubectl. For details, see **Connecting to Multiple Clusters Using kubectl**.

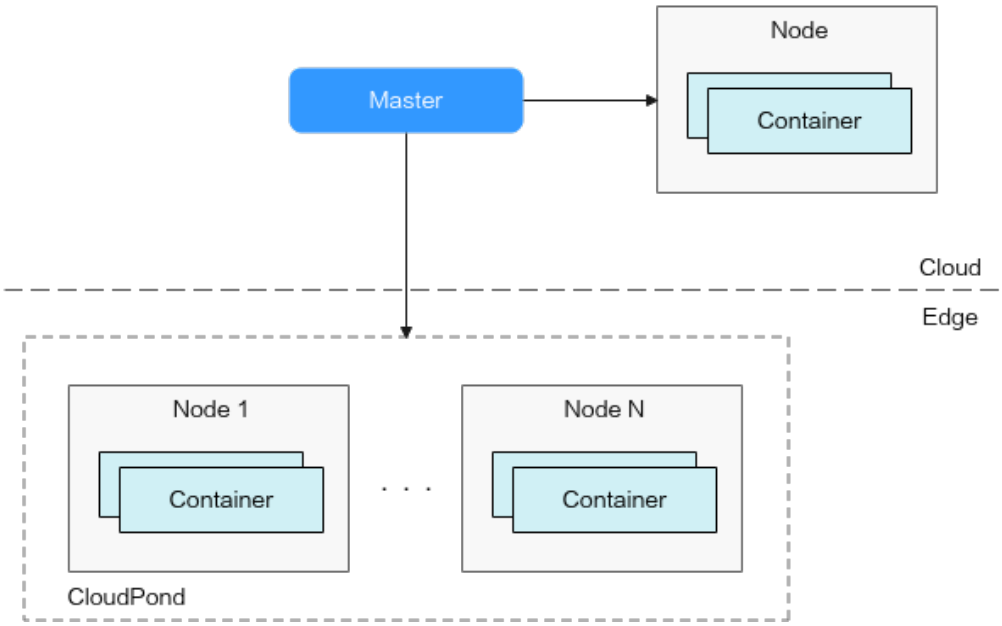
2.3.3 Using Distributed Cloud Resources in a CCE Turbo Cluster

CCE Turbo clusters support CloudPond, allowing you to **manage edge infrastructures**. In a remote cluster, you can centrally manage compute resources in the data center and the edge, and deploy your applications in the locations you want.

NOTE

Before using distributed cloud resources, register and deploy the CloudPond service.

Figure 2-9 Remote CCE Turbo management



Core Concepts

CCE introduces **partitions** to distinguish cloud resources from resources distributed in different edge locations. Partitions are defined as follows:

- From the perspective of computing, a partition is a collection of data center availability zones (AZs) that are physically isolated but close to each other (access latency within 2 ms). Deploying applications in different AZs of a partition ensures high availability.
- From the perspective of networking, nodes and containers in a partition need to use VPC subnets created in AZs of the partition. To facilitate configuration and management, configure the default subnet when creating a partition. If no subnet is specified when you create a node, the node uses the default subnet of the partition.

- Other attributes: Partition types can be center or CloudPond. This classification facilitates application scheduling.

Notes and Constraints

- Nodes: Remote clusters support only x86 VMs. Node migration is unavailable currently.
- Node pools: Random scheduling of a node pool only applies to partitions.
- Storage: You can only create EVS disks in edge partitions.
- Services and ingresses: Only dedicated load balancers are supported.
- Add-ons: The following add-ons are supported by remote clusters and preferentially deployed on cloud servers:
  - **CCE Container Storage (Everest)**
  - **CoreDNS**
  - **NodeLocal DNSCache**
  - **CCE Cluster Autoscaler**
  - **Kubernetes Metrics Server**
  - **CCE Node Problem Detector**
  - **Kubernetes Dashboard**
  - **CCE AI Suite (Ascend NPU)**
  - **CCE AI Suite (NVIDIA GPU)**

Enabling Support for Remote Clouds

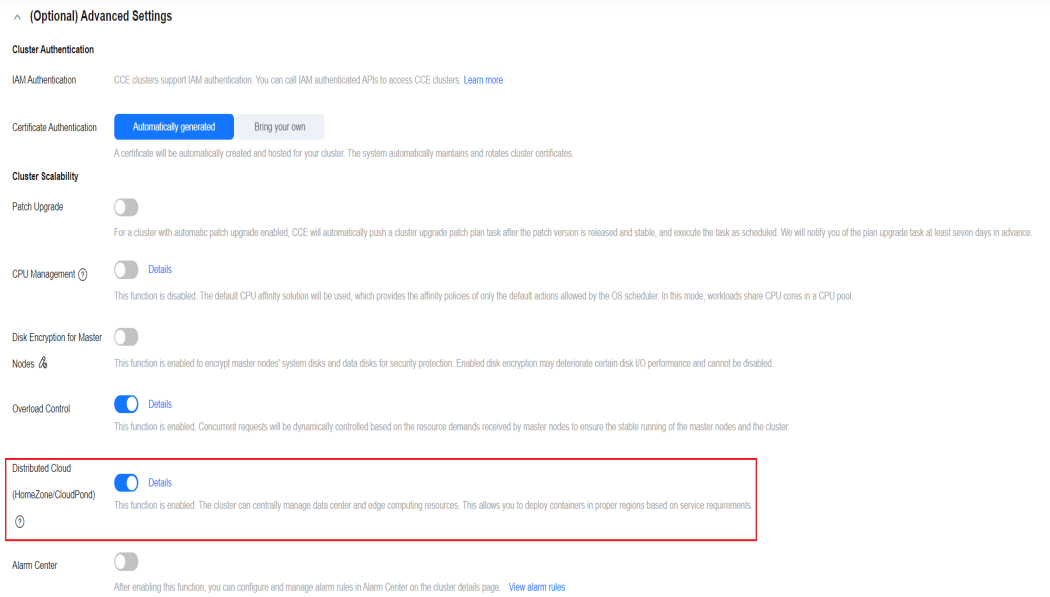
When creating a CCE Turbo cluster, you can enable support for distributed clouds (CloudPond).

NOTE

After distributed clouds are supported, the edge nodes created in the cluster will have the following taint and Kubernetes labels added by default:

- Taint: **distribution.io/category=IES:NoSchedule**
- Kubernetes labels:
  - **distribution.io/category=IES**
  - **distribution.io/partition=<AZ name>**
  - **distribution.io/publicbordergroup=<AZ name>**

Figure 2-10 Enabling distributed clouds (CloudPond)



2.3.4 Comparing iptables and IPVS

kube-proxy is a key component of a Kubernetes cluster. It is used for load balancing and forwarding data between a Service and its backend pods.

CCE supports the iptables and IP Virtual Server (IPVS) forwarding modes.

Feature Difference	iptables	IPVS
Positioning	A mature, stable kube-proxy mode, but its performance is average. It applies to scenarios where the number of Services is small (less than 3000) or there are a large number of short concurrent connections on the client. For details, see **iptables**.	A high-performance kube-proxy mode. This mode is suitable for large cluster scales or when there are a large number of Services. For details, see **IPVS**.
Throughput	Relatively low	Relatively high
Complexity	O(n). *n* increases with the number of Services and backend pods in the cluster.	O(1). In most cases, the connection processing efficiency is irrelevant to the cluster scale.

Feature Difference	iptables	IPVS
Load balancing algorithm	iptables has only one algorithm for random selection.	IPVS involves multiple load balancing algorithms, such as round-robin, shortest expected delay, least connections, and various hashing methods.
ClusterIP connectivity	The internal IP address in the cluster cannot be pinged.	The internal IP address in the cluster can be pinged. **NOTE** The IP address in clusters of v1.27 or later cannot be pinged due to **security hardening**.
Additional restrictions	When there are more than 3000 Services in a cluster, network delay may occur.	• If you use the same load balancer for both the ingress and Service in the same cluster or for the Service ports in different clusters, you will not be able to access the ingress from the nodes or containers in the cluster, or the Service ports in other clusters. This is because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge, which intercepts the load balancer traffic. Use separate load balancers for these ingresses and Services. • EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04 are not recommended. For details, see **What Should I Do If There Is a Service Access Failure After a Backend Service Upgrade or a 1-Second Latency When a Service Accesses the CCE Cluster?**

iptables

iptables is a Linux kernel function for processing and filtering a large number of data packets. It allows flexible sequences of rules to be attached to various hooks in the packet processing pipeline. When iptables is used, kube-proxy implements NAT and load balancing in the NAT pre-routing hook. For each Service, kube-proxy installs an iptables rule which captures the traffic destined for the Service's ClusterIP and ports and redirects the traffic to one of the backend pods. By default, iptables randomly selects a backend pod. For details, see **iptables proxy mode**.

IPVS

IPVS is constructed on top of Netfilter and balances transport-layer loads as part of the Linux kernel. IPVS can direct requests for TCP- or UDP-based services to the

real servers, and make services of the real servers appear as virtual services on a single IP address.

In the IPVS mode, kube-proxy uses IPVS load balancing instead of iptables. IPVS is designed to balance loads for a large number of Services. It has a set of optimized APIs and uses optimized search algorithms instead of simply searching for rules from a list. For details, see **IPVS proxy mode**.

2.4 Connecting to a Cluster

2.4.1 Connecting to a Cluster Using kubectl

Scenario

This section uses a CCE standard cluster as an example to describe how to access a CCE cluster using kubectl.

Permissions

When you access a cluster using kubectl, CCE uses **kubeconfig** generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a **kubeconfig** file vary from user to user.

For details about user permissions, see **Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)**.

Using kubectl

To connect to a Kubernetes cluster from a PC, you can use kubectl, a Kubernetes command line tool. You can log in to the CCE console and click the name of the target cluster to access the cluster console. On the **Overview** page, view the access address and kubectl connection procedure.

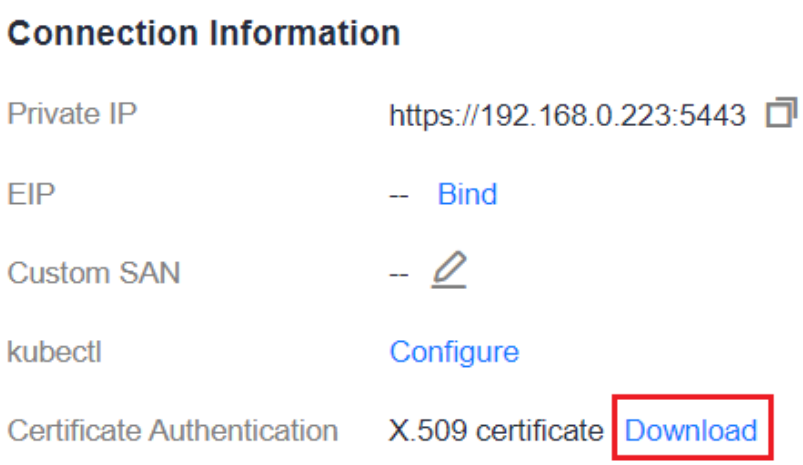
CCE allows you to access a cluster through a private network or a public network.

- Intranet access: The client that accesses the cluster must be in the same VPC as the cluster.
- Public access: The client that accesses the cluster must be able to access public networks and the cluster has been bound with a public network IP.

NOTICE

To bind an EIP to the cluster, go to the **Overview** page and click **Bind** next to **EIP** in the **Connection Information** area, as shown in **Figure 2-11**. In a cluster with an EIP bound, kube-apiserver will be exposed to the Internet and may be attacked. To solve this problem, you can configure Advanced Anti-DDoS for the EIP of the node on which kube-apiserver runs.

Figure 2-11 Cluster connection information



Download kubectl and the configuration file. Copy the file to your client, and configure kubectl. After the configuration is complete, you can access your Kubernetes clusters. The process is as follows:

Step 1 Download kubectl.

Prepare a computer that can access the public network and install kubectl in CLI mode. You can run the **kubectl version** command to check whether kubectl has been installed. If kubectl has been installed, skip this step.

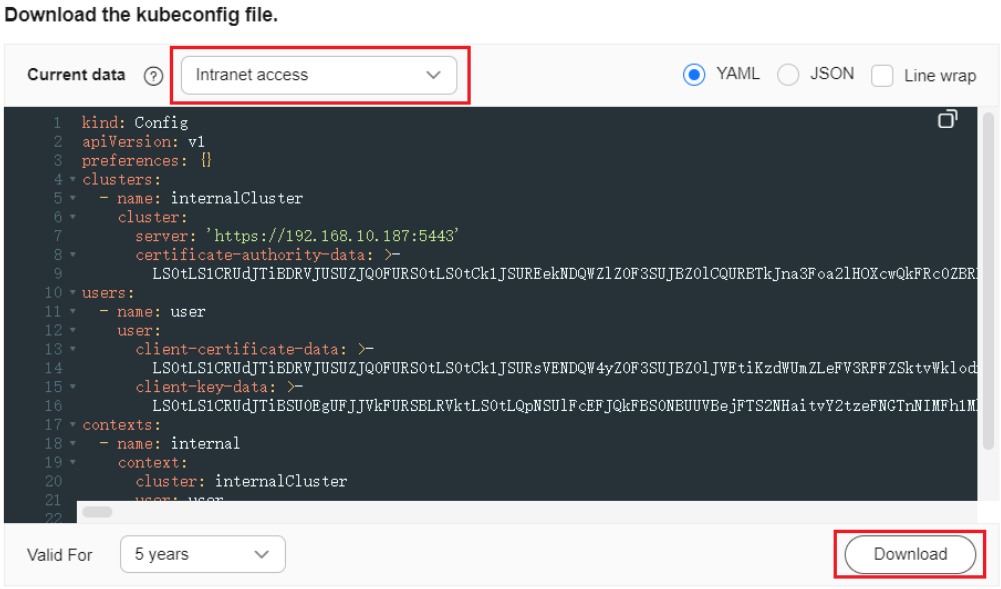
This section uses the Linux environment as an example to describe how to install and configure kubectl. For details, see **Installing kubectl**.

1. Log in to your client and download kubectl.
   ```
   cd /home
   curl -LO https://dl.k8s.io/release/{v1.25.0}/bin/linux/amd64/kubectl
   ```
   {v1.25.0} specifies the version. Replace it as required.
2. Install kubectl.
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

Step 2 Obtain the kubectl configuration file.

In the **Connection Info** pane on the **Overview** page, click **Configure** next to **kubectl** to check the kubectl connection. On the displayed page, choose **Intranet access** or **Public network access** and download the configuration file.

Figure 2-12 Downloading the configuration file



NOTE

- The kubectl configuration file **kubeconfig** is used for cluster authentication. If the file is leaked, your clusters may be attacked.
- The Kubernetes permissions assigned by the configuration file downloaded by IAM users are the same as those assigned to the IAM users on the CCE console.
- If the KUBECONFIG environment variable is configured in the Linux OS, kubectl preferentially loads the KUBECONFIG environment variable instead of **$home/.kube/config**.

Step 3 Configure kubectl.

Configure kubectl (A Linux OS is used).

1. Log in to your client and copy the configuration file (for example, **kubeconfig.yaml**) downloaded in **Step 2** to the **/home** directory on your client.
2. Configure the kubectl authentication file.
   ```
   cd /home
   mkdir -p $HOME/.kube
   mv -f kubeconfig.yaml $HOME/.kube/config
   ```
3. Switch the kubectl access mode based on service scenarios.
   - Run this command to enable intra-VPC access:
     ```
     kubectl config use-context internal
     ```
   - Run this command to enable public access (EIP required):
     ```
     kubectl config use-context external
     ```
   - Run this command to enable public access and two-way authentication (EIP required):
     ```
     kubectl config use-context externalTLSVerify
     ```
     For details about the cluster two-way authentication, see **Two-Way Authentication for Domain Names**.

----End

Two-Way Authentication for Domain Names

CCE supports two-way authentication for domain names.

- After an EIP is bound to an API Server, two-way domain name authentication is disabled by default if kubectl is used to access the cluster. You can run **kubectl config use-context externalTLSVerify** to enable the two-way domain name authentication.
- When an EIP is bound to or unbound from a cluster, or a custom domain name is configured or updated, the cluster server certificate will be added the latest cluster access address (including the EIP bound to the cluster and all custom domain names configured for the cluster).
- Asynchronous cluster synchronization takes about 5 to 10 minutes. You can view the synchronization result in **Synchronize Certificate** in **Operation Records**.
- For a cluster that has been bound to an EIP, if the authentication fails (x509: certificate is valid) when two-way authentication is used, bind the EIP again and download **kubeconfig.yaml** again.
- If the two-way domain name authentication is not supported, **kubeconfig.yaml** contains the **"insecure-skip-tls-verify": true** field, as shown in **Figure 2-13**. To use two-way authentication, download the **kubeconfig.yaml** file again and enable two-way authentication for the domain names.

Figure 2-13 Two-way authentication disabled for domain names

```
"clusters": [{
    "name": "mycluster",
    "cluster": {
        "server": "https://10.100.0.52:5443",
        "insecure-skip-tls-verify": true
    }
```

FAQs

- **Error from server Forbidden**

  When you use kubectl to create or query Kubernetes resources, the following output is returned:

  ```
  # kubectl get deploy Error from server (Forbidden): deployments.apps is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "deployments" in API group "apps" in the namespace "default"
  ```

  The cause is that the user does not have the permissions to operate the Kubernetes resources. For details about how to assign permissions, see **Namespace Permissions (Kubernetes RBAC-based)**.

- **The connection to the server localhost:8080 was refused**

  When you use kubectl to create or query Kubernetes resources, the following output is returned:

  ```
  The connection to the server localhost:8080 was refused - did you specify the right host or port?
  ```

  The cause is that cluster authentication is not configured for the kubectl client. For details, see **Step 3**.

Related Operations

- Connect to multiple clusters using kubectl.
- Configure kubeconfig for fine-grained management on cluster resources

2.4.2 Connecting to a Cluster Using CloudShell

Scenario

This section uses a CCE standard cluster as an example to describe how to access a CCE cluster using CloudShell.

Permissions

When using kubectl in CloudShell, the kubectl permissions are determined by the user that logs in.

Notes and Constraints

When using CloudShell to access a CCE cluster or container, you can open a maximum of 15 instances simultaneously.

Using CloudShell

CloudShell is a web shell used to manage and maintain cloud resources. CCE allows you to use CloudShell to access clusters and use kubectl in CloudShell to access clusters (by clicking the command line tool icon in Figure 2-14).

NOTE

- The kubectl certificate in CloudShell is valid for one day. You can reset the validity period by accessing CloudShell from the CCE console.
- CloudShell is implemented based on VPCEP. To use kubectl to access a cluster, configure the security group (*Cluster name*-cce-control-*Random number*) on the master node of the cluster to allow the following CIDR blocks to access port 5443. By default, port 5443 allows access from all CIDR blocks. If you have hardened security groups and any cluster cannot be accessed in CloudShell, check whether port 5443 allows access from 198.19.0.0/16.
- CloudShell can be used only after CoreDNS is installed in a cluster.
- Using CloudShell to access containers is available only in the following regions: CN North-Beijing1, CN North-Beijing4, CN North-Ulanqab1, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and AP-Singapore.
- CloudShell does not support an account or sub-project agency.

Figure 2-14 CloudShell

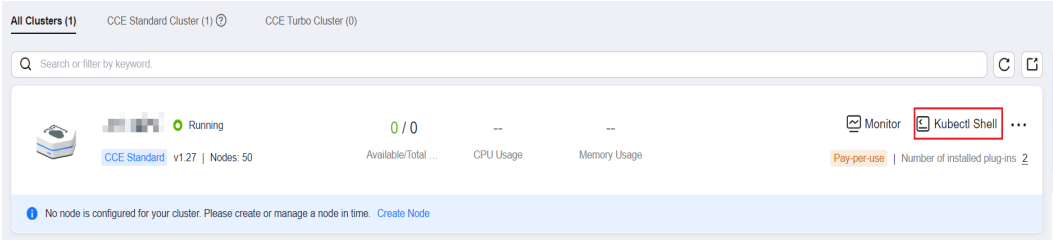
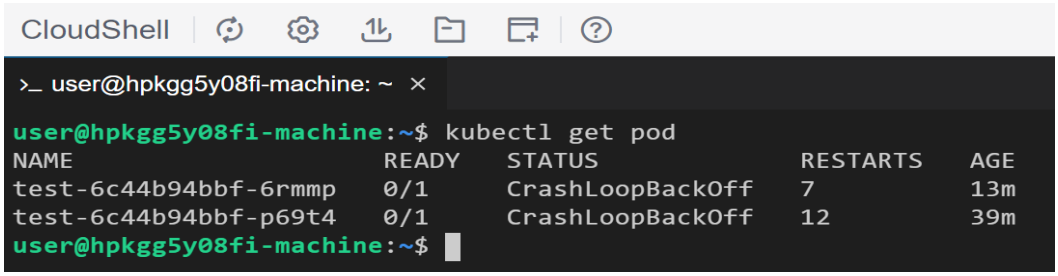


Figure 2-15 Using kubectl in CloudShell



2.4.3 Accessing a Cluster Using an X.509 Certificate

Scenario

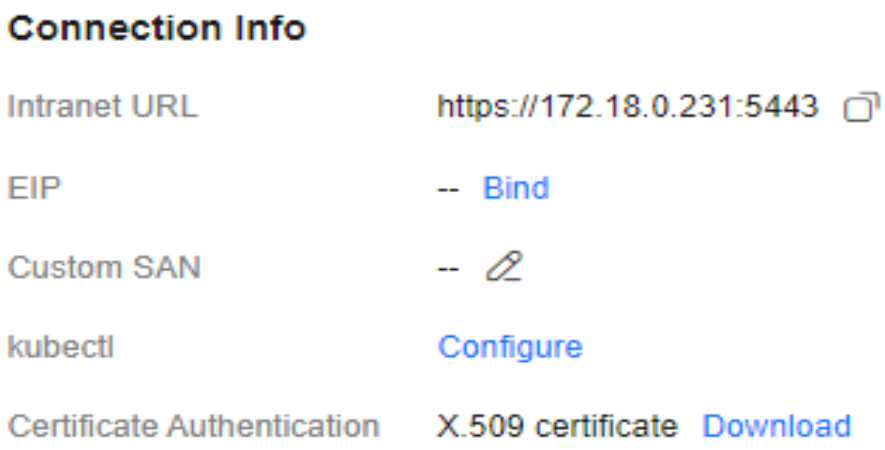
This section describes how to obtain the cluster certificate from the console and use it to access Kubernetes clusters.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 On the **Overview** page, locate the **Connection Info** area, and click **Download** next to **X.509 certificate**.

Figure 2-16 Downloading a cluster certificate



Step 3 In the **Obtain Certificate** dialog box displayed, select the certificate expiration time and download the X.509 certificate of the cluster as prompted.

NOTICE

- The downloaded certificate contains three files: **client.key**, **client.crt**, and **ca.crt**. Keep these files secure.
- Certificates are not required for mutual access between containers in a cluster.

Step 4 Call native Kubernetes APIs using the cluster certificate.

For example, run the **curl** command to call an API to view the pod information. The following is an example:

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key  https://192.168.0.18:5443/api/v1/namespaces/default/pods/
```

NOTE

- *./ca.crt*, *./client.crt*, and *./client.key* are the paths for uploading the **client.key**, **client.crt**, and **ca.crt** files, respectively.
- *192.168.0.18:5443* is the private or public network address of the API server in the cluster.

For more cluster APIs, see **Kubernetes API**.

----End

2.4.4 Accessing a Cluster Using a Custom Domain Name

Scenario

Subject Alternative Name (SAN) allows multiple values (including IP addresses, domain names, and so on) to be associated with certificates. A SAN is usually used by the client to verify the server validity in TLS handshakes. Specifically, the validity check includes whether the server certificate is issued by a CA trusted by the client and whether the SAN in the certificate matches the IP address or DNS domain name that the client actually accesses.

If the client cannot directly access the private IP or EIP of the cluster, you can sign the IP address or DNS domain name that can be directly accessed by the client into the cluster server certificate as a SAN to enable two-way authentication on the client, which improves security. Typical use cases include DNAT access and domain name access.

If you have particular proxy access requirements or need to access resources in other regions, you can customize a SAN. Typical domain name access scenarios:

- Add the domain name mapping by either adding the DNS domain name address in the host domain name configuration on the client or configuring **/etc/hosts** on the client host.
- Use domain name access in the intranet. DNS allows you to configure mappings between cluster EIPs and custom domain names. After an EIP is updated, you can continue to use two-way authentication and the domain name to access the cluster without downloading the **kubeconfig.json** file again.
- Add A records on a self-built DNS server.

Prerequisites

A cluster of v1.19 or later is available.

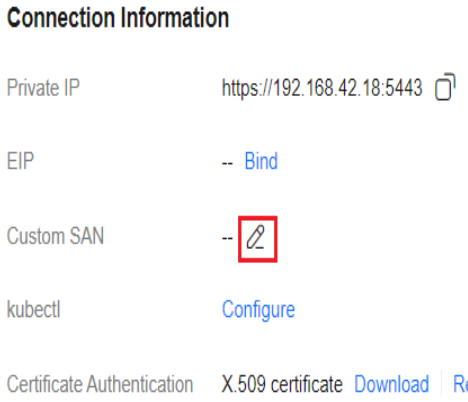
Customizing a SAN

Step 1 Log in to the CCE console.

Step 2 Click the name of the target cluster in the cluster list to go to the cluster **Overview** page.

Step 3 In the **Connection Information** area, click next to **Custom SAN**. In the dialog box displayed, enter the IP address or domain name and click **Save**.

Figure 2-17 Custom SAN



NOTE

- The kube-apiserver will be restarted and the kubeconfig certificate will be updated, which will take approximately 5 minutes. Do not perform any operations on the cluster during this period. After the operation is complete, download the updated kubeconfig certificate.
- A maximum of 128 domain names or IP addresses, separated by commas (,), are allowed.
- If a custom domain name needs to be bound to an EIP, ensure that you have configured an EIP.

----End

Connecting to a Cluster Using the SAN

Using kubectl to access the cluster

Step 1 Download the **kubeconfig.json** file again after the SAN is modified.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. On the **Overview** page, locate the **Connection Info** area, click **Configure** next to **kubectl**. On the page displayed, download the configuration file.

Step 2 Configure kubectl.

1. Log in to your client and copy the **kubeconfig.json** file downloaded in **Step 1.2** to the **/home** directory on your client.

2. Configure the kubectl authentication file.

```
cd /home
mkdir -p $HOME/.kube
mv -f kubeconfig.json $HOME/.kube/config
```

3. Change the kubectl access mode and use the SAN to access the cluster.

```
kubectl config use-context customSAN-0
```

In the preceding command, *customSAN-0* indicates the configuration name of the custom SAN. If multiple SANs are configured, the number in the configuration name of each SAN starts from **0** and increases in ascending order, for example, *customSAN-0*, *customSAN-1*, and so on.

----End

Using an X.509 certificate to access the cluster

Step 1 After the SAN is modified, download the X.509 certificate again.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. On the **Overview** page, locate the **Connection Info** area, and click **Download** next to **X.509 certificate**.
3. In the **Obtain Certificate** dialog box displayed, select the certificate expiration time and download the X.509 certificate of the cluster as prompted.

Step 2 Call native Kubernetes APIs using the cluster certificate.

For example, run the **curl** command to call an API to obtain the pod information. In the following information, *example.com:5443* indicates the custom SAN.

```
curl --cacert ./ca.crt --cert ./client.crt --key ./client.key  https://example.com:5443/api/v1/namespaces/default/pods/
```

For more cluster APIs, see **Kubernetes API**.

----End

2.4.5 Configuring a Cluster's API Server for Internet Access

You can bind an EIP to an API server of a Kubernetes cluster so that the API server can access the Internet.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 On the **Overview** page, locate the **Connection Info** area, and click **Bind** next to **EIP**.

Step 3 Select an existing EIP. If no EIP is available, click **Create EIP** to go to the EIP console and assign one.

NOTE

- Binding an EIP to an API server for Internet access can pose a risk to the cluster's security. To mitigate this risk, configure Advanced Anti-DDoS or API server access policies (**Configuring Access Policies for an API Server**) for the bound EIP.
- Binding an EIP to an API server will cause the API server to restart briefly and update the kubeconfig certificate. Do not make any changes to the cluster during this period.

Step 4 Click **OK**.

----End

Configuring Access Policies for an API Server

To ensure the security of a cluster's API server, it is important to modify the security group rules for the master nodes. This is because the EIP, which is exposed to the Internet, is at risk of being attacked.

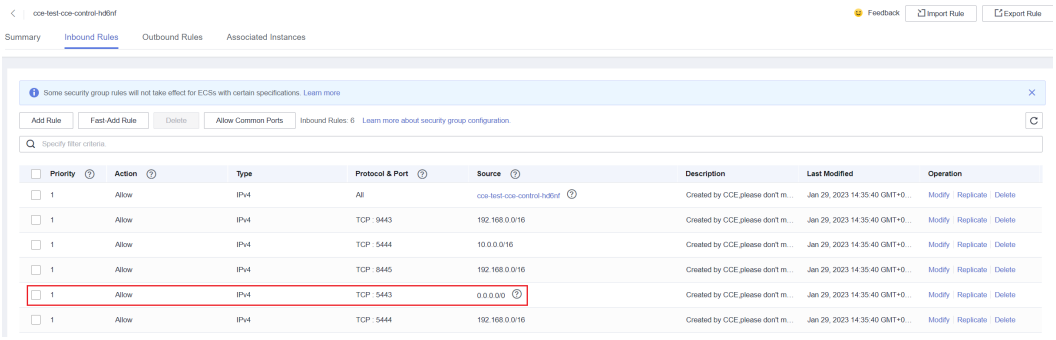
Step 1 Log in to the CCE console and click the cluster name to access the cluster console. On the **Overview** page, find and copy the cluster ID.

Step 2 Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups**.

Step 3 Select **Description** as the filter criterion and search for the target security group by cluster ID.

Step 4 Locate the row that contains the security group (starting with *{CCE cluster name}*-**cce-control**) of the master node and click **Manage Rules** in the **Operation** column.

Step 5 On the page displayed, locate the row that contains port 5443 and click **Modify** in the **Operation** column to modify its inbound rules.

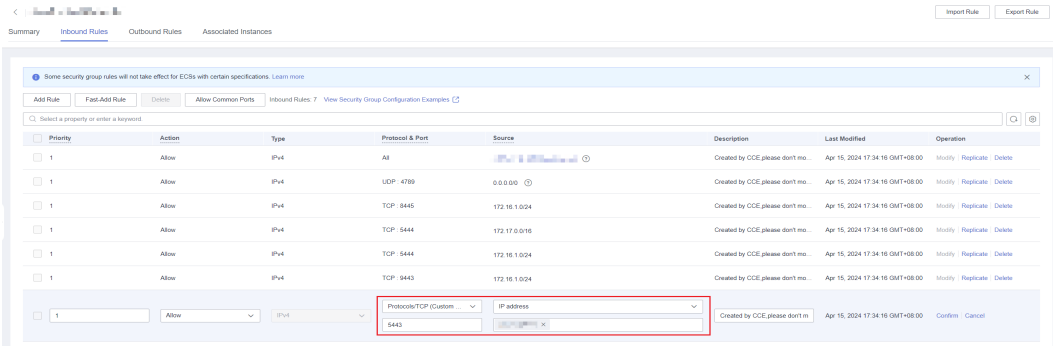


Step 6 Change the source IP address that can be accessed as required. For example, if the IP address used by the client to access the API Server is **100.*.*.***, you can add an inbound rule for port 5443 and set the source IP address to **100.*.*.***.

NOTE

In addition to the client IP address, the port must allow traffic from the CIDR blocks of the VPC, container, and the control plane of the hosted service mesh to ensure that the API Server can be accessed from within the cluster.

To use CloudShell, you need to allow traffic from 198.19.0.0/16 on port 5443. Otherwise, you cannot access the cluster using CloudShell.



Step 7 Click **Confirm**.

----End

2.4.6 Revoking a Cluster Access Credential

In multi-tenant scenarios, CCE generates a credential (kubeconfig or X.509 certificate) for you to access the corresponding cluster. The credential contains user identity and authorization information so that you can access to the cluster and perform authorized operations on it. Credentials ensure security between IAM users for user management and authorization. However, the validity period of a credential is usually fixed. When an employee holding the credential resigns or an authorized credential is disclosed, you need to manually revoke the credential to ensure cluster security.

The credentials can be revoked in the following scenarios:

- IAM users (non-administrators) intend to revoke their own credentials.
- Accounts or administrators (users in the admin user group) can revoke the credentials of other users or delegated accounts.

NOTICE

After a credential is revoked, the holder of the credential cannot access the cluster, and the revoked credential cannot be restored. Exercise caution when performing this operation.

To access the same cluster, you need to regenerate a credential.

Prerequisites

A cluster of v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, or later is available.

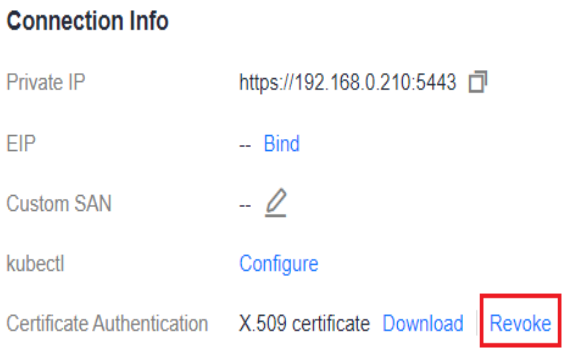
Operations Performed by IAM Users

IAM users can only revoke their own credentials. To revoke a credential, perform the following operations:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Overview** in the navigation pane. In the **Connection Info** area on the right of the page, click **Revoke**.

Figure 2-18 Revoking a credential



Step 3 In the dialog box displayed, enter **REVOKE** in the confirmation box and click **OK** if you are sure you want to revoke the credential.

----End

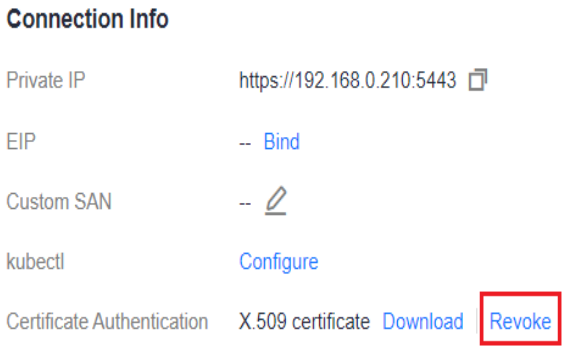
Operations Performed by Accounts or Administrators

Accounts or administrators (users in the admin user group) can revoke the credentials of other users or delegated accounts. To revoke a credential, see the following operations:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Overview** in the navigation pane. In the **Connection Info** area on the right of the page, click **Revoke**.

Figure 2-19 Revoking a credential



Step 3 Select a certificate applicant and revoke the certificate. The revoked certificate cannot be restored. Exercise caution when performing this operation.

- **User**: Select an IAM user and revoke its credential.
- **Account**: Select an agency account and revoke its credentials.
- **Specify User ID/Delegacy ID**: If the user has been deleted or you log in to the system through an identity provider, manually enter the user ID. If the delegated account has been deleted, manually enter the delegated ID.

  You can use the following solution to obtain the ID:

  – If you can obtain the certificate downloaded by the applicant, the name (**CN - Common Name**) of the certificate is the required ID.

- If you cannot obtain the certificate downloaded by the applicant, use Cloud Trace Service (CTS) to obtain the events of deleting a user (**deleteUser**) and deleting an agency (**deleteAgency**). The resource IDs of the events are the IDs of the deleted user and delegated account, respectively.

If the required ID cannot be obtained using the preceding methods, submit a service ticket to O&M personnel.

Step 4 In the dialog box displayed, enter **REVOKE** in the confirmation box and click **OK** if you are sure you want to revoke the credential.

----End

2.5 Managing a Cluster

2.5.1 Modifying Cluster Configurations

Scenario

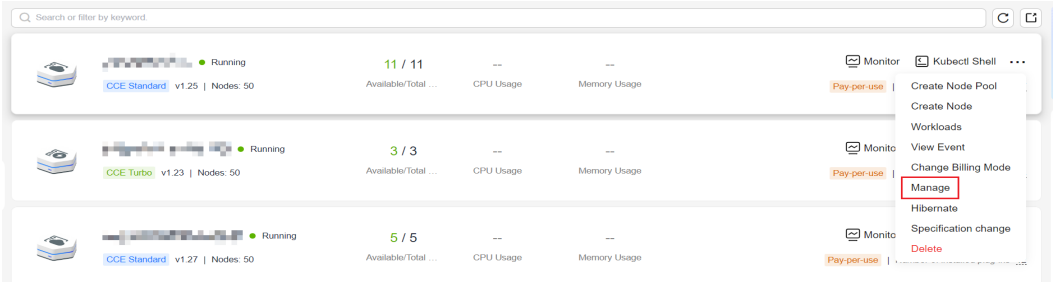
CCE allows you to manage cluster parameters, through which you can let core components work under your requirements.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the target cluster, click **...** to view more operations on the cluster, and choose **Manage**.

Figure 2-20 Configuration management



Step 3 On the **Manage Component** page, change the values of the Kubernetes parameters listed in the following table.

Table 2-17 kube-apiserver configurations

Item	Parameter	Description	Value
Toleration time for nodes in NotReady state	default-not-ready-toleration-seconds	Specifies the default tolerance time. The configuration takes effect for all pods by default. You can configure different tolerance time for pods. In this case, the tolerance time configured for the pod is used. For details, see **Configuring Tolerance Policies**. If the specified tolerance time is too short, pods may be frequently migrated in scenarios like a network jitter. If the specified tolerance time is too long, services may be interrupted during this period after the node is faulty.	Default: 300s
Toleration time for nodes in unreachable state	default-unreachable-toleration-seconds	Specifies the default tolerance time. The configuration takes effect for all pods by default. You can configure different tolerance time for pods. In this case, the tolerance time configured for the pod is used. For details, see **Configuring Tolerance Policies**. If the specified tolerance time is too short, pods may be frequently migrated in scenarios like a network jitter. If the specified tolerance time is too long, services may be interrupted during this period after the node is faulty.	Default: 300s

Item	Parameter	Description	Value
Maximum Number of Concurrent Modification API Calls	max-mutating-requests-inflight	Maximum number of concurrent mutating requests. When the value of this parameter is exceeded, the server rejects requests. The value **0** indicates that there is no limitation on the maximum number of concurrent modification requests. This parameter is related to the cluster scale. You are advised not to change the value.	Manual configuration is no longer supported since cluster v1.21. The value is automatically specified based on the cluster scale. • **200** for clusters with 50 or 200 nodes • **500** for clusters with 1000 nodes • **1000** for clusters with 2000 nodes
Maximum Number of Concurrent Non-Modification API Calls	max-requests-inflight	Maximum number of concurrent non-mutating requests. When the value of this parameter is exceeded, the server rejects requests. The value **0** indicates that there is no limitation on the maximum number of concurrent non-modification requests. This parameter is related to the cluster scale. You are advised not to change the value.	Manual configuration is no longer supported since cluster v1.21. The value is automatically specified based on the cluster scale. • **400** for clusters with 50 or 200 nodes • **1000** for clusters with 1000 nodes • **2000** for clusters with 2000 nodes

Item	Parameter	Description	Value
NodePort port range	service-node-port-range	NodePort port range. After changing the value, go to the security group page and change the TCP/UDP port range of node security groups 30000 to 32767. Otherwise, ports other than the default port cannot be accessed externally. If the port number is smaller than **20106**, a conflict may occur between the port and the CCE health check port, which may further lead to unavailable cluster. If the port number is greater than **32767**, a conflict may occur between the port and the ports in **net.ipv4.ip_local_port_range**, which may further affect the network performance.	Default: 30000 to 32767 Value range: Min > 20105 Max < 32768
Overload Control	support-overload	Cluster overload control. After this function is enabled, concurrent requests will be dynamically controlled based on the resource demands received by master nodes to ensure the stable running of the master nodes and the cluster. This parameter is available only in clusters of v1.23 or later.	• false: Overload control is disabled. • true: Overload control is enabled.
Node Restriction Add-on	enable-admission-plugin-node-restriction	This add-on allows the kubelet of a node to operate only the objects of the current node for enhanced isolation in multi-tenant scenarios or the scenarios with high security requirements. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.	Default: true

Item	Parameter	Description	Value
Pod Node Selector Add-on	enable-admission-plugin-pod-node-selector	This add-on allows cluster administrators to configure the default node selector through namespace annotations. In this way, pods run only on specific nodes and configurations are simplified. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.	Default: true
Pod Toleration Limit Add-on	enable-admission-plugin-pod-toleration-restriction	This add-on allows cluster administrators to configure the default value and limits of pod tolerations through namespaces for fine-grained control over pod scheduling and key resource protection. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.	Default: false

Item	Parameter	Description	Value
API Audience Settings	api-audiences	Audiences for a service account token. The Kubernetes component for authenticating service account tokens checks whether the token used in an API request specifies authorized audiences. Configuration suggestion: Accurately configure audiences according to the communication needs among cluster services. By doing so, the service account token is used for authentication only between authorized services, which enhances security. **NOTE** An incorrect configuration may lead to an authentication communication failure between services or an error during token verification. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default value: "https://kubernetes.default.svc.cluster.local" Multiple values can be configured, which are separated by commas (,).
Service Account Token Issuer Identity	service-account-issuer	Entity identifier for issuing a service account token, which is the value identified by the **iss** field in the payload of the service account token. Configuration suggestion: Ensure the configured issuer URL can be accessed in the cluster and trusted by the authentication system in the cluster. **NOTE** If your specified issuer URL is untrusted or inaccessible, the authentication process based on the service account may fail. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default value: "https://kubernetes.default.svc.cluster.local" Multiple values can be configured, which are separated by commas (,).

Table 2-18 Scheduler configurations

Item	Parameter	Description	Value
Default cluster scheduler	default-scheduler	• kube-scheduler scheduler: provides the standard scheduling capability of the community. • volcano scheduler: compatible with kube-scheduler scheduling capabilities and provides enhanced scheduling capabilities. For details, see **Volcano Scheduling**.	Default: kube-scheduler
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communicating with kube-apiserver.	• If the number of nodes in a cluster is less than 1000, the default value is **100**. • If the number of nodes in a cluster is 1000 or more, the default value is **200**.
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communicating with kube-apiserver.	• If the number of nodes in a cluster is less than 1000, the default value is **100**. • If the number of nodes in a cluster is 1000 or more, the default value is **200**.

Item	Parameter	Description	Value
Whether to enable GPU sharing	enable-gpu-share	Whether to enable GPU sharing. This parameter is supported only in clusters of v1.23.7-r10, v1.25.3-r0, or later versions. • When disabled, ensure that pods in the cluster cannot use shared GPUs (no **cce.io/gpu-decision** annotation in pods) and that GPU virtualization is disabled. • When enabled, ensure that there is a **cce.io/gpu-decision** annotation on all pods that use GPU resources in the cluster.	Default: true

Table 2-19 kube-controller-manager configurations

Item	Parameter	Description	Value
Number of concurrent processing of deployment	concurrent-deployment-syncs	Number of deployment objects that can be synchronized concurrently	Default: 5
Concurrent processing number of endpoint	concurrent-endpoint-syncs	Number of endpoint syncing operations that will be done concurrently	Default: 5
Concurrent number of garbage collector	concurrent-gc-syncs	Number of garbage collector workers that can be synchronized concurrently	Default: 20
Number of job objects allowed to sync simultaneously	concurrent-job-syncs	Number of job objects that can be synchronized concurrently	Default: 5

Item	Parameter	Description	Value
Number of CronJob objects allowed to sync simultaneously	concurrent-cron-job-syncs	Number of scheduled jobs that can be synchronized concurrently	Default: 5
Number of concurrent processing of namespace	concurrent-namespace-syncs	Number of namespace objects that can be synchronized concurrently	Default: 10
Concurrent processing number of replicaset	concurrent-replicaset-syncs	Number of replica sets that can be synchronized concurrently	Default: 5
Number of concurrent processing of resource quota	concurrent-resource-quota-syncs	Number of resource quotas that can be synchronized concurrently	Default: 5
Concurrent processing number of service	concurrent-service-syncs	Number of services that can be synchronized concurrently	Default: 10
Concurrent processing number of serviceaccount-token	concurrent-serviceaccount-token-syncs	Number of service account token objects that can be synchronized concurrently	Default: 5
Concurrent processing of ttl-after-finished	concurrent-ttl-after-finished-syncs	Number of **ttl-after-finished-controller** workers that can be synchronized concurrently	Default: 5
RC	concurrent_rc_syncs (used in clusters of v1.19 or earlier) concurrent-rc-syncs (used in clusters of v1.21 through v1.25.3-r0)	Number of replication controllers that can be synchronized concurrently **NOTE** This parameter is no longer supported in clusters of v1.25.3-r0 and later versions.	Default: 5

Item	Parameter	Description	Value
Cluster elastic computing period	horizontal-pod-autoscaler-sync-period	Period for the horizontal pod autoscaler to perform auto scaling on pods. A smaller value will result in a faster auto scaling response and higher CPU load. **NOTE** Make sure to configure this parameter properly as a lengthy period can cause the controller to respond slowly, while a short period may overload the cluster control plane.	Default: 15 seconds
Horizontal Pod Scaling Tolerance	horizontal-pod-autoscaler-tolerance	The configuration determines how quickly HPA will act to auto scaling policies. If the parameter is set to **0**, auto scaling will be triggered immediately when the related metrics are met. Configuration suggestion: If the service resource usage increases sharply over time, retain a certain tolerance to prevent auto scaling which is beyond expectation in high resource usage scenarios.	Default: 0.1

Item	Parameter	Description	Value
HPA CPU Initialization Period	horizontal-pod-autoscaler-cpu-initialization-period	During the period specified by this parameter, the CPU usage data used in HPA calculation is limited to pods that are both ready and have recently had their metrics collected. You can use this parameter to filter out unstable CPU usage data during the early stage of pod startup. This helps prevent incorrect scaling decisions based on momentary peak values. Configuration suggestion: If you find that HPA is making incorrect scaling decisions due to CPU usage fluctuations during pod startup, increase the value of this parameter to allow for a buffer period of stable CPU usage. **NOTE** Make sure to configure this parameter properly as a small value may trigger unnecessary scaling based on peak CPU usage, while a large value may cause scaling to be delayed. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default: 5 minutes

Item	Parameter	Description	Value
HPA Initial Readiness Delay	horizontal-pod-autoscaler-initial-readiness-delay	After CPU initialization, this period allows HPA to use a less strict criterion for filtering CPU metrics. During this period, HPA will gather data on the CPU usage of the pod for scaling, regardless of any changes in the pod's readiness status. This parameter ensures continuous tracking of CPU usage, even when the pod status changes frequently. Configuration suggestion: If the readiness status of pods fluctuates after startup and you want to prevent HPA misjudgment caused by the fluctuation, increase the value of this parameter to allow HPA to gather more comprehensive CPU usage data. **NOTE** Configure this parameter properly. If it is set to a small value, an unnecessary scale-out may occur due to CPU data fluctuations when the pod enters the ready state. If it is set to a large value, HPA may not be able to make a quick decision when a rapid response is needed. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default: 30s

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communicating with kube-apiserver	• If the number of nodes in a cluster is less than 1000, the default value is **100**. • If the number of nodes in a cluster is 1000 or more, the default value is **200**.
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communicating with kube-apiserver	• If the number of nodes in a cluster is less than 1000, the default value is **100**. • If the number of nodes in a cluster is 1000 or more, the default value is **200**.

Item	Parameter	Description	Value
The maximum number of terminated pods that can be kept before the Pod GC deletes the terminated pod	terminated-pod-gc-threshold	Number of terminated pods that can exist in a cluster. If there are more terminated pods than the expected number in the cluster, the terminated pods that exceed the number will be deleted. **NOTE** If this parameter is set to **0**, all pods in the terminated state are retained.	Default: 1000 Value range: 10 to 12500 If the cluster version is v1.21.11-r40, v1.23.8-r0, v1.25.6-r0, v1.27.3-r0, or later, the value range is changed to 0 to 100000.
Unhealthy AZ Threshold	unhealthy-zone-threshold	When more than a certain proportion of pods in an AZ are unhealthy, the AZ itself will be considered unhealthy, and scheduling pods to nodes in that AZ will be restricted to limit the impacts of the unhealthy AZ. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **NOTE** If the parameter is set to a large value, pods in unhealthy AZs will be migrated in a large scale, which may lead to risks such as overloaded clusters.	Default: 0.55 Value range: 0 to 1

Item	Parameter	Description	Value
Node Eviction Rate	node-eviction-rate	This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is healthy. The default value is **0.1**, indicating that pods can be evicted from at most one node every 10 seconds. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **NOTE** If the parameter is set to a large value, the cluster may be overloaded. Additionally, if too many pods are evicted, they cannot be rescheduled, which will slow down fault recovery.	Default: 0.1
Secondary Node Eviction Rate	secondary-node-eviction-rate	This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is unhealthy. The default value is **0.01**, indicating that pods can be evicted from at most one node every 100 seconds. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **NOTE** There is no need to set the parameter to a large value for nodes in an unhealthy AZ, and this configuration may result in overloaded clusters.	Default: 0.01 Configure this parameter with **node-eviction-rate** and set it to one-tenth of **node-eviction-rate**.

Item	Parameter	Description	Value
Large Cluster Threshold	large-cluster-size-threshold	If the number of nodes in a cluster is greater than the value of this parameter, this is a large cluster. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **NOTE** kube-controller-manager automatically adjusts configurations for large clusters to optimize the cluster performance. Therefore, an excessively small threshold for small clusters will deteriorate the cluster performance.	Default: 50 For the clusters with a large number of nodes, configure a relatively larger value than the default one for higher performance and faster responses of controllers. Retain the default value for small clusters. Before adjusting the value of this parameter in a production environment, check the impact of the change on cluster performance in a test environment.

Table 2-20 Networking components (available only for CCE Turbo clusters)

Item	Parameter	Description	Value
The minimum number of network cards bound to the container at the cluster level	nic-minimum-target	Minimum number of container ENIs bound to a node The parameter value must be a positive integer. The value **10** indicates that at least 10 container ENIs must be bound to a node. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.	Default: 10
Cluster-level node preheating container NIC upper limit check value	nic-maximum-target	After the number of ENIs bound to a node exceeds the **nic-maximum-target** value, CCE will not proactively pre-bind ENIs. Checking the upper limit of pre-bound container ENIs is enabled only when the value of this parameter is greater than or equal to the minimum number of container ENIs (**nic-minimum-target**) bound to a node. The parameter value must be a positive integer. The value **0** indicates that checking the upper limit of pre-bound container ENIs is disabled. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.	Default: 0

Item	Parameter	Description	Value
Number of NICs for dynamically warming up containers at the cluster level	nic-warm-target	Extra ENIs will be pre-bound after the **nic-minimum-target** is used up in a pod. The value can only be a number. When the sum of the **nic-warm-target** value and the number of ENIs bound to the node is greater than the **nic-maximum-target** value, CCE will pre-bind the number of ENIs specified by the difference between the **nic-maximum-target** value and the current number of ENIs bound to the node.	Default: 2
Cluster-level node warm-up container NIC recycling threshold	nic-max-above-warm-target	Only when the difference between the number of idle ENIs on a node and the **nic-warm-target** value is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number. ● A large value will accelerate pod startup but slow down the unbinding of idle container ENIs and decrease the IP address usage. **Exercise caution when performing this operation.** ● A small value will speed up the unbinding of idle container ENIs and increase the IP address usage but will slow down pod startup, especially when a large number of pods increase instantaneously.	Default: 2
Low threshold of the number of container ENIs bound to a node in a cluster	prebound-subeni-percentage	High threshold of the number of bound ENIs **NOTE** This parameter is being discarded. Use the dynamic pre-binding parameters of the other four ENIs.	Default: 0:0

Table 2-21 Networking component configurations (supported only by the clusters using a VPC network)

Item	Parameter	Description	Value
Retaining the non-masqueraded CIDR block of the original pod IP address	nonMasqueradeCIDRs	In a CCE cluster using the VPC network model, if a container in the cluster needs to access externally, the source pod IP address must be masqueraded as the IP address of the node where the pod resides through SNAT. After the configuration, the node will not SNAT the IP addresses in the CIDR block by default. This function is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. By default, nodes in a cluster do not perform SNAT on packets destined for 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16 that is detected by CCE as a private CIDR block. Instead, these packets are directly transferred using the upper-layer VPC. (The three CIDR blocks are considered as internal networks in the cluster and are reachable at Layer 3 by default.)	Default: 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16 **NOTE** To enable cross-node pod access, the CIDR block of the node where the target pod runs must be added. Similarly, to enable cross-ECS pod access in a VPC, the CIDR block of the ECS where the target pod runs must be added.

Table 2-22 Extended controller configurations (supported only by clusters of v1.21 and later)

Item	Parameter	Description	Value
Enable resource quota management	enable-resource-quota	Indicates whether to automatically create a ResourceQuota when creating a namespace. With quota management, you can control the number of workloads of each type and the upper limits of resources in a namespace or related dimensions. • **false**: Auto creation is disabled. • **true**: Auto creation is enabled. For details about the resource quota defaults, see **Configuring Resource Quotas**. **NOTE** In high-concurrency scenarios (for example, creating pods in batches), the resource quota management may cause some requests to fail due to conflicts. Do not enable this function unless necessary. To enable this function, ensure that there is a retry mechanism in the request client.	Default: false

Step 4 Click **OK**.

----End

References

- **kube-apiserver**
- **kube-controller-manager**
- **kube-scheduler**

2.5.2 Enabling Overload Control for a Cluster

Scenario

After overload control is enabled, the number of simultaneous requests is dynamically regulated according to the resource pressure on the master nodes. This ensures that both the nodes and the cluster remain accessible.

Notes and Constraints

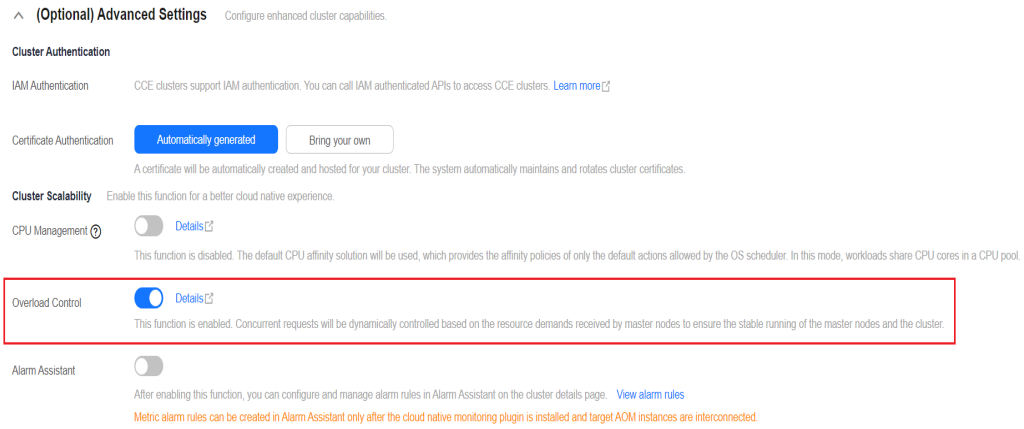
The cluster version must be 1.23 or later.

Enabling Overload Control

Method 1: Enabling it when creating a cluster

When creating a cluster of v1.23 or later, you can enable overload control during the cluster creation.

Figure 2-21 Enabling overload control during cluster creation

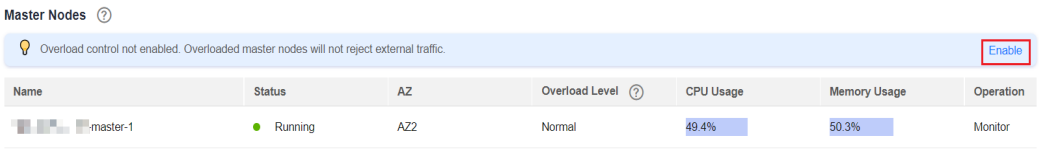


Method 2: Enabling it in an existing cluster

Step 1 Log in to the CCE console and click the name of an existing cluster whose version is v1.23 or later.

Step 2 On the **Overview** page, check the master node information. If overload control is not enabled, a message will be displayed. You can click **Enable** to enable the function.

Figure 2-22 Enabling overload control for an existing cluster



----End

Overload Monitoring

Method 1: Using the CCE console

Step 1 Log in to the CCE console and click the name of an existing cluster whose version is v1.23 or later.

Step 2 On the **Overview** page, check the master node information. The overload level metric will be displayed.

The overload levels are as follows:

- Circuit breaking: Rejects all external traffic.
- Severe overload: Rejects 75% external traffic.
- Moderate overload: Rejects 50% external traffic.
- Slight overload: Rejects 25% external traffic.
- Normal: Does not reject external traffic.

----End

Method 2: Using AOM

You can log in to the AOM console, create a dashboard, and add the metric named **vein_overload_level**. For details, see **Dashboard**.

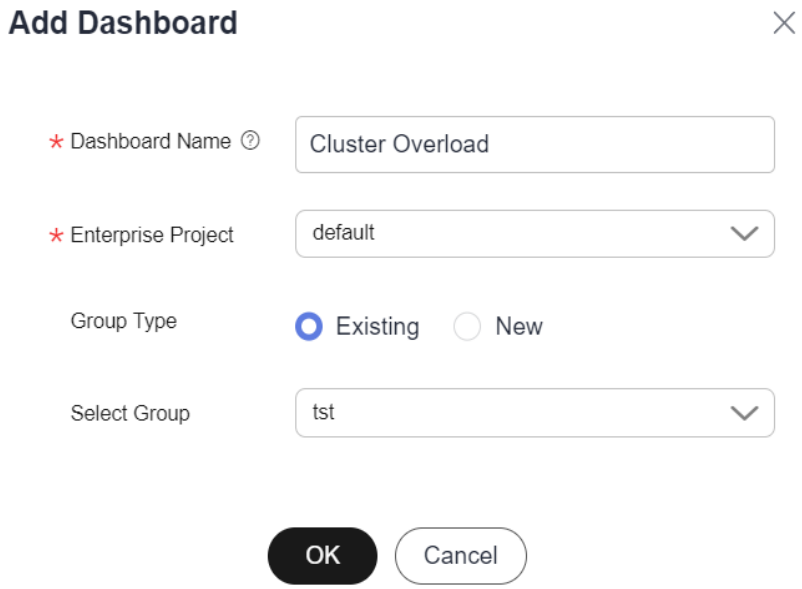
The meanings of the monitoring metrics are as follows:

- 0: Circuit breaking: Rejects all external traffic.
- 1: Severe overload: Rejects 75% external traffic.
- 2: Moderate overload: Rejects 50% external traffic.
- 3: Slight overload: Rejects 25% external traffic.
- 4: Normal: Does not reject external traffic.

The following uses the operations performed on AOM as an example.

Step 1 Log in to the AOM console, choose **Dashboard** in the navigation pane and click **Add Dashboard**.

Figure 2-23 Creating a dashboard

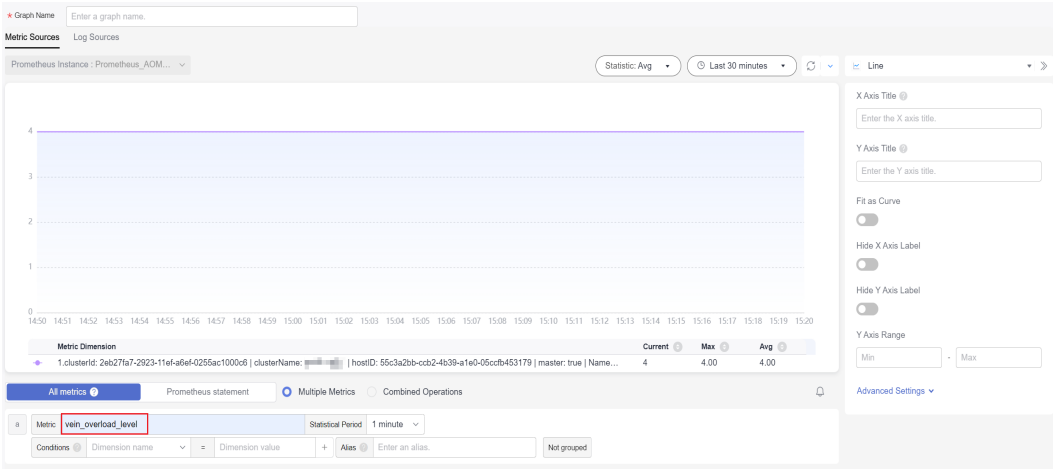


Step 2 After the dashboard is created, add a metric graph.

- **Graph Name**: **Cluster Overload Metrics**
- **Add By**: **All metrics**

- **Metric**: **vein_overload_level**
- **Scope**: Optional

Figure 2-24 Adding a metric graph



Step 3 After configuring parameters, click **Save** in the upper right corner.

Step 4 Click  in the upper right corner to save the dashboard.

Figure 2-25 Saving a dashboard

----End

Overload Alarm

The following uses the operations performed on AOM as an example.

Step 1 Log in to the AOM console. In the navigation pane, choose **Alarm Management** > **Alarm Rules**. Click **Create**.

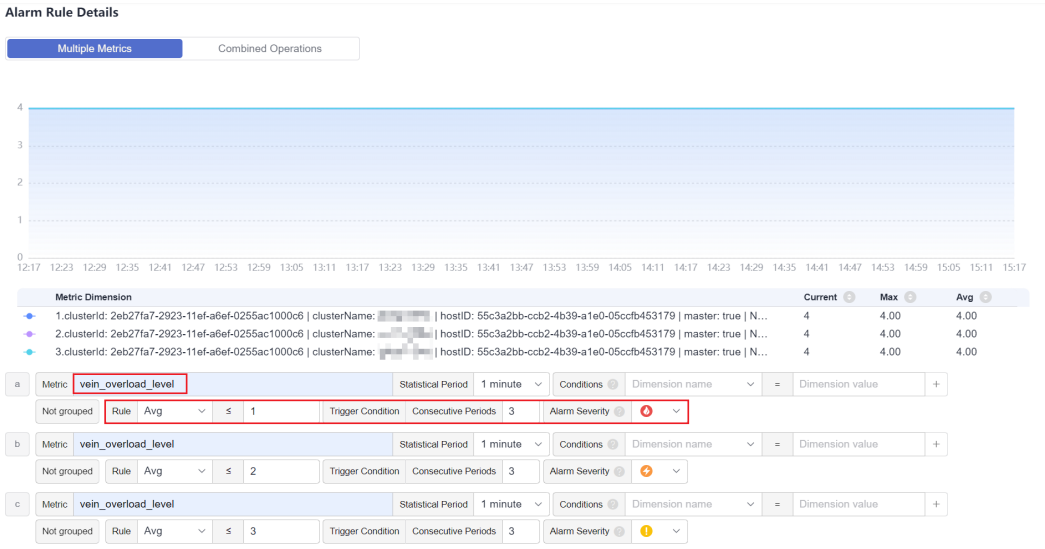
Step 2 Set the following parameters:

- **Rule Name**: Cluster overload alarms
- **Enterprise Project**: Enter **default**.
- **Rule Type**: **Metric alarm rule**

- **Configuration Mode**: **Select from all metrics**
- **Alarm Rule Details**: Select **vein_overload_level** and configure the corresponding scope.
  - If the metric value is less than or equal to **1**, the cluster is heavily overloaded. You are advised to set a critical alarm.
  - If the metric value is less than or equal to **2**, the cluster is moderately overloaded. You are advised to set a major alarm.
  - If the metric value is less than or equal to **3**, the cluster is slightly overloaded. You are advised to set a minor alarm.

Set other parameters as required.

Figure 2-26 Configuring an alarm rule for an overloaded cluster



Step 3 Click **Create Now**.

----End

Disabling Cluster Overload Control

Step 1 Log in to the CCE console and click the name of an existing cluster whose version is v1.23 or later.

Step 2 In the navigation pane, choose **Settings**.

Step 3 On the **Cluster Access** tab page, disable overload control.

Step 4 Click **OK**.

----End

2.5.3 Changing Cluster Scale

Scenario

CCE allows you to change the number of nodes managed in a cluster.

Notes and Constraints

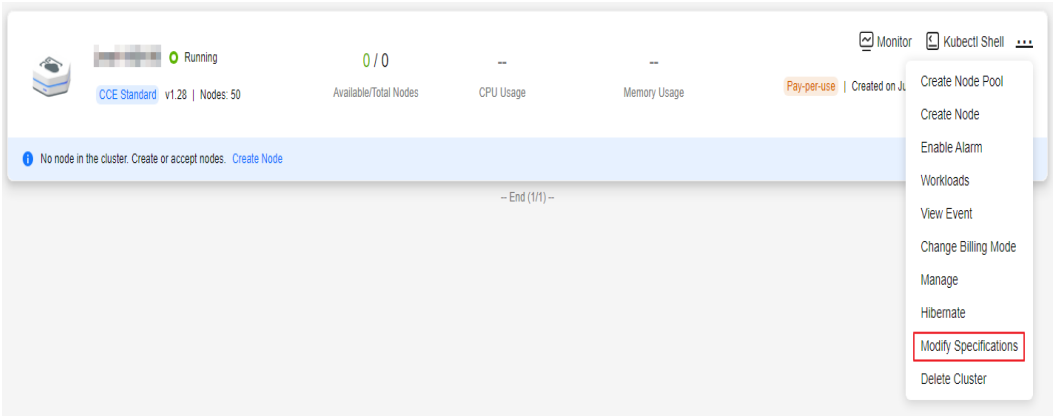
- A cluster that has only one master node supports fewer than 1000 worker nodes.
- The number of master nodes cannot be changed when you modify cluster specifications.
- A cluster can only be scaled out to a larger specification, but cannot be scaled in.
- To ensure proper cluster functionality, perform any specifications changes during off-peak hours. This is because master nodes will need to be powered off and on, which can disrupt normal operations.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster whose specifications need to be modified, click **...** to view more operations on the cluster, and choose **Modify Specifications**.

Figure 2-27 Modifying specifications



Step 3 On the page displayed, select a new cluster scale.

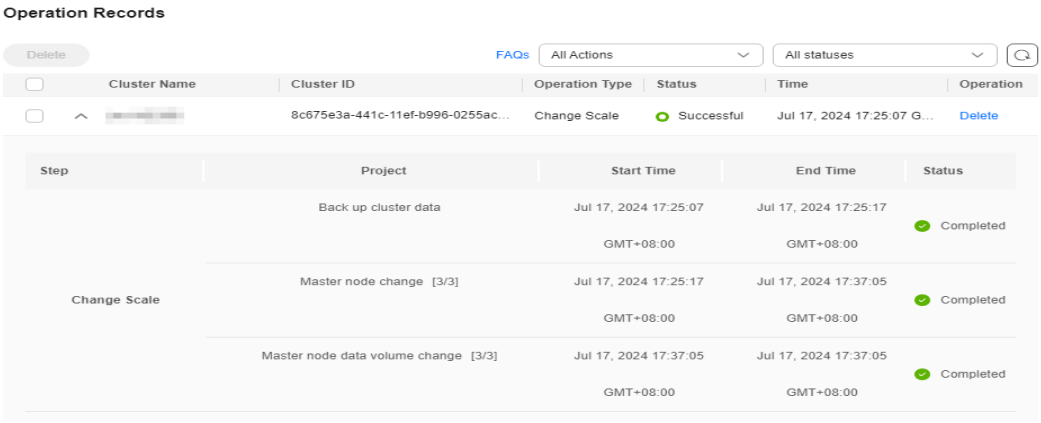
Step 4 Click **Next** to confirm the specifications and click **OK**.

You can click **Operation Records** in the upper right corner to view the cluster change history. The status changes from **Executing** to **Successful**, indicating that the cluster specifications are successfully changed.

NOTE

After the cluster scale is changed to 1000 nodes or more, some parameter values of the cluster will be automatically adjusted to ensure the cluster performance. For details, see **Modifying Cluster Configurations**.

Figure 2-28 Operation records



----End

2.5.4 Changing the Default Security Group of a Node

Scenario

When creating a cluster, you can customize a node security group to centrally manage network security policies. For a created cluster, you can change its default node security group.

Notes and Constraints

- Do not add more than 1000 pods to the same security group. Otherwise, the security group performance may be impacted. For more restrictions on security groups, see **Security Group Constraints**.
- Exercise caution when modifying the security group rules of a master node. For details, see **Configuring Cluster Security Group Rules**.

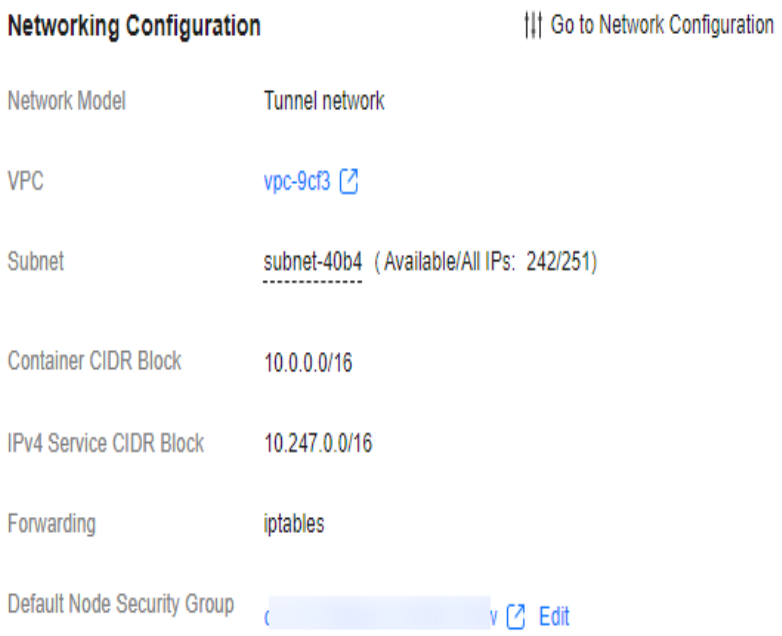
Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Click the cluster name to access the **Overview** page.

Step 3 In the **Networking Configuration** area, click **Edit** next to the **Default Node Security Group**.

Figure 2-29 Default node security group

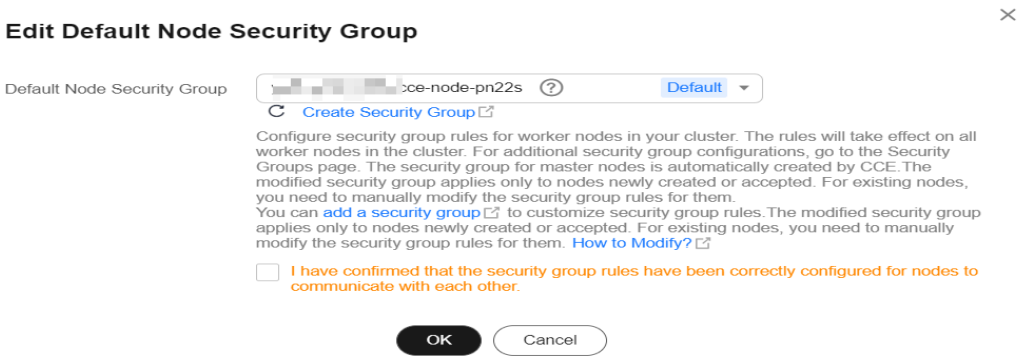


Step 4 Select an existing security group, confirm that the security group rules meet the cluster requirements, and click **OK**.

NOTICE

- Ensure that correct port rules are configured for the selected security group. Otherwise, the node cannot be created. The port rules that a security group must comply with vary with the cluster type. For details, see **Configuring Cluster Security Group Rules**.
- The new security group takes effect only for newly created or managed nodes. For existing nodes, modify the security group rules and reset the nodes in real time. The original security group is still used. For details about how to modify the security group settings of the existing nodes in batches, see **How Do I Change the Security Group of Nodes in a Cluster in Batches?**

Figure 2-30 Editing default node security group



----End

2.5.5 Deleting a Cluster

Scenario

- You can directly delete pay-per-use clusters. For details, see **Deleting a Pay-per-Use Cluster**.
- Yearly/Monthly clusters cannot be deleted directly. Unsubscribe from clusters that have not expired or release clusters that have expired and have not been renewed. For details, see **Unsubscribing from or Releasing a Yearly/Monthly Cluster**.

Precautions

- Deleting a cluster will not delete the yearly/monthly-billed resources in the cluster, and their billing continues.
- Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be recovered. Before performing this operation, ensure that related data has been backed up or migrated.
- If you choose to delete a cluster with the nodes in it, the system disks and data disks attached to the nodes will also be deleted. Back up data before the deletion.
- If you delete a cluster that is not running (for example, frozen or unavailable), associated resources, such as storage and networking resources, will remain.

Deleting a Pay-per-Use Cluster

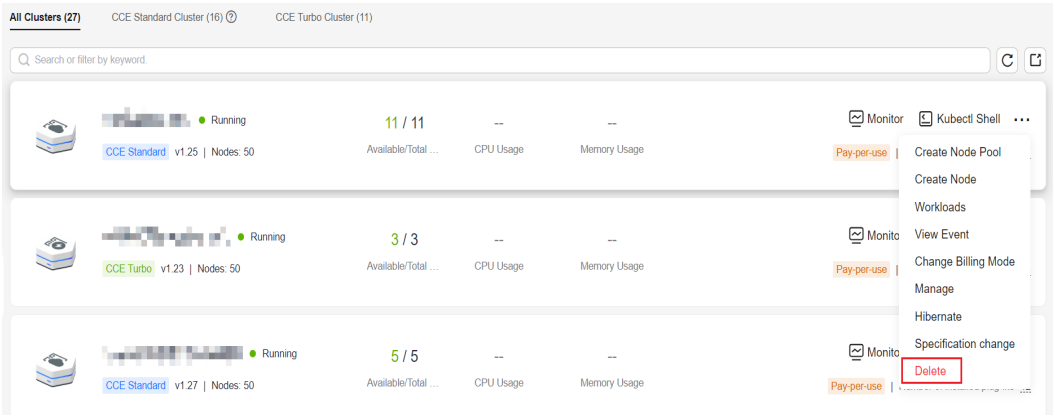
NOTICE

A hibernated cluster cannot be deleted. Wake up the cluster and try again.

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster to be deleted, click **...** to view more operations on the cluster, and choose **Delete**.

Figure 2-31 Deleting a cluster



Step 3 In the displayed **Delete Cluster** dialog box, select the resources to be released.

- When you delete nodes from a cluster, the following options are available:
  - **Retain**: The node as well as the data on the system disk and data disks will be retained.
  - **Delete**: The node will be deleted with the cluster. This option is available only for pay-per-use nodes. To delete a yearly/monthly node, manually unsubscribe from the node.
  - **Reset**: The node will be retained and reset. The data on the system disk and data disks will not be retained.
- Delete cloud storage resources associated with workloads in the cluster.

  NOTE

  When deleting underlying cloud storage resources bound to storage volumes in a cluster, pay attention to following constraints:

  - The underlying storage resources are deleted according to the reclamation policy you defined for the storage volumes. For example, if the reclamation policy of storage volumes is **Retain**, the underlying storage resources will be retained after the cluster is deleted.
  - If there are more than 1000 files in the OBS bucket, manually clear the files and then delete the cluster.

  If the storage volume's reclamation policy is set to **Retain**, the following table lists the method of deleting PVs based on their type.

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
EVS	Disks	Pay-per-use EVS disks will be deleted, but yearly/monthly EVS disks will be retained.
DSS	Disks	Yes
SFS	SFS Capacity-Oriented or SFS 3.0	Yes
OBS	OBS buckets or parallel file systems	Yes

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
SFS Turbo	SFS Turbo file systems	Yes
Local PV	Logical volumes mounted to the node	Determined based on the policy selected when the cluster is deleted. If you choose to retain the node, the logical volumes will not be deleted. If you choose to delete or reset the node, the logical volumes will be deleted.
SFS 3.0 subdirectory	A directory in SFS 3.0	No
SFS Turbo subdirectory	A directory in SFS Turbo	No

- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).
- Delete all log groups, including their log streams, for the cluster in LTS.

  NOTE

  If you do not select this option, the LTS logs will not be deleted, and you will be billed for them. For details, see **Price Calculator**.

Step 4 Enter **DELETE** and click **Yes** to start deleting the cluster.

The delete operation takes 1 to 3 minutes to complete.

----End

Unsubscribing from or Releasing a Yearly/Monthly Cluster

NOTICE

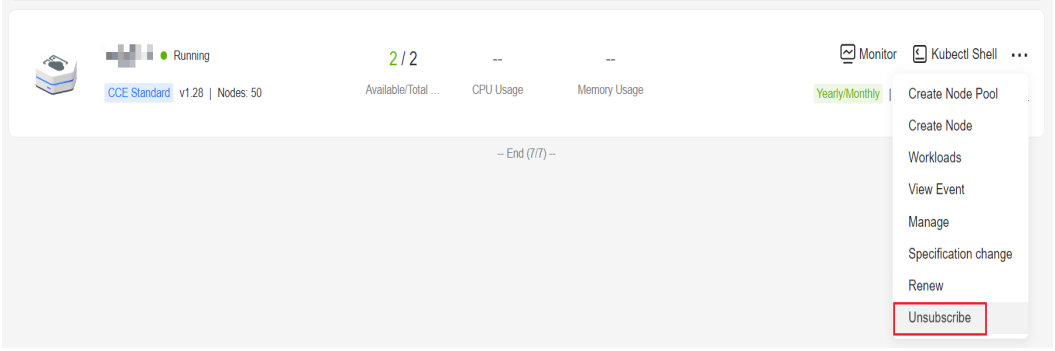
- When you unsubscribe from or release a cluster, only the resources associated with the order are unsubscribed from. Non-associated resources are retained and their billing keeps going.
- For a yearly/monthly cluster, if the retention expires, the cluster will be automatically released. For the nodes in the cluster, if they expire at the same time, they will also be released. If not, CCE will not perform any operation on your nodes. Node data is retained and the billing keeps going. Pay attention to the expired clusters under your account and renew them in a timely manner to prevent data loss caused by node reinstallation.
- If an order contains resources in a primary-secondary relationship, unsubscribe from the resources separately.
- For details about unsubscription rules, see **Unsubscription Rules**.
  - If you are unsubscribing from a resource that is being used, review the resource information and refund information carefully. The resource cannot be restored after unsubscription. You can unsubscribe from a renewal period that has not yet taken effect on renewed resources on the **Unsubscribe from Renewal Period** page.
  - Unsubscribing from a resource associated with other yearly/monthly-billed resources may affect the normal use of those resources.

    Unsubscribing from a resource associated with other pay-per-use resources will not affect the normal use of those resources. They will be billed normally.
  - If your operation is not a five-day unconditional unsubscription, you will be charged for the handling fee and the used amount. Used cash coupons and discount coupons will not be refunded.

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster to be unsubscribed from, click **...** to view more operations on the cluster, and choose **Unsubscribe** or **Release**.

Figure 2-32 Unsubscribing from a cluster



Step 3 On the displayed page, select the resources to be released.

- When you delete nodes from a cluster, the following options are available:
  - **Retain**: The node as well as the data on the system disk and data disks will be retained.
  - **Delete**: The node will be deleted with the cluster. This option is available only for pay-per-use nodes. To delete a yearly/monthly node, manually unsubscribe from the node.
  - **Reset**: The node will be retained and reset. The data on the system disk and data disks will not be retained.
- Delete cloud storage resources associated with workloads in the cluster.

  NOTE

  When deleting underlying cloud storage resources bound to storage volumes in a cluster, pay attention to following constraints:

  - The underlying storage resources are deleted according to the reclamation policy you defined for the storage volumes. For example, if the reclamation policy of storage volumes is **Retain**, the underlying storage resources will be retained after the cluster is deleted.
  - If there are more than 1000 files in the OBS bucket, manually clear the files and then delete the cluster.

  If the storage volume's reclamation policy is set to **Retain**, the following table lists the method of deleting PVs based on their type.

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
EVS	Disks	Pay-per-use EVS disks will be deleted, but yearly/monthly EVS disks will be retained.
DSS	Disks	Yes
SFS	SFS Capacity-Oriented or SFS 3.0	Yes
OBS	OBS buckets or parallel file systems	Yes
SFS Turbo	SFS Turbo file systems	Yes
Local PV	Logical volumes mounted to the node	Determined based on the policy selected when the cluster is deleted. If you choose to retain the node, the logical volumes will not be deleted. If you choose to delete or reset the node, the logical volumes will be deleted.
SFS 3.0 subdirectory	A directory in SFS 3.0	No

PV Type	Associated Underlying Storage	Whether to Delete Underlying Storage
SFS Turbo subdirectory	A directory in SFS Turbo	No

- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).
- Delete all log groups, including their log streams, for the cluster in LTS.

  NOTE

  If you do not select this option, the LTS logs will not be deleted, and you will be billed for them. For details, see **Price Calculator**.

Step 4 Click **Yes**. The unsubscription or release takes 1 to 3 minutes to complete.

----End

2.5.6 Hibernating or Waking Up a Pay-per-Use Cluster

Scenario

If a pay-per-use cluster is not needed temporarily, hibernate it to reduce costs.

After a cluster is hibernated, resources such as workloads cannot be created or managed in the cluster.

Precautions

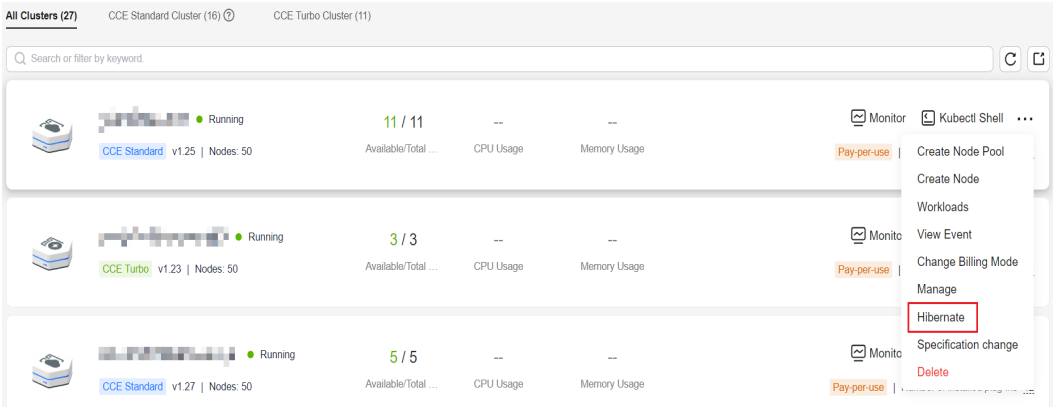
- During cluster wakeup, the master node may fail to start due to insufficient resources, which leads to a cluster wakeup failure. In this case, wait for a while and try again.
- After a cluster is woken up, it takes 3 to 5 minutes to initialize data. Deliver services after the cluster runs properly.

Hibernating a Cluster

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster to be hibernated, click **...** to view more operations on the cluster, and choose **Hibernate**.

Figure 2-33 Hibernating a cluster

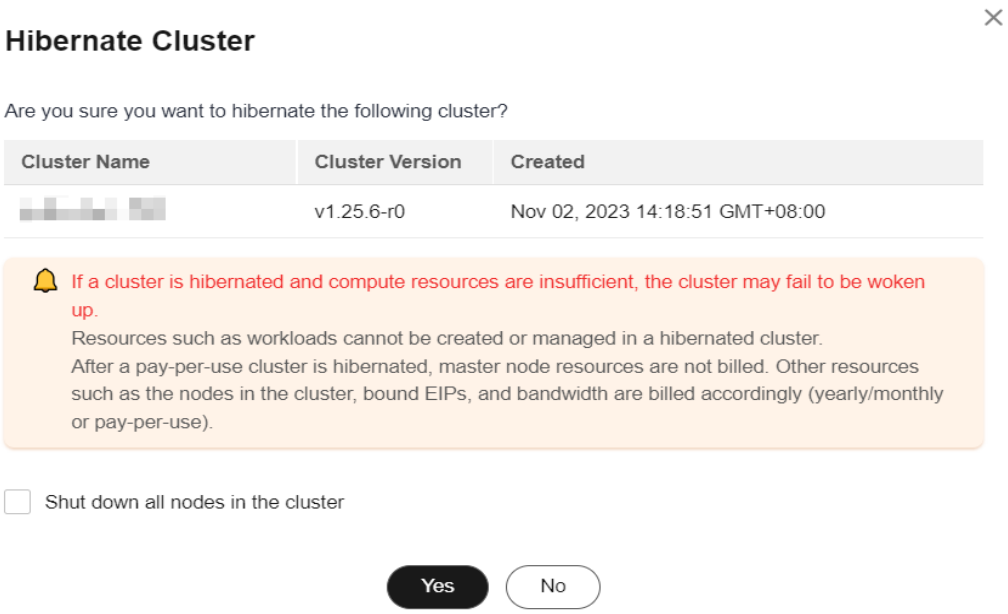


Step 3 In the dialog box displayed, check the precautions and click **Yes**. Wait until the cluster is hibernated.

After a cluster is hibernated, the billing of master node resources will stop. Resources, such as worker nodes (ECSs), bound EIPs, and bandwidth, are still billed based on their own billing modes. To shut down nodes, select **Shut down all nodes in the cluster** in the dialog box or see **Stopping a Node**.

Most nodes are no longer billed after they are stopped, excluding certain types of ECSs (ones with local disks attached, such as disk-intensive and ultra-high I/O ECSs). For details, see **ECS Billing**.

Figure 2-34 Prompt information



----End

Waking Up a Cluster

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Click **Wake Up** in the row of the target cluster.

Step 3 When the cluster status changes from **Waking up** to **Running**, the cluster is woken up. It takes about 3 to 5 minutes to wake up the cluster.

NOTE

After the cluster is woken up, the cluster management fee continues to be billed.

----End

2.5.7 Renewing a Yearly/Monthly Cluster

You can renew a yearly/monthly-billed cluster.

Procedure

This section describes how to renew a cluster billed on a **yearly/monthly** basis.

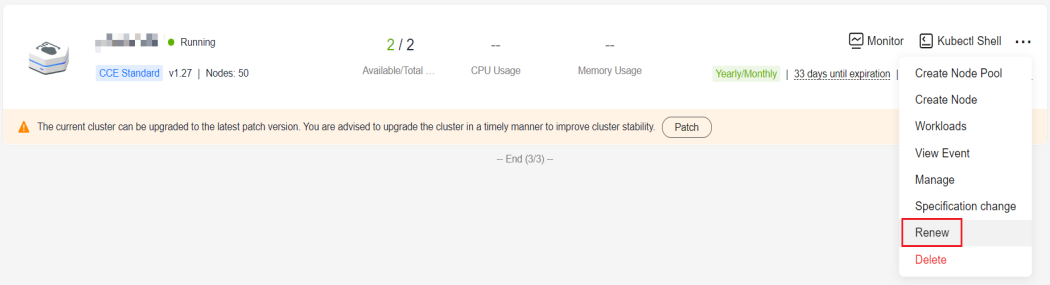
NOTICE

A yearly/monthly-billed cluster will be deleted if it is not renewed after expiration, and all nodes and the running services in the cluster will be destroyed. CCE strongly recommends that you renew the cluster before it expires or **enable auto renewal**.

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster to be renewed, click **...** to view more operations on the cluster, and choose **Renew**.

Figure 2-35 Renewing a cluster



Step 3 On the displayed page, renew the service as prompted.

NOTE

- If the selected resource (highlighted) is associated with other resources, you can decide whether you want to perform the operation on all these resources at the same time.
- If you change the resource specifications before its renewal period takes effect, the renewal period cannot be unsubscribed from.
- Renewed resources are not eligible of a 5-day unconditional unsubscription.

Step 4 Click **Pay**. On the page displayed, review the order amount, select a payment method, and click **Pay**.

Step 5 After the payment is complete, you can go back to the **Orders** or **Renewals** page to view and manage your order.

----End

2.5.8 Changing the Billing Mode of a Cluster from Pay-per-Use to Yearly/Monthly

A cluster can be billed on a pay-per-use or yearly/monthly basis. The billing mode of a cluster can be changed from pay-per-use to yearly/monthly.

To change the billing mode of a node from pay-per-use to yearly/monthly, see **Changing the Billing Mode of a Node to Yearly/Monthly**.

Changing the Billing Mode of a Cluster

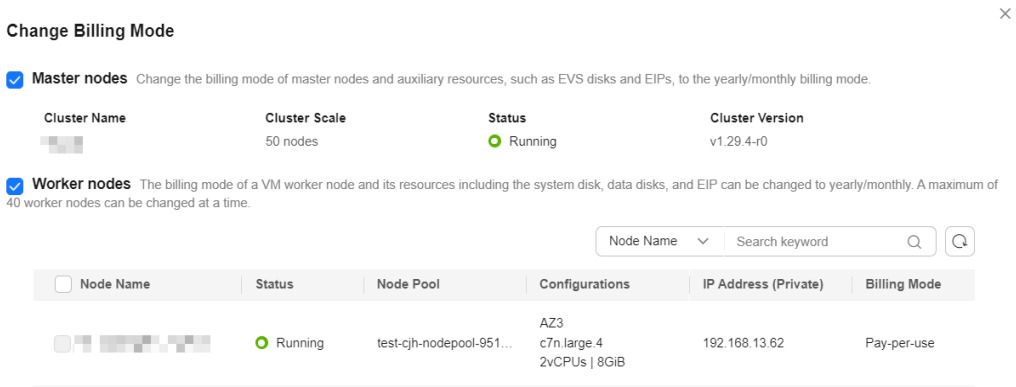
To change the billing mode of a cluster from pay-per-use to yearly/monthly, perform the following steps:

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**.

Step 2 Locate the target cluster, click **...** to view more operations on the cluster, and choose **Change Billing Mode**.

Step 3 On the page displayed, select the target cluster. You can also select the nodes whose billing modes you want to change.

Figure 2-36 Changing the billing mode of a cluster to yearly/monthly



Step 4 Click **OK**. Wait until the order is processed and the payment is complete.

----End

2.6 Upgrading a Cluster

2.6.1 Process and Method of Upgrading a Cluster

CCE strictly complies with community consistency authentication. It releases three Kubernetes versions each year and offers a maintenance period of at least 24 months after each version is released. CCE ensures the stable running of Kubernetes versions during the maintenance period.

To ensure your service rights and benefits, upgrade your Kubernetes clusters before a maintenance period ends. You can check the Kubernetes version of your cluster on the cluster list page and check whether a new version is available. Proactive cluster upgrades help you:

- Reduce security and stability risks: During the iteration of Kubernetes versions, known security and stability vulnerabilities are continuously fixed. Long-term use of EOS clusters will result in security and stability risks to services.
- Experience the latest functions: During the iteration of Kubernetes versions, new functions and optimizations are continuously released. For details about the features of the latest version, see **Release Notes for CCE Cluster Versions**.
- Minimize compatibility risks: During the iteration of Kubernetes versions, APIs are continuously modified and functions are deprecated. If a cluster has not been upgraded for a long time, more O&M assurance investment will be required when the cluster is upgraded. Periodic upgrades can effectively mitigate compatibility risks caused by accumulated version differences. It is a good practice to upgrade a patch version every quarter and upgrade a major version to the latest version every year.
- Obtain more effective technical support: CCE does not provide security patches or issue fixing for EOS Kubernetes cluster versions, and does not ensure technical support for the EOS versions.

Cluster Upgrade Path

CCE clusters evolve iteratively based on the community Kubernetes version. A CCE cluster version consists of the community Kubernetes version and the CCE patch version. Therefore, two cluster upgrade paths are provided.

- Upgrading a Kubernetes version

Source Kubernetes Version	Target Kubernetes Version
v1.13 or earlier	Not supported
v1.15	v1.19
v1.17	v1.19
v1.19	v1.21 or v1.23
v1.21	v1.23
v1.23	v1.25, v1.27, or v1.28
v1.25	v1.27 or v1.28
v1.27	v1.28
v1.28	v1.29
v1.29	v1.30

NOTE

- A version that has been end of maintenance cannot be directly upgraded to the latest version. You need to upgrade such a version for multiple times, for example, from v1.15 to v1.19, v1.23, and then to v1.27 or v1.28.
- A Kubernetes version can be upgraded only after the patch is upgraded to the latest version. CCE will automatically generate an optimal upgrade path on the console based on the current cluster version.

- Upgrading a patch version

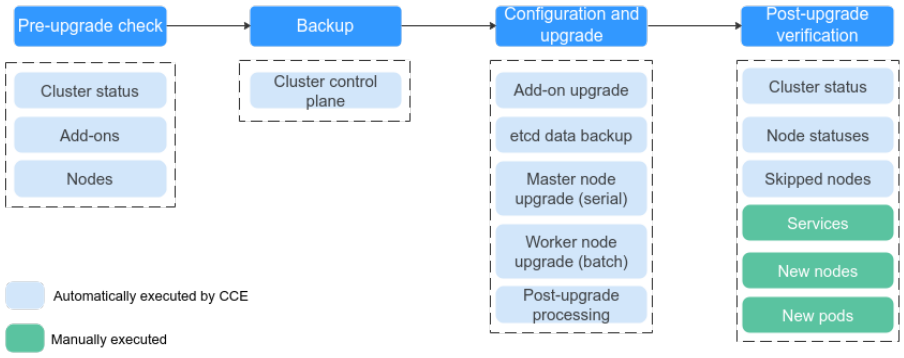
  Patch version management is available for CCE clusters of v1.19 or later to provide new features and fix bugs and vulnerability for in-maintenance clusters without requiring a major version upgrade.

  After a new patch version is released, you can directly upgrade any patch version to the latest patch version. For details about the release history of patch versions, see **Patch Version Release Notes**.

Cluster Upgrade Process

The cluster upgrade process involves pre-upgrade check, backup, upgrade, and post-upgrade verification.

Figure 2-37 Process of upgrading a cluster



After determining the target version of the cluster, read the **precautions** carefully and prevent function incompatibility during the upgrade.

1. **Pre-upgrade check**

   Before a cluster upgrade, CCE checks mandatory items such as the cluster status, add-ons, workload compatibility, and nodes to ensure that the cluster meets the upgrade requirements. For more details, see **Pre-upgrade Check**. If any check item is abnormal, rectify the fault as prompted on the console.

2. **Backup**

   You can use disk snapshots to back up master node data, including CCE component images, component configurations, and etcd data. Back up data before an upgrade. If unexpected cases occur during an upgrade, you can use the backup to quickly restore the cluster.

Backup Type	Backup Object	Backup Mode	Backup Duration	Rollback Duration	Description
etcd data backup	etcd data	Automatic backup during an upgrade	1-5 minutes	2 hours	Mandatory. The data is automatically backed up during an upgrade.
CBR cloud server backup	Master node disks, including component images, configurations, logs, and etcd data	One-click backup on a web page (manually triggered)	20 minutes to 2 hours (based on the cloud backup tasks in the current region)	20 minutes	This function is gradually replaced by EVS snapshot backup.
EVS snapshot backup	Master node disks, including component images, configurations, logs, and etcd data	One-click backup on a web page (manually triggered)	1-5 minutes	20 minutes	This function is coming soon. After this function is released, it will replace CBR cloud server backup.

3. **Configuration and upgrade**

   Configure parameters before an upgrade. CCE has provided default settings, which can be modified as needed. After the configuration, upgrade add-ons, master nodes, and worker nodes in sequence.

   - **Add-on Upgrade Configuration**: Add-ons that have been installed in your cluster are listed. During the cluster upgrade, CCE automatically upgrades the selected add-ons to be compatible with the target cluster version. You can click **Set** to re-define the add-on parameters.

     NOTE

     If an add-on is marked with ⚠ on its right side, the add-on cannot be compatible with both the source and target versions of the cluster upgrade. In this case, CCE will upgrade the add-on after the cluster upgrade. The add-on may be unavailable during the cluster upgrade.

   - **Node Upgrade Configuration**

- **Max. Nodes for Batch Upgrade**: You can configure the maximum number of nodes to be upgraded in a batch.

  Node pools will be upgraded in sequence. Nodes in node pools will be upgraded in batches. One node is upgraded in the first batch, two nodes in the second batch, and the number of nodes to be upgraded in each subsequent batch increases by a power of 2 until the maximum number of nodes to be upgraded in each batch is reached. The next cluster is upgraded after the previous one is upgraded. By default, 20 nodes are upgraded in a batch, and the number can be increased to the maximum of 120.

- **Node Priority**: You can customize node upgrade priorities. If the priorities are not specified, CCE will perform the upgrade based on the priorities generated by the default policy.
  - **Add Upgrade Priority**: You can custom the priorities for upgrading node pools. If the priorities are not specified, CCE will preferentially upgrade the node pool with the least number of nodes based on the default policy.
  - **Add Node Priority**: You can custom the priorities for upgrading nodes in a node pool. If the priorities are not specified, CCE will preferentially upgrade the node with lightest load (calculated based on the number of pods, resource request rate, and number of PVs) based on the default policy.

- **Scope of Node Upgrade Batches**: By default, this parameter is set to a cluster, but it can be customized.
  - If the scope is set to a cluster, the upgrade batch will remain unchanged throughout the entire upgrade process.
  - If the scope is set to node pools, the upgrade batch will be reset for each node pool separately.

4. **Post-upgrade verification**

   After an upgrade, CCE will automatically check items including the cluster status and node status. You need to manually check services, new nodes, and new pods to ensure that the cluster functions properly after the upgrade. For details, see **Performing Post-Upgrade Verification**.

Upgrade Modes

Table 2-23 Upgrade modes

Upgrade Mode	Description	Upgrade Scope	Advantage	Constraint
In-place upgrade	Kubernetes components, network components, and CCE management components are upgraded on nodes. During an upgrade, service pods and networks are not affected. Nodes are upgraded in batches. Only the nodes that have been upgraded can be used to schedule services.	• Node OSs are not upgraded. • The add-ons that are incompatible with the target cluster version will be automatically upgraded. • Kubernetes components will be automatically upgraded.	The one-click upgrade does not need to migrate services. This ensures service continuity.	In-place upgrade is supported only in clusters of v1.15 or later.
Migration	The services running in a cluster of an earlier version are migrated to a cluster of a later version. This mode is suitable for cross-version cluster upgrades.	All resources in the cluster must be redeployed.	This mode prevents version incompatibility caused by consecutive upgrades of earlier versions.	None

2.6.2 Before You Start

Before the upgrade, you can check whether your cluster can be upgraded and which versions are available on the CCE console. For details, see **Process and Method of Upgrading a Cluster**.

Precautions

Before upgrading a cluster, pay attention to the following points:

- **Perform an upgrade during off-peak hours to minimize the impact on your services.**

- Before upgrading a cluster, learn about the features and differences of each cluster version in **Kubernetes Release Notes** to prevent exceptions due to the use of an incompatible cluster version. For example, check whether any APIs deprecated in the target version are used in the cluster. Otherwise, calling the APIs may fail after the upgrade. For details, see **Deprecated APIs**.

During a cluster upgrade, pay attention to the following points that may affect your services:

- During a cluster upgrade, do not perform any operation on the cluster. If you **stop, restart, or delete nodes** while upgrading the cluster, the upgrade will fail.
- Before upgrading a cluster, **ensure no high-risk operations are performed in the cluster**. Otherwise, the cluster upgrade may fail or the configuration may be lost after the upgrade. Common high-risk operations include modifying cluster node configurations locally and modifying the configurations of the listeners managed by CCE on the ELB console. Instead, modify configurations on the CCE console so that the modifications can be automatically inherited during the upgrade.
- During a cluster upgrade, the running workloads will not be interrupted, but access to the API server will be temporarily interrupted.
- By default, application scheduling is not restricted during a cluster upgrade. During an upgrade of the following early cluster versions, the **node.kubernetes.io/upgrade** taint (equivalent to **NoSchedule**) will be added to the nodes in the cluster and removed after the cluster is upgraded:
  - All v1.15 clusters
  - All v1.17 clusters
  - v1.19 clusters with patch versions earlier than or equal to v1.19.16-r4
  - v1.21 clusters with patch versions earlier than or equal to v1.21.7-r0
  - v1.23 clusters with patch versions earlier than or equal to v1.23.5-r0
- During a cluster upgrade, if an add-on is also upgraded and cluster resources are limited, add-on pods can use resources that would otherwise be allocated to service pods. This may result in the eviction of service pods. After the add-on is upgraded, the evicted service pods will automatically recover.

Notes and Constraints

- If an error occurred during a cluster upgrade, the cluster can be rolled back using the backup data. If you perform other operations (for example, modifying cluster specifications) after a successful cluster upgrade, the cluster cannot be rolled back using the backup data.
- If your cluster is upgraded to v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or a later version, the following add-ons must be upgraded to the target version if these add-ons are installed in the cluster (for details, see **Data Disk Shared Between a Container Engine and kubelet Components**):
  - NPD: v1.18.10 or later
  - log-agent: v1.3.0 or later
- When clusters using the tunnel network model are upgraded to v1.19.16-r4, v1.21.7-r0, v1.23.5-r0, v1.25.1-r0, or later, the SNAT rule whose destination address is the container CIDR block but the source address is not the

container CIDR block will be removed. If you have configured VPC routes to directly access all pods outside the cluster, only the pods on the corresponding nodes can be directly accessed after the upgrade.

- The new add-on versions (see **Change History**) of the NGINX Ingress Controller allow graceful exit and the grace period for deleting the ELB backend controller and support hitless upgrades. During the upgrade of the add-on versions listed in the following table, services may be unavailable for a short period of time. If the following add-on versions have been installed in the cluster, perform the upgrade during off-peak hours.

Add-on Version	Version Range
2.1.x	x < 32
2.2.x	x < 41
2.4.x	x < 4

- Upgrading a cluster will restart NetworkManager. This will trigger the DHCP client to renew IP address leasing. By default, **/etc/resolv.conf** is updated based on the subnet DNS configuration. Modify the DNS DNS configuration on the VPC console. For details, see **How Do I Change the DNS Server Address of an ECS?**
- For more details, see **Version Differences**.

Version Differences

Upgrade Path	Version Difference	Self-Check
v1.23 or v1.25 Upgraded to v1.27	Docker is no longer recommended. Use containerd instead. For details, see **Container Engines**.	This item has been included in the pre-upgrade check.

Upgrade Path	Version Difference	Self-Check
v1.21 or v1.19 Upgraded to v1.23	For the NGINX Ingress Controller of an earlier version (community version v0.49 or earlier, or CCE nginx-ingress version v1.x.x), the created ingresses can be managed by the NGINX Ingress Controller even if **kubernetes.io/ingress.class: nginx** is not set in the ingress **annotations**. However, for the NGINX Ingress Controller of a later version (community version v1.0.0 or later, or CCE nginx-ingress version v2.x.x), the ingresses created without specifying the Nginx type will not be managed by the NGINX Ingress Controller, and ingress rules will become invalid, which interrupts services.	This item has been included in the pre-upgrade check. You can also perform the self-check by referring to **NGINX Ingress Controller**.
v1.19 to v1.21	The bug of **exec probe timeouts** is fixed in Kubernetes 1.21. Before this bug is fixed, the exec probe does not consider the **timeoutSeconds** field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. If this field is not specified, the default value **1** is used. This field takes effect after the upgrade. If the probe runs over 1 second, the application health check may fail and the application may restart frequently.	Before the upgrade, check whether the timeout is properly set for the exec probe.

Upgrade Path	Version Difference	Self-Check
	kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly. Root cause: X.509 **CommonName** is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the **CommonName** field of the X.509 certificate as the host name by default. As a result, the authentication fails.	Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server. • If you do not have your own webhook server, you can skip this check. • If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate.
v1.15 to v1.19	The control plane of CCE clusters of v1.19 is incompatible with kubelet v1.15. If a node fails to be upgraded or the node to be upgraded restarts after the master node is successfully upgraded, there is a high probability that the node is in the **NotReady** status. This is because the node failed to be upgraded restarts the kubelet and trigger the node registration. In clusters of v1.15, the default registration tags (**failure-domain.beta.kubernetes.io/is-baremetal** and **kubernetes.io/availablezone**) are regarded as invalid tags by the clusters of v1.19. The valid tags in the clusters of v1.19 are **node.kubernetes.io/baremetal** and **failure-domain.beta.kubernetes.io/zone**.	1. In normal cases, this scenario is not triggered. 2. After the master node is upgraded, do not suspend the upgrade so the node can be quickly upgraded. 3. If a node fails to be upgraded and cannot be restored, evict applications on the node as soon as possible. Contact technical support and skip the node upgrade. After the upgrade is complete, reset the node.

Upgrade Path	Version Difference	Self-Check

	In CCE v1.15 and v1.19 clusters, the Docker storage driver file system has been changed from XFS to Ext4. As a result, the import package sequence may be abnormal in the pods of upgraded Java application, leading to pod exceptions.	Before the upgrade, check the Docker configuration file **/etc/docker/daemon.json** on the node. Check whether the value of **dm.fs** is **xfs**.
		• If the value is **ext4** or the storage driver is Overlay, you can skip the next steps.
		• If the value is **xfs**, you are advised to deploy applications in the cluster of the new version in advance to test whether the applications are compatible with the new cluster version.
		{
		"storage-driver": "devicemapper",
		"storage-opts": [
		"dm.thinpooldev=/dev/mapper/vgpaas-thinpool",
		"dm.use_deferred_removal=true",
		"dm.fs=xfs",
		"dm.use_deferred_deletion=true"
		]
		}

	kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly. Root cause: X.509 **CommonName** is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. The **CommonName** field is processed as the host name. As a result, the authentication fails.	Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server. • If you do not have your own webhook server, you can skip this check. • If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate. **NOTICE** To mitigate the impact of version differences on cluster upgrade, CCE performs special processing during the upgrade from v1.15 to v1.19 and still supports certificates without SANs. However, no special processing is required for subsequent upgrades. You are advised to rectify your certificate as soon as possible.

Upgrade Path	Version Difference	Self-Check
	In clusters of v1.17.17 and later, CCE automatically creates pod security policies (PSPs) for you, which restrict the creation of pods with unsafe configurations, for example, pods for which **net.core.somaxconn** under a sysctl is configured in the security context.	After an upgrade, you can allow insecure system configurations as required. For details, see **Configuring a Pod Security Policy**.
	If initContainer or Istio is used in the in-place upgrade of a cluster of v1.15, pay attention to the following restrictions: In kubelet v1.16 and later versions, **QoS classes** are different from those in earlier versions. In kubelet v1.15 and earlier versions, only containers in **spec.containers** are counted. In kubelet v1.16 and later versions, containers in both **spec.containers** and **spec.initContainers** are counted. The QoS class of a pod will change after the upgrade. As a result, the container in the pod restarts.	You are advised to modify the QoS class of the service container before the upgrade to avoid this problem. For details, see **Table 2-24**.
v1.13 to v1.15	After a VPC network cluster is upgraded, the master node occupies an extra CIDR block due to the upgrade of network components. If no container CIDR block is available for the new node, the pod scheduled to the node cannot run.	Generally, this problem occurs when the nodes in the cluster are about to fully occupy the container CIDR block. For example, the container CIDR block is 10.0.0.0/16, the number of available IP addresses is 65,536, and the VPC network allocates a CIDR block with the fixed size (using the mask to determine the maximum number of container IP addresses allocated to each node). If the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes, including the three master nodes. After the cluster is upgraded, each of the three master nodes occupies one CIDR block. As a result, 506 nodes are supported.

Table 2-24 QoS class changes before and after the upgrade

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Guaranteed	Besteffort	Burstable	Yes
Guaranteed	Burstable	Burstable	No
Guaranteed	Guaranteed	Guaranteed	No
Besteffort	Besteffort	Besteffort	No
Besteffort	Burstable	Burstable	No
Besteffort	Guaranteed	Burstable	Yes
Burstable	Besteffort	Burstable	Yes
Burstable	Burstable	Burstable	No
Burstable	Guaranteed	Burstable	Yes

Deprecated APIs

With the evolution of Kubernetes APIs, APIs are periodically reorganized or upgraded, and certain APIs are deprecated and finally deleted. The following tables list the deprecated APIs in each Kubernetes community version. For details about more deprecated APIs, see **Deprecated API Migration Guide**.

- **APIs Deprecated in Kubernetes v1.29**
- No APIs deprecated in Kubernetes v1.28
- **APIs Deprecated in Kubernetes v1.27**
- **APIs Deprecated in Kubernetes v1.25**
- **APIs Deprecated in Kubernetes v1.22**
- **APIs Deprecated in Kubernetes v1.16**

NOTE

When an API is deprecated, the existing resources are not affected. However, when you create or edit the resources, the API version will be intercepted.

Table 2-25 APIs deprecated in Kubernetes v1.29

Resource	Deprecated API Version	Substitute API Version	Change Description
FlowSchema and PriorityLevelConfiguration	flowcontrol.apiserver.k8s.io/v1beta2	flowcontrol.apiserver.k8s.io/v1 (This API has been available since v1.29.) flowcontrol.apiserver.k8s.io/v1beta3 (This API has been available since v1.26.)	• Significant changes in **flowcontrol.apiserver.k8s.io/v1**: **spec.limited.assuredConcurrencyShares** of **PriorityLevelConfiguration** has been renamed **spec.limited.nominalConcurrencyShares**. The default value is **30** only when it is not specified, and the explicit value **0** does not change to **30**. • Key changes in **flowcontrol.apiserver.k8s.io/v1beta3**: **spec.limited.assuredConcurrencyShares** of **PriorityLevelConfiguration** has been renamed **spec.limited.nominalConcurrencyShares**.

Table 2-26 APIs deprecated in Kubernetes v1.27

Resource	Deprecated API Version	Substitute API Version	Change Description
CSIStorageCapacity	storage.k8s.io/v1beta1	storage.k8s.io/v1 (This API has been available since v1.24.)	None
FlowSchema and PriorityLevelConfiguration	flowcontrol.apiserver.k8s.io/v1beta1	flowcontrol.apiserver.k8s.io/v1beta3 (This API has been available since v1.26.)	None
HorizontalPodAutoscaler	autoscaling/v2beta2	autoscaling/v2 (This API has been available since v1.23.)	None

Table 2-27 APIs deprecated in Kubernetes v1.25

Resource	Deprecated API Version	Substitute API Version	Change Description
CronJob	batch/v1beta1	batch/v1 (This API has been available since v1.21.)	None
EndpointSlice	discovery.k8s.io/v1beta1	discovery.k8s.io/v1 (This API has been available since v1.21.)	Pay attention to the following changes: • In each endpoint, the **topology["kubernetes.io/hostname"]** field has been deprecated. Replace it with the **nodeName** field. • In each endpoint, the **topology["kubernetes.io/zone"]** field has been deprecated. Replace it with the **zone** field. • The **topology** field is replaced with **deprecatedTopology** and cannot be written in v1.

Resource	Deprecated API Version	Substitute API Version	Change Description
Event	events.k8s.io/v1beta1	events.k8s.io/v1 (This API has been available since v1.19.)	Pay attention to the following changes: • The **type** field can only be set to **Normal** or **Warning**. • The **involvedObject** field is renamed **regarding**. • The **action**, **reason**, **reportingController**, and **reportingInstance** fields are mandatory for creating a new **events.k8s.io/v1** event. • Use **eventTime** instead of the deprecated **firstTimestamp** field (this field has been renamed **deprecatedFirstTimestamp** and is not allowed to appear in the new **events.k8s.io/v1** event object). • Use **series.lastObservedTime** instead of the deprecated **lastTimestamp** field (this field has been renamed **deprecatedLastTimestamp** and is not allowed to appear in the new **events.k8s.io/v1** event object). • Use **series.count** instead of the deprecated **count** field (this field has been renamed **deprecatedCount** and is not allowed to appear in the new **events.k8s.io/v1** event object). • Use **reportingController** instead of the deprecated **source.component** field (this field has been renamed **deprecatedSource.component** and is not allowed to appear in the new

Resource	Deprecated API Version	Substitute API Version	Change Description
			events.k8s.io/v1 event object). • Use **reportingInstance** instead of the deprecated **source.host** field (this field has been renamed **deprecatedSource.host** and is not allowed to appear in the new **events.k8s.io/v1** event object).
HorizontalPodAutoscaler	autoscaling/v2beta1	autoscaling/v2 (This API has been available since v1.23.)	None
PodDisruptionBudget	policy/v1beta1	policy/v1 (This API has been available since v1.21.)	If **spec.selector** is set to null (**{}**) in **PodDisruptionBudget** of **policy/v1**, all pods in the namespace are selected. (In **policy/v1beta1**, an empty **spec.selector** means that no pod will be selected.) If **spec.selector** is not specified, pod will be selected in neither API version.
PodSecurityPolicy	policy/v1beta1	None	Since v1.25, the PodSecurityPolicy resource no longer provides APIs of the **policy/v1beta1** version, and the PodSecurityPolicy access controller is deleted. Use **Pod Security Admission** instead.
RuntimeClass	node.k8s.io/v1beta1	node.k8s.io/v1 (This API has been available since v1.20.)	None

Table 2-28 APIs deprecated in Kubernetes v1.22

Resource	Deprecated API Version	Substitute API Version	Change Description
MutatingWebhookConfiguration ValidatingWebhookConfiguration	admissionregistration.k8s.io/v1beta1	admissionregistration.k8s.io/v1 (This API has been available since v1.16.)	• The default value of **webhooks[*].failurePolicy** is changed from **Ignore** to **Fail** in v1. • The default value of **webhooks[*].matchPolicy** is changed from **Exact** to **Equivalent** in v1. • The default value of **webhooks[*].timeoutSeconds** is changed from **30s** to **10s** in v1. • The default value of **webhooks[*].sideEffects** is deleted, and this field must be specified. In v1, the value can only be **None** or **NoneOnDryRun**. • The default value of **webhooks[*].admissionReviewVersions** is deleted. In v1, this field must be specified. (**AdmissionReview** v1 and v1beta1 are supported.) • **webhooks[*].name** must be unique in the list of objects created through **admissionregistration.k8s.io/v1**.

Resource	Deprecated API Version	Substitute API Version	Change Description
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	apiextensions/v1 (This API has been available since v1.16.)	• The default value of **spec.scope** is no longer **Namespaced**. This field must be explicitly specified. • **spec.version** is deleted from v1. Use **spec.versions** instead. • **spec.validation** is deleted from v1. Use **spec.versions[*].schema** instead. • **spec.subresources** is deleted from v1. Use **spec.versions[*].subresources** instead. • **spec.additionalPrinterColumns** is deleted from v1. Use **spec.versions[*].additionalPrinterColumns** instead. • **spec.conversion.webhookClientConfig** is moved to **spec.conversion.webhook.clientConfig** in v1. • **spec.conversion.conversionReviewVersions** is moved to **spec.conversion.webhook.conversionReviewVersions** in v1. • **spec.versions[*].schema.openAPIV3Schema** becomes a mandatory field when the **CustomResourceDefinition** object of the v1 version is created, and its value must be a **structural schema**. • **spec.preserveUnknownFields: true** cannot be specified when the **CustomResourceDefinition** object of the v1 version is created. This configuration must be specified using **x-kubernetes-preserve-**

Resource	Deprecated API Version	Substitute API Version	Change Description
			unknown-fields: true in the schema definition. • In v1, the **JSONPath** field in the **additionalPrinterColumns** entry is renamed **jsonPath** (patch **#66531**).
APIService	apiregistration/v1beta1	apiregistration.k8s.io/v1 (This API has been available since v1.10.)	None
TokenReview	authentication.k8s.io/v1beta1	authentication.k8s.io/v1 (This API has been available since v1.6.)	None
LocalSubjectAccessReview SelfSubjectAccessReview SubjectAccessReview SelfSubjectRulesReview	authorization.k8s.io/v1beta1	authorization.k8s.io/v1 (This API has been available since v1.16.)	**spec.group** was renamed **spec.groups** in v1 (patch **#32709**).

Resource	Deprecated API Version	Substitute API Version	Change Description
CertificateSigningRequest	certificates.k8s.io/v1beta1	certificates.k8s.io/v1 (This API has been available since v1.19.)	Pay attention to the following changes in **certificates.k8s.io/v1**: • For an API client that requests a certificate: – **spec.signerName** becomes a mandatory field (see **Known Kubernetes Signers**). In addition, the **certificates.k8s.io/v1** API cannot be used to create requests whose signer is **kubernetes.io/legacy-unknown**. – **spec.usages** now becomes a mandatory field, which cannot contain duplicate string values and can contain only known usage strings. • For an API client that needs to approve or sign a certificate: – **status.conditions** cannot contain duplicate types. – The **status.conditions[*].status** field is now mandatory. – The **status.certificate** must be PEM-encoded and can contain only the **CERTIFICATE** data block.
Lease	coordination.k8s.io/v1beta1	coordination.k8s.io/v1 (This API has been available since v1.14.)	None

Resource	Deprecated API Version	Substitute API Version	Change Description
Ingress	networking.k8s.io/v1beta1 extensions/v1beta1	networking.k8s.io/v1 (This API has been available since v1.19.)	• The **spec.backend** field is renamed **spec.defaultBackend**. • The **serviceName** field of the backend is renamed **service.name**. • The backend **servicePort** field represented by a number is renamed **service.port.number**. • The backend **servicePort** field represented by a string is renamed **service.port.name**. • The **pathType** field is mandatory for all paths to be specified. The options are **Prefix**, **Exact**, and **ImplementationSpecific**. To match the behavior of not defining the path type in v1beta1, use **ImplementationSpecific**.
IngressClass	networking.k8s.io/v1beta1	networking.k8s.io/v1 (This API has been available since v1.19.)	None
ClusterRole ClusterRoleBinding Role RoleBinding	rbac.authorization.k8s.io/v1beta1	rbac.authorization.k8s.io/v1 (This API has been available since v1.8.)	None
PriorityClass	scheduling.k8s.io/v1beta1	scheduling.k8s.io/v1 (This API has been available since v1.14.)	None

Resource	Deprecated API Version	Substitute API Version	Change Description
CSIDriver CSINode StorageClass VolumeAttachment	storage.k8s.io/v1beta1	storage.k8s.io/v1	• CSIDriver is available in **storage.k8s.io/v1** since v1.19. • CSINode is available in **storage.k8s.io/v1** since v1.17. • StorageClass is available in **storage.k8s.io/v1** since v1.6. • VolumeAttachment is available in **storage.k8s.io/v1** since v1.13.

Table 2-29 APIs deprecated in Kubernetes v1.16

Resource	Deprecated API Version	Substitute API Version	Change Description
NetworkPolicy	extensions/v1beta1	networking.k8s.io/v1 (This API has been available since v1.8.)	None
DaemonSet	extensions/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	• The **spec.templateGeneration** field is deleted. • **spec.selector** is now a mandatory field and cannot be changed after the object is created. The label of an existing template can be used as a selector for seamless migration. • The default value of **spec.updateStrategy.type** is changed to **RollingUpdate** (the default value in the **extensions/v1beta1** API version is **OnDelete**).

Resource	Deprecated API Version	Substitute API Version	Change Description
Deployment	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	• The **spec.rollbackTo** field is deleted. • **spec.selector** is now a mandatory field and cannot be changed after the Deployment is created. The label of an existing template can be used as a selector for seamless migration. • The default value of **spec.progressDeadlineSeconds** is changed to 600 seconds (the default value in **extensions/v1beta1** is unlimited). • The default value of **spec.revisionHistoryLimit** is changed to **10**. (In the **apps/v1beta1** API version, the default value of this field is **2**. In the **extensions/v1beta1** API version, all historical records are retained by default.) • The default values of **maxSurge** and **maxUnavailable** are changed to **25%**. (In the **extensions/v1beta1** API version, these fields default to **1**.)
StatefulSet	apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	• **spec.selector** is now a mandatory field and cannot be changed after the StatefulSet is created. The label of an existing template can be used as a selector for seamless migration. • The default value of **spec.updateStrategy.type** is changed to **RollingUpdate** (the default value in the **apps/v1beta1** API version is **OnDelete**).

Resource	Deprecated API Version	Substitute API Version	Change Description
ReplicaSet	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API has been available since v1.9.)	**spec.selector** is now a mandatory field and cannot be changed after the object is created. The label of an existing template can be used as a selector for seamless migration.
PodSecurityPolicy	extensions/v1beta1	policy/v1beta1 (This API has been available since v1.10.)	PodSecurityPolicy for the **policy/v1beta1** API version will be removed in v1.25.

Upgrade Backup

The following table lists how to back up cluster data.

Backup Type	Backup Object	Backup Method	Backup Duration	Rollback Duration	Description
etcd data backup	etcd data	Automatic backup during an upgrade	1-5 minutes	2 hours	Mandatory. The data is automatically backed up during an upgrade.
CBR cloud server backup	Master node disks, including component images, configurations, logs, and etcd data	One-click backup on a web page (manually triggered)	20 minutes to 2 hours (based on the cloud backup tasks in the current region)	20 minutes	This function is gradually replaced by EVS snapshot backup.

Backup Type	Backup Object	Backup Method	Backup Duration	Rollback Duration	Description
EVS snapshot backup	Master node disks, including component images, configurations, logs, and etcd data	One-click backup on a web page (manually triggered)	1-5 minutes	20 minutes	This function is coming soon. After this function is released, it will replace CBR cloud server backup.

2.6.3 Performing Post-Upgrade Verification

2.6.3.1 Cluster Status Check

Check Items

After a cluster is upgraded, check whether the cluster is in the **Running** state.

Procedure

CCE automatically checks your cluster status. Go to the cluster list page and confirm the cluster status based on the diagnosis result.

Solution

If your cluster malfunctions, contact technical support.

2.6.3.2 Node Status Check

Check Items

After a cluster is upgraded, check whether nodes in the cluster are in the **Running** state.

Procedure

CCE automatically checks your node statuses. Go to the node list page and confirm the node statuses based on the diagnosis result.

Solution

If a node malfunctions, **reset the node**. If the fault persists, contact technical support.

2.6.3.3 Node Skipping Check

Check Items

After a cluster is upgraded, check whether there are any nodes that skip the upgrade in the cluster. These nodes may affect the proper running of the cluster.

Procedure

CCE automatically checks whether there are nodes that skip the upgrade in the cluster. Go to the node list page and confirm the nodes based on the diagnosis result. The skipped nodes are labeled with **upgrade.cce.io/skipped=true**.

Solution

The skipped nodes are displayed on the upgrade details page. Reset the skipped nodes after the upgrade is complete. For details about how to reset a node, see **Resetting a Node**.

NOTE

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

2.6.3.4 Service Check

Check Items

After a cluster is upgraded, check whether its services are running properly.

Procedure

Different services have different verification mode. Select a suitable one and verify the service before and after the upgrade.

You can verify the service from the following aspects:

- The service page is available.
- No alarm or event is generated on the normal platform.
- No error log is generated for key processes.
- The API dialing test is normal.

Solution

If your online services malfunction after the cluster upgrade, contact technical support.

2.6.3.5 New Node Check

Check Items

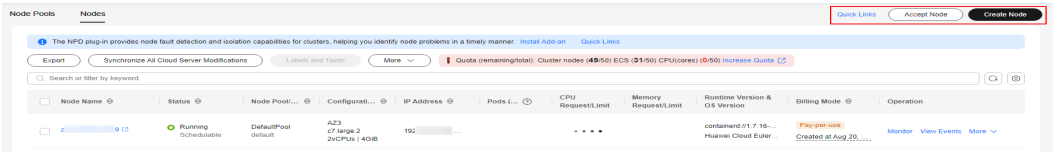
Check whether nodes can be created in the cluster.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab and then **Create Node**. For details about the node configuration, see **Creating a Node**.

Figure 2-38 Creating a node



----End

Solution

If nodes cannot be created in your cluster after the cluster is upgraded, contact technical support.

2.6.3.6 New Pod Check

Check Items

- Check whether pods can be created on the existing nodes after the cluster is upgraded.
- Check whether pods can be created on new nodes after the cluster is upgraded.

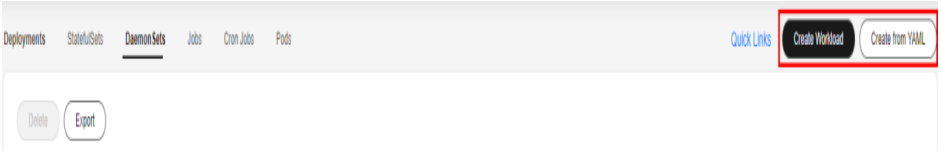
Procedure

After creating a node based on **New Node Check**, create a DaemonSet workload to create pods on each node.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. On the displayed page, click **Create Workload** or **Create from YAML** in the upper right corner. For details about how to create a DaemonSet, see **Creating a DaemonSet**.

Figure 2-39 Creating a DaemonSet



It is a good practice to use the image for routine tests as the base image. You can deploy minimum pods for an application by referring to the following YAML file.

NOTE

In this test, a DaemonSet is deployed in the default namespace using YAML. It uses the **nginx:perl** base image and specifies a request of 10m vCPUs and 10 MiB of memory. The resource limits for this DaemonSet are set to 100m vCPUs and 50 MiB of memory.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: post-upgrade-check
  namespace: default
spec:
  selector:
    matchLabels:
      app: post-upgrade-check
      version: v1
  template:
    metadata:
      labels:
        app: post-upgrade-check
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:perl
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 10m
              memory: 10Mi
            limits:
              cpu: 100m
              memory: 50Mi
```

Step 3 After the workload is created, check whether the pods of the workload are running properly.

Step 4 After the check is complete, choose **Workloads** in the navigation pane. On the displayed page, click the **DaemonSets** tab, locate the **post-upgrade-check** workload, and choose **More** > **Delete** in the **Operation** column to delete the test workload.

----End

Solution

If the pod cannot be created or the pod status is abnormal, contact technical support and specify whether the exception occurs on new nodes or existing nodes.

2.6.4 Migrating Services Across Clusters of Different Versions

Application Scenarios

This section describes how to migrate services from a cluster of an earlier version to a cluster of a later version in CCE.

This operation is applicable when a cross-version cluster upgrade is required (for example, upgrade from v1.19.* to v1.28.*) and new clusters can be created for service migration.

Prerequisites

Table 2-30 Checklist before migration

Category	Description
Cluster	NodeIP-related: Check whether node IP addresses (including EIPs) of the cluster before the migration have been used in other configurations or trustlists.
Workloads	Record the number of workloads for post-migration check.
Storage	1. Check whether the storage resources in use are provisioned by the cloud or by your organization. 2. Change the automatically created storage to the existing storage in the new cluster.
Network	1. Pay special attention to the ELB and ingress. 2. Clusters of an earlier version support only the classic load balancer. To migrate services to a new cluster, change load balancer type to shared load balancer. Then, the corresponding ELB service will be re-established.
O&M	Private configuration: Check whether kernel parameters or system data have been configured on nodes in the cluster.

Procedure

Step 1 Create a CCE cluster.

Create a cluster with the same specifications and configurations as the cluster of the earlier version. For details, see **Buying a CCE Standard/Turbo Cluster**.

Step 2 Add a node.

Add a node with the same specifications and manual configuration items. For details, see **Creating a Node**.

Step 3 Create a storage volume in the new cluster.

Use the existing storage to create a PVC in the new cluster. The PVC name remains unchanged. For details, see **Using an Existing OBS Bucket Through a Static PV** or **Using an Existing SFS Turbo File System Through a Static PV**.

NOTE

Storage switching supports only shared storage such as OBS and SFS Turbo. If non-shared storage is used, suspend the workloads in the old cluster to switch the storage resources. In this case, services will be unavailable.

Step 4 Create a workload in the new cluster.

Create a workload in the new cluster. The name and specifications remain unchanged. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

Step 5 Mount the storage again.

Remount the existing storage in the workload. For details, see **Using an Existing OBS Bucket Through a Static PV** or **Using an Existing SFS Turbo File System Through a Static PV**.

Step 6 Create a Service in the new cluster.

The Service name and specifications remain unchanged. For details about how to create a Service, see **Service**.

Step 7 Commission services.

After all resources are created, commission the containerized services. If the commissioning is successful, migrate the services to the new cluster.

Step 8 Delete or unsubscribe from the old cluster.

When all functions of the new cluster are stable, unsubscribe from or delete the old cluster. For details about how to delete a cluster, see **Deleting a Cluster**.

----End

2.6.5 Troubleshooting for Pre-upgrade Check Exceptions

2.6.5.1 Pre-upgrade Check

The system automatically checks a cluster before its upgrade. If the cluster does not meet the pre-upgrade check conditions, the upgrade cannot continue. To avoid risks, you can perform pre-upgrade check according to the check items and solutions described in this section.

Table 2-31 Check items

No.	Check Item	Description
1	**Node Restrictions**	• Check whether the node is available. • Check whether the node OS supports the upgrade. • Check whether the node is marked with unexpected node pool labels. • Check whether the Kubernetes node name is the same as the ECS name.
2	**Upgrade Management**	Check whether the target cluster is under upgrade management.
3	**Add-ons**	• Check whether the add-on status is normal. • Check whether the add-on supports the target version.

No.	Check Item	Description
4	**Helm Charts**	Check whether the current HelmRelease record contains discarded Kubernetes APIs that are not supported by the target cluster version. If yes, the Helm chart may be unavailable after the upgrade.
5	**SSH Connectivity of Master Nodes**	Check whether your master nodes can be accessed using SSH.
6	**Node Pools**	Check the node pool status.
7	**Security Groups**	Check whether the **Protocol & Port** of the worker node security groups is set to **ICMP: All** and whether the security group with the source IP address set to the master node security group is deleted.
8	**Residual Nodes**	Check whether nodes need to be migrated.
9	**Discarded Kubernetes Resources**	Check whether there are discarded resources in the clusters.
10	**Compatibility Risks**	Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.
11	**CCE Agent Versions**	Check whether cce-agent on the current node is of the latest version.
12	**Node CPU Usage**	Check whether the node's CPU usage is above 90%.
13	**CRDs**	• Check whether the key CRD **packageversions.version.cce.io** of the cluster is deleted. • Check whether the cluster key CRD **network-attachment-definitions.k8s.cni.cncf.io** is deleted.
14	**Node Disks**	• Check whether the key data disks on the node meet the upgrade requirements. • Check whether the **/tmp** directory has 500 MB available space.
15	**Node DNS**	• Check whether the DNS configuration of the current node can resolve the OBS address. • Check whether the current node can access the OBS address of the storage upgrade component package.
16	**Node Key Directory File Permissions**	Check whether the owner and owner group of the files in the **/var/paas** directory used by the CCE are both **paas**.

No.	Check Item	Description
17	**kubelet**	Check whether the kubelet on the node is running properly.
18	**Node Memory**	Check whether the node's memory usage is above 90%.
19	**Node Clock Synchronization Server**	Check whether the clock synchronization server ntpd or chronyd of the node is running properly.
20	**Node OS**	Check whether the OS kernel version of the node is supported by CCE.
21	**Node CPU Cores**	Check and make sure that the master nodes in your cluster have more than 2 CPU cores.
22	**Node Python Commands**	Check whether the Python commands are available on a node.
23	**ASM Version**	• Check whether ASM is used by the cluster. • Check whether the current ASM version supports the target cluster version.
24	**Node Readiness**	Check whether the nodes in the cluster are ready.
25	**Node journald**	Check whether journald of a node is normal.
26	**containerd.sock**	Check whether the **containerd.sock** file is on the node. This file affects the startup of container runtime in the Euler OS.
27	**Internal Error**	This check item is not typical and implies that an internal error was found during the pre-upgrade check.
28	**Node Mount Points**	Check whether there are inaccessible mount points on the node.
29	**Kubernetes Node Taints**	Check whether the taint needed for cluster upgrade exists on the node.
30	**Everest Restrictions**	Check whether there are any compatibility restrictions on the current Everest add-on.
31	**cce-hpa-controller Limitations**	Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.
32	**Enhanced CPU Policies**	Check whether the current cluster version and the target version support **enhanced CPU policy**.
33	**Health of Worker Node Components**	Check whether the container runtime and network components on the worker nodes are healthy.

No.	Check Item	Description
34	**Health of Master Node Components**	Check whether cluster components such as the Kubernetes component, container runtime component, and network component are running properly before the upgrade.
35	**Memory Resource Limit of Kubernetes Components**	Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.
36	**Discarded Kubernetes APIs**	The system scans the audit logs of the past day to check whether the user calls the deprecated APIs of the target Kubernetes version. **NOTE** Due to the limited time range of audit logs, this check item is only an auxiliary method. APIs to be deprecated may have been used in the cluster, but their usage is not included in the audit logs of the past day. Check the API usage carefully.
37	**NetworkManager**	Check whether NetworkManager of a node is normal.
38	**Node ID File**	Check the ID file format.
39	**Node Configuration Consistency**	When you upgrade a cluster to v1.19 or later, the system checks whether the following configuration files have been modified on the backend:
40	**Node Configuration File**	Check whether the configuration files of key components exist on the node.
41	**CoreDNS Configuration Consistency**	Check whether the current CoreDNS key configuration Corefile is different from the Helm release record. The difference may be overwritten during the add-on upgrade, **affecting domain name resolution in the cluster**.
42	**sudo**	Check whether the sudo commands and sudo-related files of the node are working.
43	**Key Node Commands**	Whether some key commands that the node upgrade depends on are working
44	**Mounting of a Sock File on a Node**	Check whether the **docker/containerd.sock** file is directly mounted to the pods on a node. During an upgrade, Docker or containerd restarts and the sock file on the host changes, but the sock file mounted to pods does not change accordingly. As a result, your services cannot access Docker or containerd due to sock file inconsistency. After the pods are rebuilt, the sock file is mounted to the pods again, and the issue is resolved accordingly.

No.	Check Item	Description
45	**HTTPS Load Balancer Certificate Consistency**	Check whether the certificate used by an HTTPS load balancer has been modified on ELB.
46	**Node Mounting**	Check whether the default mount directory and soft link on the node have been manually mounted or modified.
47	**Login Permissions of User paas on a Node**	Check whether user **paas** is allowed to log in to a node.
48	**Private IPv4 Addresses of Load Balancers**	Check whether the load balancer associated with a Service is allocated with a private IPv4 address.
49	**Historical Upgrade Records**	Check the historical upgrade records of the cluster and confirm that the current version of the cluster meets the requirements for upgrading to the target version.
50	**CIDR Block of the Cluster Management Plane**	Check whether the CIDR block of the cluster management plane is the same as that configured on the backbone network.
51	**GPU Add-on**	The GPU add-on is involved in the upgrade, which may affect the GPU driver installation during the creation of a GPU node.
52	**Nodes' System Parameters**	Check whether the default system parameter settings on your nodes are modified.
53	**Residual Package Version Data**	Check whether there are residual package version data in the current cluster.
54	**Node Commands**	Check whether the commands required for the upgrade are available on the node.
55	**Node Swap**	Check whether swap has been enabled on cluster nodes.
56	**NGINX Ingress Controller**	Check whether there are compatibility issues that may occur during NGINX Ingress Controller upgrade.
57	**ELB Listener Access Control**	If so, check whether their configurations are correct.
58	**Master Node Flavor**	Check whether the flavor of the master nodes in the cluster is the same as the actual flavor of these nodes.

No.	Check Item	Description
59	**Subnet Quota of Master Nodes**	Check whether the number of available IP addresses in the cluster subnet supports rolling upgrade.
60	**Node Runtime**	Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.
61	**Node Pool Runtime**	Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.
62	**Number of Node Images**	Check the number of images on your node. If there are more than 1000 images, it takes a long time for Docker to start, affecting the standard Docker output and functions such as Nginx.
63	**OpenKruise Compatibility Check**	Check whether the OpenKruise add-on is compatible before upgrading a cluster.
64	**Compatibility Check of Secret Encryption**	Check whether the target version supports secret encryption. If it does not, clusters that have this feature enabled cannot be upgraded to the target version.
65	**Compatibility Between the Ubuntu Kernel and GPU Driver**	Make sure that the GPU add-on and Ubuntu nodes are compatible before using them in a cluster. If the Ubuntu kernel is 5.15.0-113-generic, the driver of the GPU add-on must be 535.161.08 or later.
66	**Drainage Tasks**	An unfinished drainage task is detected in the cluster, which may resume after the upgrade. If this happens, running pods will be evicted, which could impact your services.
67	**Image Layers on a Node**	Check the number of image layers on your node. If there are more than 5000 layers, it will take a long time for Docker or containerd to start, affecting the stdout of Docker or containerd.
68	**Cluster Rolling Upgrade**	Check whether your cluster is eligible for a rolling upgrade. The result shows that the rolling upgrade is not supported.

No.	Check Item	Description
69	Rotation Certificates	Check whether the number of certificates on your node is greater than 1000. During an upgrade, certificate files will be processed in batches. An excessive number of certificate files will lead to a slow node upgrade and result in pod eviction from the node.
70	Network Policies of Cluster Network Components	Check the network policy settings on the master nodes in your cluster. If any manual modifications have been made, they will be reset during the upgrade.
71	Cluster and Node Pool Configurations	Check whether the **nic-max-above-warm-target** value configured for the network component of the current cluster exceeds the maximum value allowed.
72	Time Zone of Master Nodes	Check whether the time zone of the master nodes matches the cluster's time zone. If they are different, the master nodes will be updated to match the cluster's time zone during a rolling upgrade.

2.6.5.2 Node Restrictions

Check Items

Check the following items:

- Check whether the node is available.
- Check whether the node OS supports the upgrade.
- Check whether the node is marked with unexpected node pool labels.
- Check whether the Kubernetes node name is the same as the ECS name.

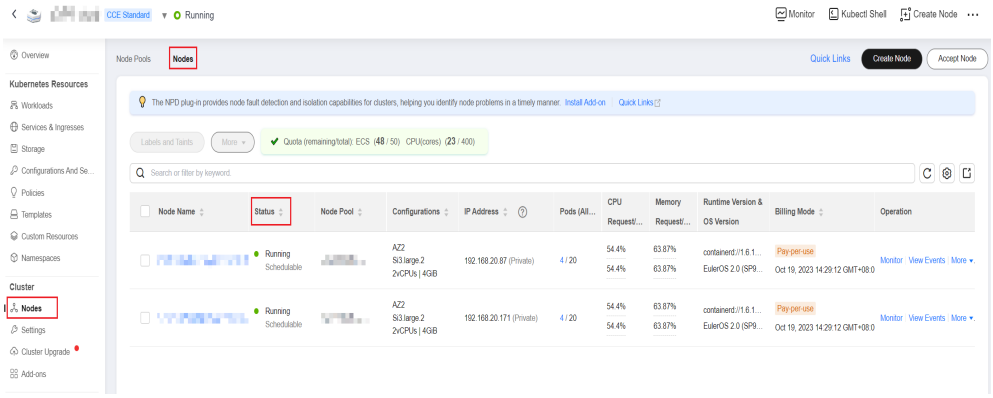
Solution

1. **The node is unavailable. Preferentially recover the node.**

   If a node is unavailable, log in to the CCE console and click the cluster name to access the cluster console. Then, choose **Nodes** in the navigation pane. In the right pane, click the **Nodes** tab. Ensure that the node is in the **Running** state. A node in the **Installing** or **Deleting** state cannot be upgraded.

   If a node is unavailable, recover the node and retry the check task. For details, see **What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?**

Figure 2-40 Checking node statuses



2. **The node OS does not support the upgrade.**

   The following table lists the node OSs that support the upgrade. You can reset the node OS to an available OS in the list.

   Table 2-32 OSs that support the upgrade

OS	Constraint
EulerOS 2.x	If the target version is earlier than v1.27, there are no constraints. If the target version is v1.27 or later, only EulerOS 2.9 and EulerOS 2.10 support the upgrade.
CentOS 7.x	None
Ubuntu	If the target version is v1.27 or later, only Ubuntu 22.04 supports the upgrade.
Huawei Cloud EulerOS	None

3. **The affected node belongs to the default node pool but it is configured with a non-default node pool label, which will affect the upgrade.**

   If a node is migrated from a common node pool to the default node pool, the **cce.cloud.com/cce-nodepool** label will affect the cluster upgrade. Check whether load scheduling on the node depends on the label.

   - If no, delete the label.
   - If yes, modify the load balancing policy, remove the dependency, and then delete the label.

4. **The node is marked with a CNIProblem taint. Preferentially recover the node.**

   The node contains a taint whose key is **node.cloudprovider.kubernetes.io/cni-problem**, and the effect is **NoSchedule**. The taint is added by the NPD add-on. Upgrade the NPD add-on to the latest version and check again. If the problem persists, contact technical support.

5. **The resources for the affected node in Kubernetes are not available. It is possible that the node is being deleted. Try again later.**

Recheck after the node is deleted.

6. **The OS running on the master node is EulerOS 2.5, which does not support the cluster to be upgraded to v1.27.5-r0.**

   You can upgrade the version to 1.25 or 1.28. If you choose to upgrade the version to 1.28, EulerOS 2.5 will be upgraded to HCE 2.0 during the upgrade. If you have other requirements, submit a service ticket.

2.6.5.3 Upgrade Management

Check Items

Check whether the target cluster is under upgrade management.

Solution

CCE may temporarily restrict the cluster upgrade due to the following reasons:

- The cluster is identified as the core production cluster.
- Other O&M tasks are being or will be performed, for example, 3-AZ reconstruction on master nodes.
- If the cluster contains Docker nodes but their OSs are different from that of the node pool, reset these nodes and run the pre-upgrade check again.

To resolve this issue, contact technical support based on logs displayed on the console.

2.6.5.4 Add-ons

Check Items

Check the following items:

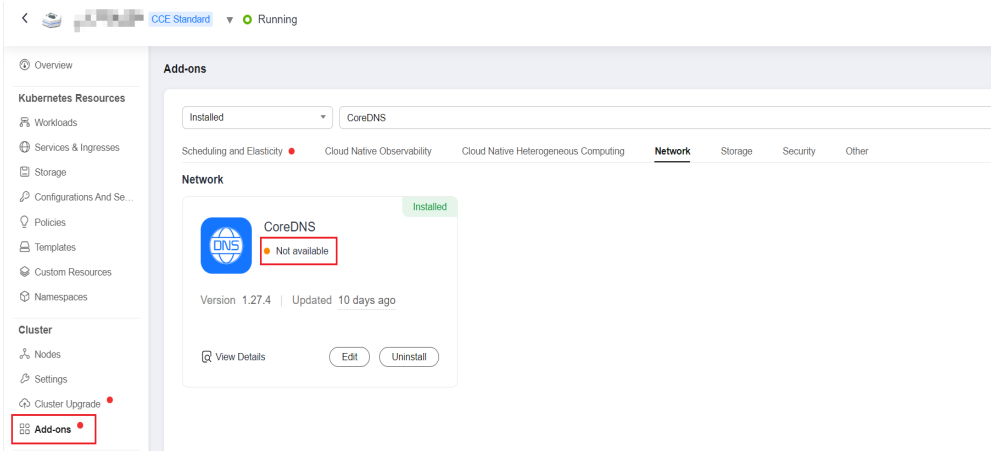
- Check whether the add-on status is normal.
- Check whether the add-on supports the target version.

Solution

- **Scenario 1: The add-on malfunctions.**

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and obtain add-ons. Then, handle malfunctional add-ons.

Figure 2-41 Checking add-on statuses



- **Scenario 2: The target version of the cluster upgrade does not support the add-on.**

  The following error information is displayed during the pre-upgrade check:

  ```
  addon [***] does not support cluster target version, check and try again
  ```

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and manually uninstall the add-on. For details about the supported add-on versions and substitutions, see the **Help** document.

- **Scenario 3: The add-on configuration does not meet the upgrade requirements. Upgrade the add-on and try again.**

  The following error information is displayed during the pre-upgrade check:

  ```
  please upgrade addon [ ] in the page of addon managecheck and try again
  ```

  In this case, log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane and manually upgrade the add-on.

2.6.5.5 Helm Charts

Check Items

Check whether the current HelmRelease record contains discarded Kubernetes APIs that are not supported by the target cluster version. If yes, the Helm chart may be unavailable after the upgrade.

Solution

Convert the discarded Kubernetes APIs to APIs that are compatible with both the source and target versions.

NOTE

This item has been automatically processed in the upgrade process. You can ignore this item.

2.6.5.6 SSH Connectivity of Master Nodes

Check Items

Check whether your master nodes can be accessed using SSH.

Solution

There is a low probability that the SSH connectivity check fails due to network fluctuations. Perform the pre-upgrade check again.

If the check still fails, submit a service ticket to contact technical support.

2.6.5.7 Node Pools

Check Items

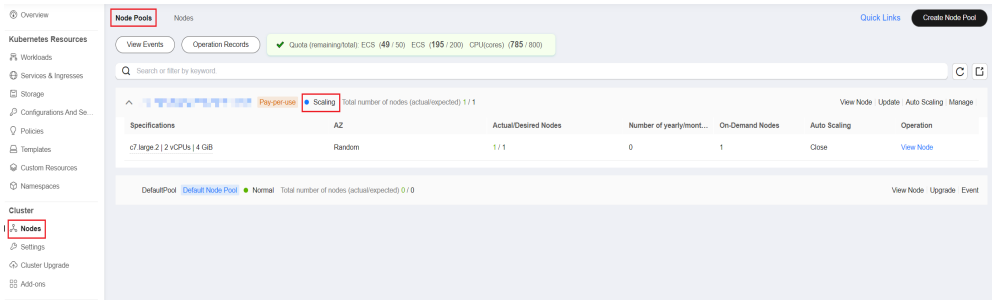
- Check the node pool status.
- Check whether the node pool OS or container runtime is supported after the upgrade.

Solution

- **Scenario: The node pool malfunctions.**

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane and check the status of the affected node pool on the **Node Pools** tab. If the node pool is being scaled, wait until the node pool scaling is complete.

  Figure 2-42 Checking node pool statuses

- **Scenario: The node pool OS is not supported.**

  The runtime and OS vary depending on the cluster version. This issue typically occurs when a cluster of an earlier version is upgraded to v1.27 or later. For details about the mapping between CCE cluster versions and OSs, see **Node OSs**.

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane, view the status of the affected node pool on the **Node Pools** tab, and click **Upgrade**. Change the supported OSs based on the pre-upgrade check result, and click **OK**.

  If there are nodes in the affected node pool, choose **More** > **Synchronize** in the operation column to synchronize the OS of the existing nodes. For details, see **Synchronizing Node Pools**.

2.6.5.8 Security Groups

Check Items

Check whether the **Protocol & Port** of the worker node security groups is set to **ICMP: All** and whether the security group with the source IP address set to the master node security group is deleted.

NOTE

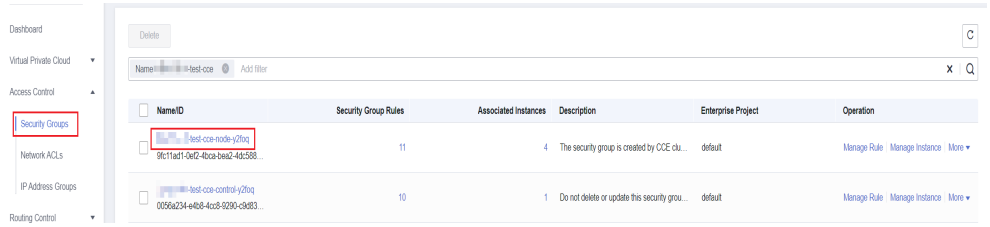
This check item is performed only for clusters using VPC networking. For clusters using other networking, skip this check item.

Solution

Log in to the VPC console, choose **Access Control** > **Security Groups**, and enter the target cluster name in the search box. Two security groups are expected to display:

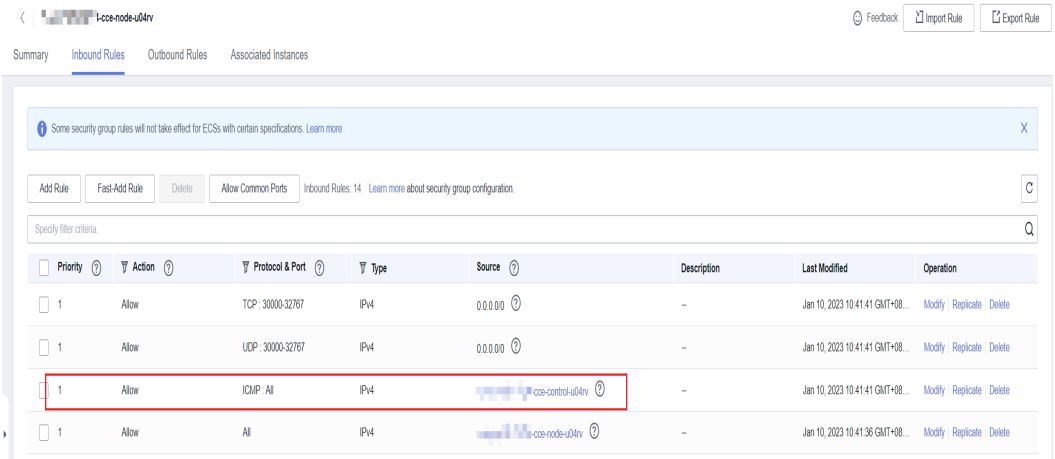
- The security group name is **cluster name-node-*xxx***. This security group is associated with the worker nodes.
- The security group name is **cluster name-control-*xxx***. This security group is associated with the master nodes.

Figure 2-43 Cluster security groups



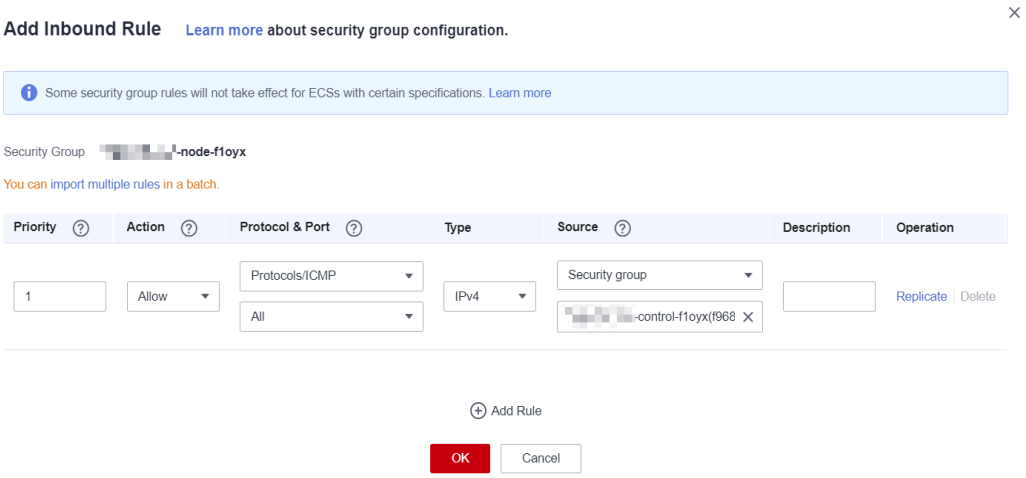
Click the node security group and ensure that the following rules are configured to allow the master node to access the node using **ICMP**.

Figure 2-44 Node security group rules



If the preceding security group rule is unavailable, add the rule with the following configurations to the node security group: Set **Protocol & Port** to **Protocols/ICMP** and **All**, and **Source** to **Security group** and the master security group. Describe the rule as "Created by CCE,please don't modify! Used by the master node to access the worker node."

Figure 2-45 Allowing ICMP for the master security group



2.6.5.9 Residual Nodes

Check Items

Check whether nodes need to be migrated.

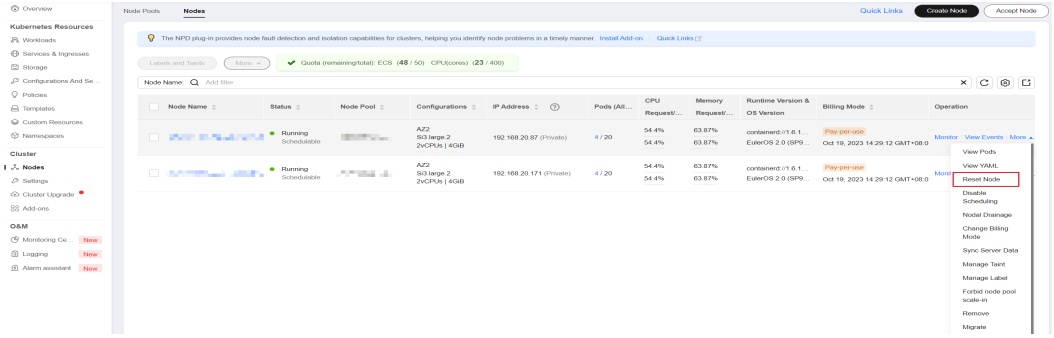
Solution

This issue is caused by either an error in the node's package pull component or the absence of key system components on the node, which could be due to an upgrade from an earlier version.

Solution 1

Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. Locate the row containing the target node and choose **More** > **Reset Node** in the **Operation** column. For details, see **Resetting a Node**. After the node is reset, retry the check task.

Figure 2-46 Resetting a node



NOTE

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

Solution 2

After creating a node, delete the faulty node.

2.6.5.10 Discarded Kubernetes Resources

Check Items

Check whether there are discarded resources in the clusters.

Solution

- **Scenario 1: The Service in the clusters of v1.25 or later has discarded annotation tolerate-unready-endpoints.**

  Error log:

  ```
  some check failed in cluster upgrade: this cluster has deprecated service list: map[***] with deprecated annotation list [tolerate-unready-endpoints]
  ```

  Check whether the Service provided in the log information contains the annotation **tolerate-unready-endpoint**. If so, replace the annotation with the following fields in Service **spec**:

  ```
  publishNotReadyAddresses: true
  ```

- **Scenario 2: The Service in the clusters of v1.27 or later has discarded annotation service.kubernetes.io/topology-aware-hints.**

  Error log:

  ```
  some check failed in cluster upgrade: this cluster has deprecated service list: map[***] with deprecated annotation list [service.kubernetes.io/topology-aware-hints]
  ```

  Check whether the Service provided in the log information contains the annotation **service.kubernetes.io/topology-aware-hints**. If so, replace it with the annotation **service.kubernetes.io/topology-mode** in the affected Service.

2.6.5.11 Compatibility Risks

Check Items

Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.

Version compatibility

Upgrade Path	Version Difference	Self-Check
v1.23 or v1.25 Upgraded to v1.27	Docker is no longer recommended. Use containerd instead. For details, see **Container Engines**.	This item has been included in the pre-upgrade check.
v1.21 or v1.19 Upgraded to v1.23	For the NGINX Ingress Controller of an earlier version (community version v0.49 or earlier, or CCE nginx-ingress version v1.x.x), the created ingresses can be managed by the NGINX Ingress Controller even if **kubernetes.io/ingress.class: nginx** is not set in the ingress **annotations**. However, for the NGINX Ingress Controller of a later version (community version v1.0.0 or later, or CCE nginx-ingress version v2.x.x), the ingresses created without specifying the Nginx type will not be managed by the NGINX Ingress Controller, and ingress rules will become invalid, which interrupts services.	This item has been included in the pre-upgrade check. You can also perform the self-check by referring to **NGINX Ingress Controller**.
v1.19 to v1.21	The bug of **exec probe timeouts** is fixed in Kubernetes 1.21. Before this bug is fixed, the exec probe does not consider the **timeoutSeconds** field. Instead, the probe will run indefinitely, even beyond its configured deadline. It will stop until the result is returned. If this field is not specified, the default value **1** is used. This field takes effect after the upgrade. If the probe runs over 1 second, the application health check may fail and the application may restart frequently.	Before the upgrade, check whether the timeout is properly set for the exec probe.

Upgrade Path	Version Difference	Self-Check
	kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly. Root cause: X.509 **CommonName** is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. If your webhook certificate does not have SANs, kube-apiserver does not process the **CommonName** field of the X.509 certificate as the host name by default. As a result, the authentication fails.	Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server. • If you do not have your own webhook server, you can skip this check. • If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate.
v1.15 to v1.19	The control plane of CCE clusters of v1.19 is incompatible with kubelet v1.15. If a node fails to be upgraded or the node to be upgraded restarts after the master node is successfully upgraded, there is a high probability that the node is in the **NotReady** status. This is because the node failed to be upgraded restarts the kubelet and trigger the node registration. In clusters of v1.15, the default registration tags (**failure-domain.beta.kubernetes.io/is-baremetal** and **kubernetes.io/availablezone**) are regarded as invalid tags by the clusters of v1.19. The valid tags in the clusters of v1.19 are **node.kubernetes.io/baremetal** and **failure-domain.beta.kubernetes.io/zone**.	1. In normal cases, this scenario is not triggered. 2. After the master node is upgraded, do not suspend the upgrade so the node can be quickly upgraded. 3. If a node fails to be upgraded and cannot be restored, evict applications on the node as soon as possible. Contact technical support and skip the node upgrade. After the upgrade is complete, reset the node.

Upgrade Path	Version Difference	Self-Check

	In CCE v1.15 and v1.19 clusters, the Docker storage driver file system has been changed from XFS to Ext4. As a result, the import package sequence may be abnormal in the pods of upgraded Java application, leading to pod exceptions.	Before the upgrade, check the Docker configuration file **/etc/docker/daemon.json** on the node. Check whether the value of **dm.fs** is **xfs**.
		• If the value is **ext4** or the storage driver is Overlay, you can skip the next steps.
		• If the value is **xfs**, you are advised to deploy applications in the cluster of the new version in advance to test whether the applications are compatible with the new cluster version.
		{
		"storage-driver": "devicemapper",
		"storage-opts": [
		"dm.thinpooldev=/dev/mapper/vgpaas-thinpool",
		"dm.use_deferred_removal=true",
		"dm.fs=xfs",
		"dm.use_deferred_deletion=true"
		]
		}

	kube-apiserver of CCE v1.19 or later requires that the Subject Alternative Names (SANs) field be configured for the certificate of your webhook server. Otherwise, kube-apiserver fails to call the webhook server after the upgrade, and containers cannot be started properly. Root cause: X.509 **CommonName** is discarded in Go v1.15. kube-apiserver of CCE v1.19 is compiled using Go v1.15. The **CommonName** field is processed as the host name. As a result, the authentication fails.	Before the upgrade, check whether the SAN field is configured in the certificate of your webhook server. • If you do not have your own webhook server, you can skip this check. • If the field is not set, use the SAN field to specify the IP address and domain name supported by the certificate. **NOTICE** To mitigate the impact of version differences on cluster upgrade, CCE performs special processing during the upgrade from v1.15 to v1.19 and still supports certificates without SANs. However, no special processing is required for subsequent upgrades. You are advised to rectify your certificate as soon as possible.

Upgrade Path	Version Difference	Self-Check
	In clusters of v1.17.17 and later, CCE automatically creates pod security policies (PSPs) for you, which restrict the creation of pods with unsafe configurations, for example, pods for which **net.core.somaxconn** under a sysctl is configured in the security context.	After an upgrade, you can allow insecure system configurations as required. For details, see **Configuring a Pod Security Policy**.
	If initContainer or Istio is used in the in-place upgrade of a cluster of v1.15, pay attention to the following restrictions: In kubelet v1.16 and later versions, **QoS classes** are different from those in earlier versions. In kubelet v1.15 and earlier versions, only containers in **spec.containers** are counted. In kubelet v1.16 and later versions, containers in both **spec.containers** and **spec.initContainers** are counted. The QoS class of a pod will change after the upgrade. As a result, the container in the pod restarts.	You are advised to modify the QoS class of the service container before the upgrade to avoid this problem. For details, see **Table 2-24**.
v1.13 to v1.15	After a VPC network cluster is upgraded, the master node occupies an extra CIDR block due to the upgrade of network components. If no container CIDR block is available for the new node, the pod scheduled to the node cannot run.	Generally, this problem occurs when the nodes in the cluster are about to fully occupy the container CIDR block. For example, the container CIDR block is 10.0.0.0/16, the number of available IP addresses is 65,536, and the VPC network allocates a CIDR block with the fixed size (using the mask to determine the maximum number of container IP addresses allocated to each node). If the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes, including the three master nodes. After the cluster is upgraded, each of the three master nodes occupies one CIDR block. As a result, 506 nodes are supported.

Table 2-33 QoS class changes before and after the upgrade

Init Container (Calculated Based on spec.initContainers)	Service Container (Calculated Based on spec.containers)	Pod (Calculated Based on spec.containers and spec.initContainers)	Impacted or Not
Guaranteed	Besteffort	Burstable	Yes
Guaranteed	Burstable	Burstable	No
Guaranteed	Guaranteed	Guaranteed	No
Besteffort	Besteffort	Besteffort	No
Besteffort	Burstable	Burstable	No
Besteffort	Guaranteed	Burstable	Yes
Burstable	Besteffort	Burstable	Yes
Burstable	Burstable	Burstable	No
Burstable	Guaranteed	Burstable	Yes

2.6.5.12 CCE Agent Versions

Check Items

Check whether cce-agent on the current node is of the latest version.

Solution

- **Scenario 1: The error message "you cce-agent no update, please restart it" is displayed.**

  This issue occurs if cce-agent is not restarted. To resolve it, log in to the node and manually restart cce-agent.

  Solution: Log in to the node and run the following command:

  ```
  systemctl restart cce-agent
  ```

  Perform the pre-upgrade check again.

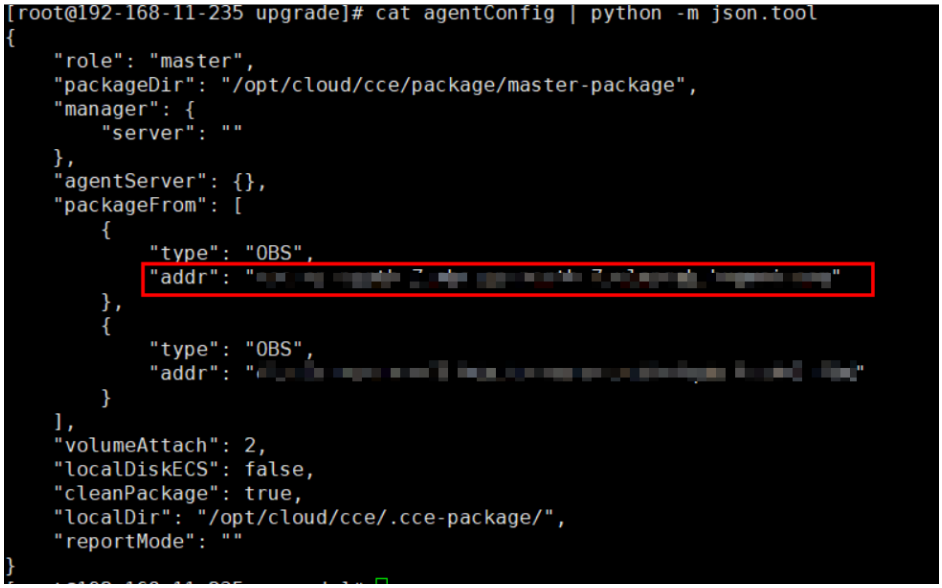
- **Scenario 2: The error message "your cce-agent is not the latest version" is displayed.**

  This issue occurs if cce-agent is not of the latest version and the automatic update failed. This issue is typically caused by an invalid OBS path or the component version is outdated.

  Solution

  a. Log in to the affected node and run the following command to obtain a valid OBS address:


```
cat /home/paas/upgrade/agentConfig | python -m json.tool
```



b. Log in to a where the check failed, obtain the OBS address again by referring to the previous step, and check whether the OBS addresses are the same. If they are different, change the OBS address of the abnormal node to the correct address.

c. Run the following commands to download the latest binary file:

- x86

  ```
  curl -k "https://{OBS address you have obtained}/cluster-versions/base/cce-agent" > /tmp/cce-agent
  ```

- Arm

  ```
  curl -k "https://{OBS address you have obtained}/cluster-versions/base/cce-agent-arm" > /tmp/cce-agent-arm
  ```

d. Replace the original cce-agent binary file.

- x86

  ```
  mv -f /tmp/cce-agent /usr/local/bin/cce-agent
  chmod 750 /usr/local/bin/cce-agent
  chown root:root /usr/local/bin/cce-agent
  ```

- Arm

  ```
  mv -f /tmp/cce-agent-arm /usr/local/bin/cce-agent-arm
  chmod 750 /usr/local/bin/cce-agent-arm
  chown root:root /usr/local/bin/cce-agent-arm
  ```

e. Restart cce-agent.

```
systemctl restart cce-agent
```

If you have any questions about the preceding operations, contact technical support.

2.6.5.13 Node CPU Usage

Check Items

Check whether the node's CPU usage is above 90%.

Solution

- **Upgrade the cluster during off-peak hours.**
- Check whether too many pods are deployed on the node. If yes, reschedule pods to other idle nodes.

2.6.5.14 CRDs

Check Items

Check the following items:

- Check whether the key CRD **packageversions.version.cce.io** of the cluster is deleted.
- Check whether the cluster key CRD **network-attachment-definitions.k8s.cni.cncf.io** is deleted.

Solution

If check results are abnormal, contact technical support.

2.6.5.15 Node Disks

Check Items

Check the following items:

- Check whether the key data disks on the node meet the upgrade requirements.
- Check whether the **/tmp** directory has 500 MB available space.

Solution

During the node upgrade, the key disks store the upgrade component package, and the **/tmp** directory stores temporary files.

- **Scenario 1: Master node disks fail to meet the upgrade requirements.**

  Contact technical support.

- **Scenario 2: Worker node disks fail to meet the upgrade requirements.**

  Check the usage of each key disk. After ensuring that the available space meets the requirements, check again.

  - Disk partition of Docker: at least 1 GB of available space
    ```
    df -h /var/lib/docker
    ```
  - Disk partition of containerd: at least 1 GB of available space
    ```
    df -h /var/lib/containerd
    ```
  - Disk partition of kubelet: at least 1 GB of available space
    ```
    df -h /mnt/paas/kubernetes/kubelet
    ```
  - System disk: at least 2 GB of available space
    ```
    df -h /
    ```

- **Scenario 3: The available space of the /tmp directory on worker nodes is insufficient.**

Run the following command to check the usage of the file system where the **/tmp** directory is located. Ensure that the space is greater than 500 MB and check again.

```
df -h /tmp
```

2.6.5.16 Node DNS

Check Items

Check the following items:

- Check whether the DNS configuration of the current node can resolve the OBS address.
- Check whether the current node can access the OBS address of the storage upgrade component package.

Solution

During the node upgrade, obtain the upgrade component package from OBS. If this check fails, contact technical support.

2.6.5.17 Node Key Directory File Permissions

Check Items

Check whether the owner and owner group of the files in the **/var/paas** directory used by the CCE are both **paas**.

Solution

- **Scenario 1: The error message "xx file permission has been changed!" is displayed.**

  Solution: Enable CCE to use the **/var/paas** directory to manage nodes and store file data whose owner and owner group are both **paas**.

  During the current cluster upgrade, the owner and owner group of the files in the **/var/paas** directory are reset to paas.

  Check whether file data in the current service pod is stored in the **/var/paas** directory. If yes, do not use this directory, remove abnormal files from this directory, and check again. After the check is passed, proceed with the upgrade.

  ```
  find /var/paas -not \( -user paas -o -user root \) -print
  ```

- **Scenario 2: The error message "user paas must have at least read and execute permissions on the root directory" is displayed.**

  Solution: Change the permission on the root directory to the default permission 555. If the permission on the root directory of the node is modified, user **paas** does not have the read permission on the root directory. As a result, restarting the component failed during the upgrade.

2.6.5.18 kubelet

Check Items

Check whether the kubelet on the node is running properly.

Solution

- **Scenario 1: The kubelet status is abnormal.**

  If the kubelet malfunctions, the node will be unavailable. Restore the node and check again. For details, see **What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?**

- **Scenario 2: The cce-pause version is incorrect.**

  The version of the pause container image on which kubelet depends is not cce-pause:3.1. If you continue the upgrade, pods will restart in batches. Currently, the upgrade is not supported. Contact technical support.

2.6.5.19 Node Memory

Check Items

Check whether the node's memory usage is above 90%.

Solution

- **Upgrade the cluster during off-peak hours.**
- Check whether too many pods are deployed on the node. If yes, reschedule pods to other idle nodes.

2.6.5.20 Node Clock Synchronization Server

Check Items

Check whether the clock synchronization server ntpd or chronyd of the node is running properly.

Solution

- **Scenario 1: ntpd is running abnormally.**

  Log in to the node and run the **systemctl status ntpd** command to obtain the running status of ntpd. If the command output is abnormal, run the **systemctl restart ntpd** command and obtain the status again.

  The normal command output is as follows:

  Figure 2-47 Running status of ntpd

```
[root@[redacted] paas]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-12-06 14:52:30 CST; 4 days ago
 Main PID: 8587 (ntpd)
    Tasks: 2
   Memory: 1.6M
   CGroup: /system.slice/ntpd.service
           └─8587 /usr/sbin/ntpd -u ntp:ntp -g -x
```

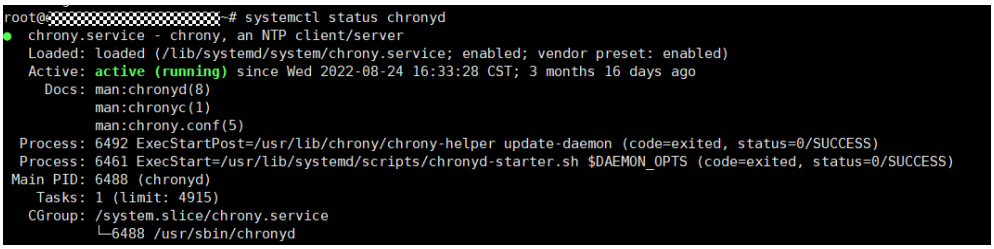

If the problem persists after ntpd is restarted, contact technical support.

- **Scenario 2: chronyd is running abnormally.**

  Log in to the node and run the **systemctl status chronyd** command to obtain the running status of chronyd. If the command output is abnormal, run the **systemctl restart chronyd** command and obtain the status again.

  The normal command output is as follows:

  Figure 2-48 Running status of chronyd

  If the problem persists after chronyd is restarted, contact technical support.

2.6.5.21 Node OS

Check Items

Check whether the OS kernel version of the node is supported by CCE.

Solution

- **Case 1: The node image is not a standard CCE image.**

  CCE nodes run depending on the initial standard kernel version specified when they are created. CCE has performed comprehensive compatibility tests based on this kernel version. A non-standard kernel version may lead to unexpected compatibility issues during a node upgrade and the upgrade may fail. For details, see **High-Risk Operations and Solutions**.

  Do not directly upgrade this type of nodes. Instead, **reset the nodes** to a standard kernel version and then upgrade the nodes.

- **Case 2: An image of a special version is defective.**

  A EulerOS release 2.8 (Arm) image of v1.17 is used in the source version. Such an image is defective because the **docker exec** command will be affected after Docker is restarted. When the cluster version is upgraded, the Docker version will be updated and Docker will be restarted. To resolve this issue, do as follows:

  a. Empty and isolate the affected nodes before upgrading the cluster.

  b. Upgrade the version to v1.19 or later and reset the nodes to replace the image with one of a later version, for example, EulerOS release 2.9.

2.6.5.22 Node CPU Cores

Check Items

Check and make sure that the master nodes in your cluster have more than 2 CPU cores.

Solution

The number of CPU cores on the master nodes is 2, which may lead to a cluster upgrade failure.

Contact technical support to expand the number of CPU cores to four or more.

2.6.5.23 Node Python Commands

Check Items

Check whether the Python commands are available on a node.

Check Method

```
/usr/bin/python --version
echo $?
```

If the command output is not 0, the check fails.

Solution

Reset the node or manually install Python before attempting the upgrade again.

2.6.5.24 ASM Version

Check Items

Check the following items:

- Check whether ASM is used by the cluster.
- Check whether the current ASM version supports the target cluster version.

Solution

- Upgrade ASM and then upgrade the cluster. The adaptation rules between ASM and cluster versions are as follows:

  Table 2-34 Adaptation rules between ASM and cluster versions

ASM Version	Cluster Version
1.3	v1.13, v1.15, v1.17, or v1.19
1.6	v1.15 or v1.17
1.8	v1.15, v1.17, v1.19, or v1.21

ASM Version	Cluster Version
1.13	v1.21 or v1.23
1.15	v1.21, v1.23, v1.25, v1.27, or v1.28
1.18	v1.25, v1.27, v1.28, v1.29, or v1.30

- If ASM is not required, delete it before the upgrade. After the upgrade, the cluster cannot be bound to ASM that does not match the table. For example, a cluster of v1.21 and ASM of v1.8 are used. If you want to upgrade the cluster to v1.25, upgrade ASM to v1.15 first.
- If you have not installed ASM, check whether the open-source Istio is installed in the cluster. If it is, check whether the current Istio version is compatible with the target cluster version. If they are compatible, skip this step and perform the check again. If they are not compatible, upgrade the Istio version and then upgrade the cluster.

2.6.5.25 Node Readiness

Check Items

Check whether the nodes in the cluster are ready.

Solution

- **Scenario 1: The nodes are in the unavailable status.**

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane and filter out unavailable nodes, rectify the faulty nodes by referring to the suggestions provided by the console, and check again.
- **Scenario 2: The displayed node status is inconsistent with the actual status.**

  The possible causes are as follows:

  a. The node status is normal on the nodes page, but the check result shows that the node is not ready. Check again.

  b. The node is not found on the nodes page, but the check result shows that the node is in the cluster. Contact technical support.

2.6.5.26 Node journald

Check Items

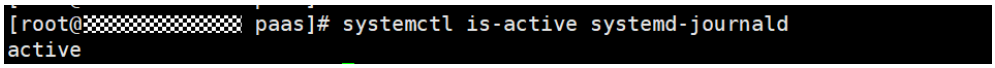
Check whether journald of a node is normal.

Solution

Log in to the node and run the **systemctl is-active systemd-journald** command to obtain the running status of journald. If the command output is abnormal, run the **systemctl restart systemd-journald** command and obtain the status again.

The normal command output is as follows:

Figure 2-49 Running status of journald



If the problem persists after journald is restarted, contact technical support.

2.6.5.27 containerd.sock

Check Items

Check whether the **containerd.sock** file is on the node. This file affects the startup of container runtime in the Euler OS.

Solution

Scenario: The Docker used by the node is the customized Euler-docker.

Step 1 Log in to the node.

Step 2 Run the **rpm -qa | grep docker | grep euleros** command. If the command output is not empty, the Docker used on the node is Euler-docker.

Step 3 Run the **stat /run/containerd/containerd.sock** command. If the file exists, Docker startup will fail.

Step 4 Run the **rm -rf /run/containerd/containerd.sock** command and perform the cluster upgrade check again.

----End

2.6.5.28 Internal Error

Check Items

This check item is not typical and implies that an internal error was found during the pre-upgrade check.

Solution

Perform the pre-upgrade check again.

If it fails again, submit a service ticket to contact technical support.

2.6.5.29 Node Mount Points

Check Items

Check whether there are inaccessible mount points on the node.

Solution

Scenario: There are inaccessible mount points on the node.

If NFS (such as obsfs or SFS) is used by the node and the node is disconnected from the NFS server, the mount point would be inaccessible and all processes that access this mount point are in D state.

Step 1 Log in to the node.

Step 2 Create a script file, for example, **/tmp/check_hang_mount.sh** on the node. The content of the script file is as follows:

```
for mount_path in `cat /proc/self/mountinfo | awk '{print $5}' | grep -v netns`
do
   timeout 10 sh -c "cd $mount_path"
   if [ $? == 124 ];then
      echo "$mount_path hang mount"
   fi
done
```

Step 3 Run the saved script and check the output.

```
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# bash test.sh
/root/foo hang mount
/root/bar hang mount
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```

The mount points of the **/root/foo** and **/root/bar** folders are incorrect.

Step 4 Run the following command to check the suspended mount points:

```
mount -n | grep /root/foo
```

```
/root/bar hang mount
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# mount -n | grep /root/foo
localhost:/tmp/nfs on /root/foo type nfs (rw,relatime,vers=3,rsize=524288,wsize=524288,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=127.0.0.1,mountvers=3,mountport=20048,mountproto=udp,local_lock=none,addr=127.0.0.1)
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```

When a mount point is suspended, it typically indicates that services are not using it anymore. After confirming that the mount point is no longer required, run the following command to unmount it and then re-execute the previously mentioned script:

```
umount -l -f localhost:/tmp/nfs
```

```
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# umount -l -f localhost:/tmp/nfs
[root@test-02-upgrade-v115-v119-vpc-06491 ~]# bash test.sh
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
[root@test-02-upgrade-v115-v119-vpc-06491 ~]#
```

After the execution, perform the check again on the pre-upgrade check page.

----End

2.6.5.30 Kubernetes Node Taints

Check Items

Check whether the taint needed for cluster upgrade exists on the node.

Table 2-35 Taint checklist

Taint Name	Impact
node.kubernetes.io/upgrade	NoSchedule

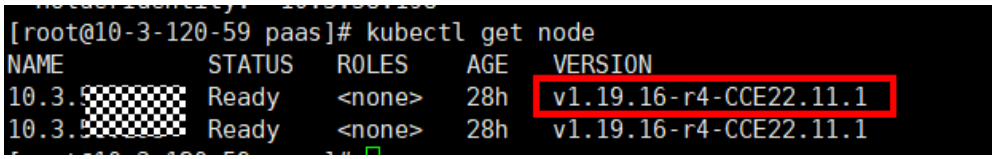
Solution

Scenario 1: The node is skipped during the cluster upgrade.

Step 1 Configure the **kubectl** command. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Check the kubelet version of the corresponding node. The following information is expected:

Figure 2-50 kubelet version



If the version of the node is different from that of other nodes, the node is skipped during the upgrade. Reset the node and upgrade the cluster again. For details about how to reset a node, see **Resetting a Node**.

NOTE

Resetting a node will reset all node labels, which may affect workload scheduling. Before resetting a node, check and retain the labels that you have manually added to the node.

----End

2.6.5.31 Everest Restrictions

Check Items

Check whether there are any compatibility restrictions on the current Everest add-on.

Table 2-36 List of Everest add-on versions with compatibility restrictions

Add-on Name	Versions Involved
everest	v1.0.2-v1.0.7 v1.1.1-v1.1.5

Solution

There are compatibility restrictions on the current Everest add-on and it cannot be upgraded with the cluster upgrade. Contact technical support.

2.6.5.32 cce-hpa-controller Limitations

Check Items

Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.

Solution

There are compatibility limitations between the current and target versions of the cce-controller-hpa add-on. To address this, install an add-on that provides metrics APIs, like metrics-server, in the cluster.

Install the metrics add-on in the cluster and try again.

2.6.5.33 Enhanced CPU Policies

Check Items

Check whether the current cluster version and the target version support **enhanced CPU policy**.

Solution

Scenario: Only the current cluster version supports the enhanced CPU policy function. The target version does not support the enhanced CPU policy function.

Upgrade to a cluster version that supports the enhanced CPU policy function. The following table lists the cluster versions that support the enhanced CPU policy function.

Table 2-37 List of cluster versions that support the enhanced CPU policy function

Cluster Version	Enhanced CPU Policy
Clusters of v1.17 or earlier	Not supported
Clusters of v1.19	Not supported
Clusters of v1.21	Not supported
Clusters of v1.23 or later	Supported

2.6.5.34 Health of Worker Node Components

Check Items

Check whether the container runtime and network components on the worker nodes are healthy.

Solution

- Issue 1: CNI Agent is not active.

  If your cluster version is earlier than v1.17.17, or you are using the underlay_ipvlan network model and your cluster version is later than v1.17.17, log in to the node and run the **systemctl status canal** command to check the status of canal. If you encounter an error, run the **systemctl restart canal** command and check the status again.

  If you are using the VPC or Cloud Native 2.0 network model and your cluster version is later than v1.17.17, log in to the node and run the **systemctl status yangtse** command to check the status of Yangtse. If you encounter an error, run the **systemctl restart yangtse** command and check the status again.
- Issue 2: kube-proxy is not active.

  Log in to the node and run the **systemctl is-active kube-proxy** command to check the status of kube-proxy. If you encounter an error, run the **systemctl restart kube-proxy** command and check the status again.

  If the fault persists, reset the node. For details, see **Resetting a Node**.

2.6.5.35 Health of Master Node Components

Check Items

Check whether cluster components such as the Kubernetes component, container runtime component, and network component are running properly before the upgrade.

Solution

Perform the pre-upgrade check again.

If it fails again, submit a service ticket to contact technical support.

2.6.5.36 Memory Resource Limit of Kubernetes Components

Check Items

Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.

Solution

- Solution 1: Reduce Kubernetes resources that are needed.
- Solution 2: Modify cluster specifications. For details, see **Changing Cluster Scale**.

2.6.5.37 Discarded Kubernetes APIs

Check Items

The system scans the audit logs of the past day to check whether the user calls the deprecated APIs of the target Kubernetes version.

NOTE

Due to the limited time range of audit logs, this check item is only an auxiliary method. APIs to be deprecated may have been used in the cluster, but their usage is not included in the audit logs of the past day. Check the API usage carefully.

Solution

Check Description

Based on the check result, it is detected that your cluster calls a deprecated API of the target cluster version using kubectl or other applications. You can rectify the fault before the upgrade. Otherwise, the API will be intercepted by kube-apiserver after the upgrade. For details about each deprecated API, see **Deprecated APIs**.

Case Study

Ingresses of the extensions/v1beta1 and networking.k8s.io/v1beta1 APIs are deprecated in Kubernetes v1.22. If you upgrade a cluster from v1.19 or v1.21 to v1.23, existing resources are not affected, but the v1beta1 API may be intercepted in the creation and editing scenarios.

For details about the YAML configuration structure changes, see **Using kubectl to Create a LoadBalancer Ingress**.

2.6.5.38 NetworkManager

Check Items

Check whether NetworkManager of a node is normal.

Solution

Log in to the node and run the **systemctl is-active NetworkManager** command to obtain the running status of NetworkManager. If the command output is abnormal, run the **systemctl restart NetworkManager** command and obtain the status again.

If the fault persists, reset the node. For details, see **Resetting a Node**. Alternatively, contact technical support to restore the file and then perform the upgrade.

2.6.5.39 Node ID File

Check Items

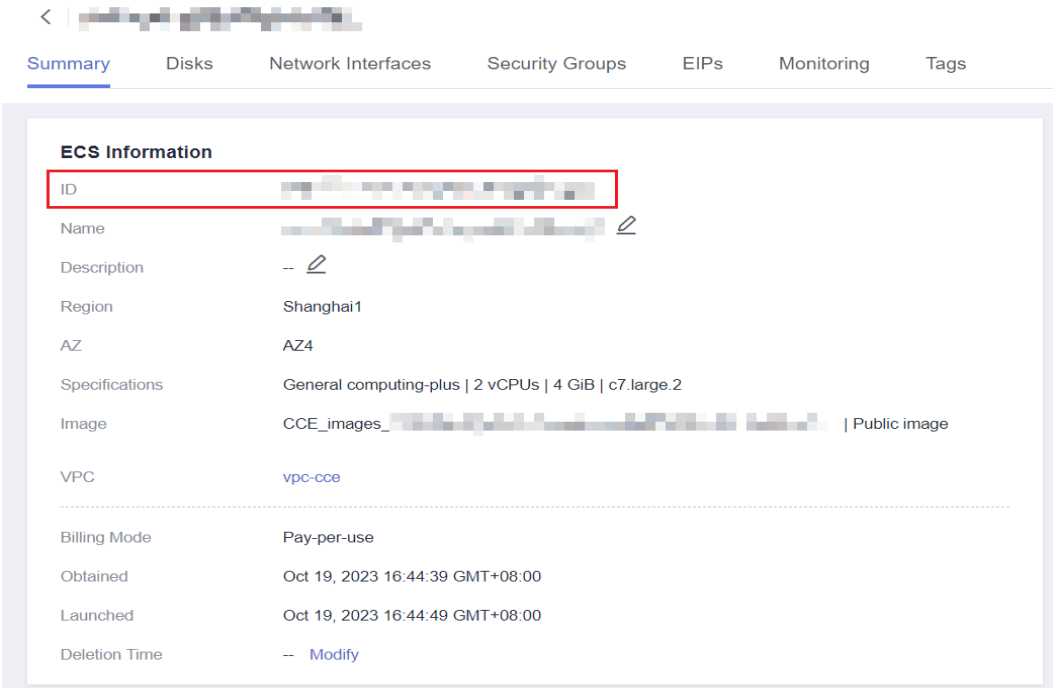
Check the ID file format.

Solution

Step 1 On the **Nodes** page of the CCE console, click the name of the abnormal node to go to the ECS page.

Step 2 Copy the node ID and save it to the local host.

Figure 2-51 Copying a node ID



Step 3 Log in to the abnormal node and back up files.

```
cp /var/lib/cloud/data/instance-id /tmp/instance-id
cp /var/paas/conf/server.conf /tmp/server.conf
```

Step 4 Log in to the abnormal node and write the obtained node ID to the file.

```
echo "Node ID" > /var/lib/cloud/data/instance-id
echo "Node ID" > /var/paas/conf/server.conf
```

----End

2.6.5.40 Node Configuration Consistency

Check Items

When you upgrade a cluster to v1.19 or later, the system checks whether the following configuration files have been modified on the backend:

- /opt/cloud/cce/kubernetes/kubelet/kubelet
- /opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml
- /opt/cloud/cce/kubernetes/kube-proxy/kube-proxy
- /etc/containerd/default_runtime_spec.json
- /etc/sysconfig/docker
- /etc/default/docker

- /etc/docker/daemon.json

If you modify some parameters in these files, the cluster upgrade may fail or services may be abnormal after the upgrade. If you confirm that the modification does not affect services, continue the upgrade.

NOTE

CCE uses the standard image script to check node configuration consistency. If you use other custom images, the check may fail.

The expected modification will not be intercepted. The following table lists the parameters that can be modified.

Table 2-38 Parameters that can be modified

Component	Configuration File	Parameter	Upgrade Version
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	cpuManagerPolicy	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	maxPods	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIQPS	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubeAPIBurst	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	podPidsLimit	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Policy	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	resolvConf	Later than v1.19
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	eventRecordQPS	Later than v1.21
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	topologyManager-Scope	Later than v1.21

Component	Configuration File	Parameter	Upgrade Version
kubelet	/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	allowedUnsafeSysctls	Later than v1.19
Docker	/etc/docker/daemon.json	dm.basesize	Later than v1.19

Solution

If you modify some parameters in these files, exceptions may occur after the upgrade. If you are not sure whether the modified parameters will affect the upgrade, contact technical support.

2.6.5.41 Node Configuration File

Check Items

Check whether the configuration files of key components exist on the node.

The following table lists the files to be checked.

File Name	File Content	Remarks
/opt/cloud/cce/kubernetes/kubelet/kubelet	kubelet command line startup parameters	None
/opt/cloud/cce/kubernetes/kubelet/kubelet_config.yaml	kubelet startup parameters	None
/opt/cloud/cce/kubernetes/kube-proxy/kube-proxy	kube-proxy command line startup parameters	None
/etc/sysconfig/docker	Docker configuration file	Not checked when containerd or the Debain-Group machine is used.
/etc/default/docker	Docker configuration file	Not checked when containerd or the Centos-Group machine is used.

Solution

Reset the node. For details, see **Resetting a Node**. Alternatively, contact technical support to restore the file and then perform the upgrade.

2.6.5.42 CoreDNS Configuration Consistency

Check Items

Check whether the current CoreDNS key configuration Corefile is different from the Helm release record. The difference may be overwritten during the add-on upgrade, **affecting domain name resolution in the cluster**.

Solution

You can upgrade CoreDNS separately after confirming the configuration differences.

Step 1 Configure the **kubectl** command. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Obtain the Corefile that takes effect currently.

```
kubectl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}' > corefile_now.txt
cat corefile_now.txt
```

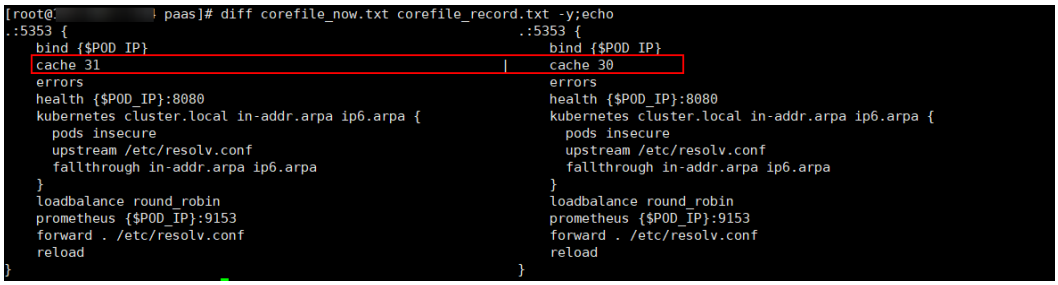
Step 3 Obtain the Corefile in the Helm Release records.

```
latest_release=`kubectl get secret -nkube-system -l owner=helm -l name=cceaddon-coredns --sort-by=.metadata.creationTimestamp | awk 'END{print $1}'`
kubectl get secret -nkube-system $latest_release -o jsonpath='{.data.release}' | base64 -d | base64 -d | gzip -d | python -m json.tool | python -c "
from __future__ import print_function
import json,sys,re,yaml;
manifests = json.load(sys.stdin)['manifest']
files = re.split('(?:^|\s*\n)---\s*',manifests)
for file in files:
    if 'coredns/templates/configmap.yaml' in file and 'Corefile' in file:
      corefile = yaml.safe_load(file)['data']['Corefile']
      print(corefile,end='')
      exit(0);
print('error')
exit(1);
" > corefile_record.txt
cat corefile_record.txt
```

Step 4 Compare the output differences between **Step 2** and **Step 3**.

```
diff corefile_now.txt corefile_record.txt -y;
```

Figure 2-52 Viewing output differences



Step 5 Return to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, select CoreDNS, and click **Upgrade**.

To retain custom configurations, use either of the following methods:

- (Recommended) Set **parameterSyncStrategy** to **inherit**. In this case, custom settings are automatically inherited. The system automatically parses, identifies, and inherits custom parameters.
- Set **parameterSyncStrategy** to **force**. Manually enter the differential configuration. For details, see **CoreDNS**.

Step 6 Click **OK**. After the add-on upgrade is complete, check whether all CoreDNS instances are available and whether Corefile meets the expectation.

```
kubectl get cm -nkube-system coredns -o jsonpath='{.data.Corefile}'
```

Step 7 Change the value of **parameterSyncStrategy** to **ensureConsistent** to enable configuration consistency verification.

In addition, it is a good practice to use the parameter configuration function of CCE add-ons to modify the Corefile configuration for consistency.

----End

2.6.5.43 sudo

Check Items

Check whether the sudo commands and sudo-related files of the node are working.

Solution

- Scenario 1: The sudo command fails to be executed.

  During the in-place cluster upgrade, the sudo command must be available. Log in to the node and run the following command to check whether the sudo command is available:

  ```
  sudo echo hello
  ```

  If the sudo command is unavailable, copy this command from another node to the target node.

- Scenario 2: Key files cannot be modified.

  During the in-place cluster upgrade, the **/etc/sudoers** and **/etc/sudoers.d/sudoerspaas** files are modified to obtain the sudo permission and update the components (such as Docker and kubelet) whose owner and owner group are **root** and related configuration files on the node. Log in to the node and run the following command to check whether the file can be modified:

  ```
  lsattr -l /etc/sudoers.d/sudoerspaas /etc/sudoers
  ```

  If **immutable** is displayed in the command output, the file is locked by the **i** lock and cannot be modified. You are advised to remove the **i** lock.

  ```
  chattr -i /etc/sudoers.d/sudoerspaas /etc/sudoers
  ```


2.6.5.44 Key Node Commands

Check Items

Whether some key commands that the node upgrade depends on are working

Solution

- Scenario 1: Executing the package manager command failed.

  Executing the **rpm** or **dpkg** command failed. In this case, log in to the affected node and check whether the following commands are available:

  ```
  rpm -qa
  ```

  If the preceding command is unavailable, run the following command:

  ```
  rpm --rebuilddb
  ```

- Scenario 2: The **systemctl status** command fails to be executed.

  If the **systemctl status** command on a node is unavailable, many check items will be affected. Log in to the node and check the availability of the following commands:

  ```
  systemctl status kubelet
  ```

If the fault persists, reset the node. For details, see **Resetting a Node**.

2.6.5.45 Mounting of a Sock File on a Node

Check Items

Check whether the **docker/containerd.sock** file is directly mounted to the pods on a node. During an upgrade, Docker or containerd restarts and the sock file on the host changes, but the sock file mounted to pods does not change accordingly. As a result, your services cannot access Docker or containerd due to sock file inconsistency. After the pods are rebuilt, the sock file is mounted to the pods again, and the issue is resolved accordingly.

Kubernetes cluster users typically use sock files in the following scenarios:

1. Monitoring applications deployed as DaemonSets use a sock file to access Docker or containerd to obtain pod statuses on a node.
2. Compilation platform applications use a sock file to access Docker or containerd to obtain containers for compiling programs.

Solution

- Scenario 1: This issue occurred on an application, and operations need to be taken to resolve this issue.

  Mount the sock file by mounting a directory. For example, if the sock file is stored in **/var/run/docker.sock** on the host, perform the following operations to resolve this issue (the following modifications will lead to the rebuilding of pods):

  ```
  kind: Deployment
  apiVersion: apps/v1
  metadata:
    name: test
  spec:
  ```



```
replicas: 1
selector:
  matchLabels:
    app: nginx
template:
  metadata:
    app: nginx
  spec:
    containers:
      - name: container-1
        image: 'nginx'
        imagePullPolicy: IfNotPresent
        volumeMounts:
          - name: sock-dir
            mountPath: /var/run
    imagePullSecrets:
      - name: default-secret
    volumes:
      - name: sock-dir
        hostPath:
          path: /var/run
```

- Scenario 2: This issue occurred on an application, and the risk that sock cannot be accessed for a short time is acceptable.

  Skip this check item and perform the check again. After the cluster is upgraded, delete the existing pods to trigger pod rebuilding. Then, the access to sock will be recovered.

- Scenario 3: This issue occurred on some CCE add-ons of earlier versions.

  Upgrade the CCE add-ons to the latest version. For example, if this issue occurred on CCE Network Metrics Exporter of a version earlier than v1.2.2, upgrade the add-on to v1.2.2 or later.

- Scenario 4: The "failed to execute docker ps -aq" error is displayed in the log analysis.

  This error is usually caused by a container engine exception. Submit a service ticket and contact O&M personnel.

2.6.5.46 HTTPS Load Balancer Certificate Consistency

Check Items

Check whether the certificate used by an HTTPS load balancer has been modified on ELB.

Solution

The certificate referenced by an HTTPS ingress created on CCE is modified on the ELB console. This leads to inconsistent certificate content in the CCE cluster and that required by the load balancer. After the CCE cluster is upgraded, the load balancer's certificate is overwritten.

Step 1 Log in to the ELB console, choose **Elastic Load Balance** > **Certificates**, locate the certificate, and find the **secret_id** in the certificate description.

Figure 2-53 Viewing a certificate

The **secret_id** is the **metadata.uid** of the secret in the cluster. Use this UID to obtain the secret name in the cluster.

Run the following kubectl command to obtain the secret name (replace *<secret_id>* with the actual value):

```
kubectl get secret --all-namespaces -o jsonpath='{range .items[*]}{"uid:"}{.metadata.uid}{" namespace:"}{.metadata.namespace}{" name:"}{.metadata.name}{"\n"}{end}' | grep <secret_id>
```

Step 2 Only clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, and later versions support certificates required by load balancers. For clusters of the earlier versions, see **Solution 1**. For clusters of other versions, see **Solution 2**.

- Solution 1: Replace the certificate used by an ingress with the one used by the load balancer. Then, you can create or edit the certificate on the ELB console.
  a. Log in to the CCE console and click the cluster name to access the cluster console. Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, locate the row containing the ingress that uses the certificate, and choose **More** > **Update** in the **Operation** column. If multiple ingresses are using this certificate, update the certificate for all of these ingresses. To check which ingresses are using a certificate, use the **secretName** parameter in **spec.tls** of the ingress YAML files.

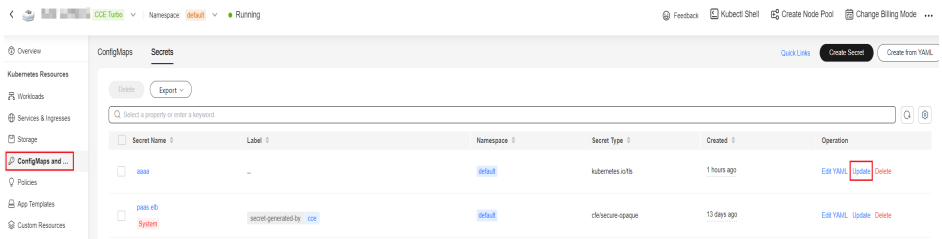
     Run the following kubectl command to obtain the ingresses using a certificate (replace *<secret_name>* with the actual value):

     ```
     kubectl get ingress --all-namespaces -o jsonpath='{range .items[*]}{"namespace:"}{.metadata.namespace}{" name:"}{.metadata.name}{" tls:"}{.spec.tls[*]}{"\n"}{end}' | grep <secret_name>
     ```

  b. When configuring a listener, select **ELB server certificate** for **Certificate Source** and click **OK**. In this way, the certificate can be created or edited on the ELB console.
  c. On the **ConfigMaps and Secrets** page, delete the target secret. Before the deletion, back up data.
- Solution 2: Overwrite the certificate used by an ingress with the corresponding secret resource of the cluster to prevent the certificate being updated on the ELB console during the cluster upgrade.

  Log in to the CCE console and click the cluster name to access the cluster console. Choose **ConfigMaps and Secrets** from the navigation pane, click the **Secrets** tab, locate the row containing the target secret, click **Update** in the **Operation** column, and enter the certificate you are using.

Figure 2-54 Modifying a secret



----End

2.6.5.47 Node Mounting

Check Items

Check whether the default mount directory and soft link on the node have been manually mounted or modified.

- Non-shared disk
  - By default, **/var/lib/docker**, **containerd**, or **/mnt/paas/kubernetes/kubelet** is mounted to CCE nodes. Check whether **/var**, **/var/lib**, **/mnt**, **/mnt/paas**, and **/mnt/paas/kubernetes** have been manually mounted.
  - CCE creates the **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet** link by default. Check whether the link has been manually modified.
- Shared disk
  - By default, **/mnt/paas/** is mounted to CCE nodes. Check whether **/mnt** has been manually mounted.
  - CCE creates the **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet** and **/var/lib/docker** or **/var/lib/containerd** to **/mnt/paas/runtime** soft links by default. Check whether the links have been manually modified.

Solution

How Do I Check Whether a Disk Is Shared?

Step 1 Log in to the target node based on the check information.

Step 2 Run the **lsblk** command to check whether **vgpaas-share** is mounted to **/mnt/paas**. If yes, a shared disk is used.

Figure 2-55 Checking whether a shared disk is used

```
[root@test-os-ugrade-35777 ~]# lsblk
NAME           MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda            253:0    0   50G  0 disk
└─vda1         253:1    0   50G  0 part /
vdb            253:16   0  100G  0 disk
└─vgpaas-share 252:0    0  100G  0 lvm  /mnt/paas
```

----End

What Can I Do If an Error Occurred in a Node Mounting Check?

1. Cancel the manually modified mount point.
2. Cancel the modification on the default soft links.

2.6.5.48 Login Permissions of User paas on a Node

Check Items

Check whether user **paas** is allowed to log in to a node.

Solution

Run the following command to check whether user **paas** is allowed to log in to a node:

```
sudo grep "paas" /etc/passwd
```

If the permissions assigned to user **paas** contain **nologin** or **false**, the user does not have the login permission. In this case, restore the login permission of user **paas**.

Run the following command to restore the login permission of user **paas**:

```
usermod -s /bin/bash paas
```

2.6.5.49 Private IPv4 Addresses of Load Balancers

Check Items

Check whether the load balancer associated with a Service is allocated with a private IPv4 address.

Solution

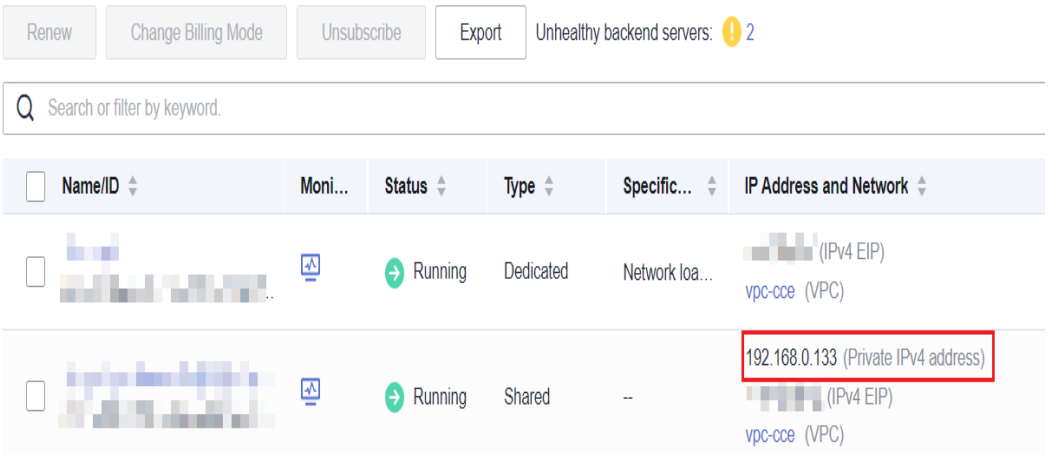
Solution 1: Delete the Service that is associated with a load balancer without a private IPv4 address.

Solution 2: Bind a private IP address to the load balancer without a private IPv4 address. The procedure is as follows:

Step 1 Obtain the load balancer associated with the target Service.

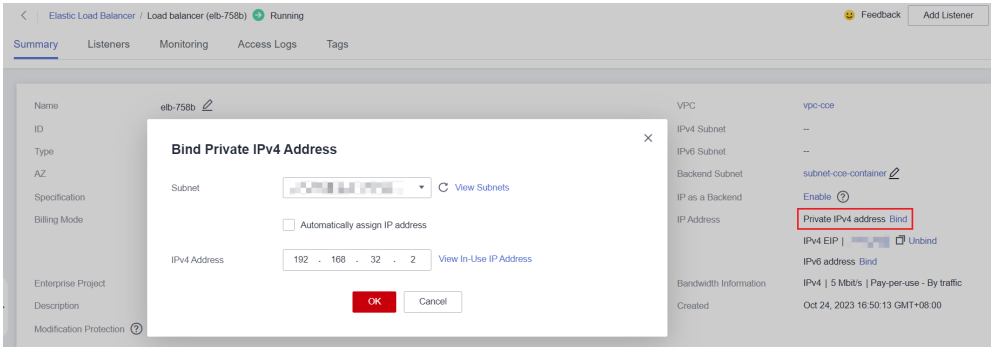
- Method 1: Obtain the load balancer ID based on the pre-upgrade check log. Go to the ELB console and filter load balancers by load balancer ID.
  ```
  elbs (ids: [*****]) without ipv4 private ip, please bind private ip to these elbs and try again
  ```
- Method 2: Log in to the CCE console and click the cluster name to access the cluster console. Then, choose **Services & Ingresses** in the navigation pane. In the right pane, click the name of the target load balancer to go to the ELB page.

Step 2 Check whether the load balancer has a private IPv4 address.



Step 3 Bind a private IP address to the load balancer without a private IPv4 address.

1. Log in to the CCE console and click the name of the target load balancer.
2. On the **Summary** tab, click **Bind** next to **Private IPv4 address**.
3. Configure the subnet and IPv4 address, and click **OK**.



----End

2.6.5.50 Historical Upgrade Records

Check Items

Check the historical upgrade records of the cluster and confirm that the current version of the cluster meets the requirements for upgrading to the target version.

Solution

Upgrading your cluster from an earlier version can be risky and may result in this issue. To avoid this, it is recommended that you migrate the cluster beforehand.

If you still want to proceed with the cluster upgrade, submit a service ticket to contact technical support for evaluation.

2.6.5.51 CIDR Block of the Cluster Management Plane

Check Items

Check whether the CIDR block of the cluster management plane is the same as that configured on the backbone network.

Solution

The CIDR block of the management plane has been modified in your region. As a result, the CIDR block of the management plane and that of the backbone network are inconsistent.

Submit a service ticket to contact technical support to modify the settings, and recheck the configurations.

2.6.5.52 GPU Add-on

Check Items

The GPU add-on is involved in the upgrade, which may affect the GPU driver installation during the creation of a GPU node.

Solution

The GPU add-on driver needs to be configured by yourself. Check the compatibility between the GPU add-on and the GPU driver. It is a good practice to verify the upgrade of the GPU driver to the target version in the test environment, configure the current GPU driver, and check whether the created GPU node can run properly.

Perform the following operations to check the upgrade of the GPU driver to the target version and current driver configuration of GPU add-on:

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. Then, check the CCE AI Suite (NVIDIA GPU) add-on.

Step 2 Click **Upgrade** of the add-on and check the target version and driver version of the add-on.

Step 3 Verify the upgrade of the GPU driver to the target version in the test environment, configure the current GPU driver, and check whether the created GPU node can run properly.

If the GPU add-on and the GPU driver are incompatible, install the driver of a later version. If necessary, contact technical support.

----End

2.6.5.53 Nodes' System Parameters

Check Items

Check whether the default system parameter settings on your nodes are modified.

Solution

If the MTU value of the bond0 network on your BMS node is not the default value 1500, this check item failed.

Non-default parameter settings may lead to service packet loss. Change them back to the default values.

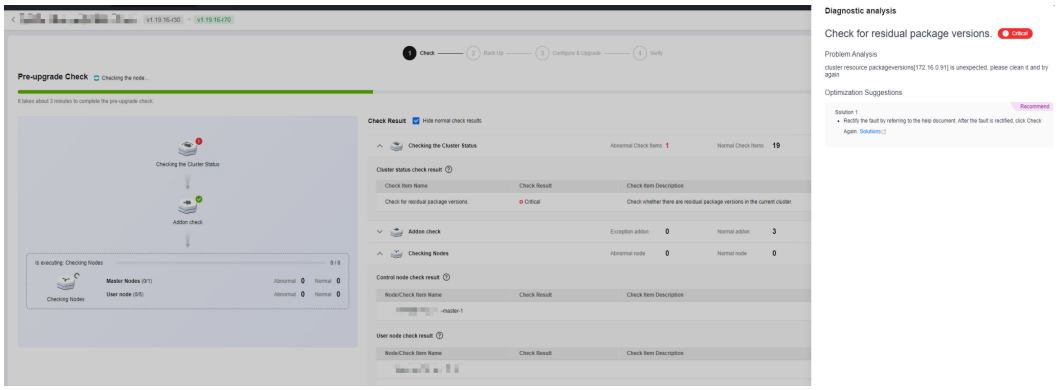
2.6.5.54 Residual Package Version Data

Check Items

Check whether there are residual package version data in the current cluster.

Solution

A message is displayed indicating that there are residual 10.12.1.109 CRD resources in your cluster. This issue occurs because CRD resources are not cleared after nodes in earlier CCE versions are deleted.



Manually perform the following operations to clear the residual resources:

Step 1 Back up the residual CRD resources. Take CRD resource 10.12.1.109 as an example. Replace it with the resource displayed in the error message.

```
kubectl get packageversion 10.12.1.109 -oyaml > /tmp/packageversion-109.bak
```

Step 2 Clear the residual CRD resources.

```
kubectl delete packageversion 10.12.1.109
```

Step 3 Check residual package versions again.

----End

2.6.5.55 Node Commands

Check Items

Check whether the commands required for the upgrade are available on the node.

Solution

The cluster upgrade failure is typically caused by the lack of key node commands that are required in the cluster upgrade.

Error messages:

```
__error_code#ErrorCommandNotExist#chage command is not exists#__
__error_code#ErrorCommandNotExist#chown command is not exists#__
__error_code#ErrorCommandNotExist#chmod command is not exists#__
__error_code#ErrorCommandNotExist#mkdir command is not exists#__
__error_code#ErrorCommandNotExist#in command is not exists#__
__error_code#ErrorCommandNotExist#touch command is not exists#__
__error_code#ErrorCommandNotExist#pidof command is not exists#__
```

The preceding error messages indicate the lack of node commands such as **chage**, **chown**, and **chmod**. Add these commands and check the node commands again.

2.6.5.56 Node Swap

Check Items

Check whether swap has been enabled on CCE nodes.

Solution

By default, swap is disabled on CCE nodes. Check the necessity of enabling swap manually and determine the impact of disabling this function.

If there is no impact, run the **swapoff -a** command to disable swap and perform the check again.

2.6.5.57 NGINX Ingress Controller

Check Items

- Check item 1: Check whether there is an Nginx Ingress route whose ingress type is not specified (**kubernetes.io/ingress.class: nginx** is not added to **annotations**) in the cluster.
- Check item 2: Check whether the DefaultBackend Service specified by the NGINX Ingress Controller backend is available.

Fault Locating

For Check Item 1

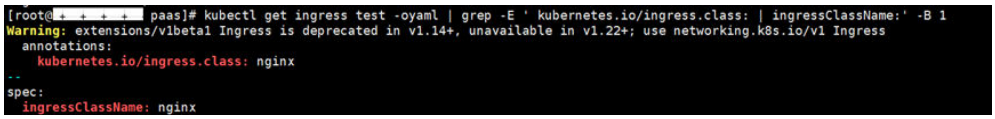
For Nginx Ingress, check the YAML. If the ingress type is not specified in the YAML file and the ingress is managed by the NGINX Ingress Controller, the ingress is at risk. For details, see **Possible Causes**.

Step 1 Check the ingress type.

Run the following command:

```
kubectl get ingress <ingress-name> -oyaml | grep -E ' kubernetes.io/ingress.class: | ingressClassName:' -B 1
```

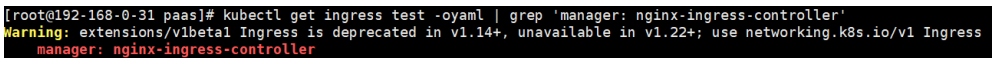
- Fault scenario: If the command output is empty, the ingress type is not specified.
- Normal scenario: The command output is not empty, indicating that the ingress type has been specified by **annotations** or **ingressClassName**.



Step 2 Ensure that the ingress is managed by the Nginx ingress Controller. The LoadBalancer Ingresses are not affected by this issue.

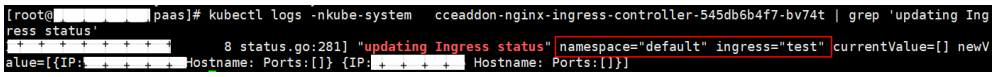
- For clusters of v1.19, confirm this issue using **managedFields**.

```
kubectl get ingress <ingress-name> -oyaml | grep 'manager: nginx-ingress-controller'
```



- For clusters of other versions, check the logs of the NGINX Ingress Controller pod.


```
kubectl logs -nkube-system cceaddon-nginx-ingress-controller-545db6b4f7-bv74t | grep 'updating Ingress status'
```



If the fault persists, contact technical support.

----End

For Check Item 2

Step 1 View the DefaultBackend Service in the namespace where the NGINX Ingress Controller is deployed.

```
kubectl get pod cceaddon-nginx-ingress-<controller-name>-controller-*** -n <namespace> -oyaml | grep 'default-backend'
```

In the preceding command, *cceaddon-nginx-ingress-<controller-name>-controller-**** is the controller pod name, *<controller-name>* is the controller name specified during add-on installation, and *<namespace>* is the namespace where the controller is deployed.

Command output:

```
- '--default-backend-service=<namespace>/<backend-svc-name>'
```

In the preceding command, *<backend-svc-name>* is the name of the DefaultBackend Service for the NGINX Ingress Controller.

Step 2 Check whether the DefaultBackend Service of the NGINX Ingress Controller is available.

```
kubectl get svc <backend-svc-name> -n <namespace>
```

If the Service is unavailable, this check item failed.

----End

Solution

For Check Item 1

Add an annotation to the Nginx ingresses as follows:

```
kubectl annotate ingress <ingress-name> kubernetes.io/ingress.class=nginx
```

NOTICE

There is no need to add this annotation to LoadBalancer ingresses. **Verify** that these ingresses are managed by NGINX Ingress Controller.

For Check Item 2

Create the DefaultBackend Service again.

- If a custom DefaultBackend Service has been specified in the default 404 service configuration during add-on installation, create the same Service.
- If the default DefaultBackend Service is used during add-on installation, the re-created YAML example is as follows:


```
apiVersion: v1
kind: Service
metadata:
  name: cceaddon-nginx-ingress-<controller-name>-default-backend    # <controller-name> is the
controller name.
  namespace: kube-system
  labels:
    app: nginx-ingress-<controller-name>
    app.kubernetes.io/managed-by: Helm
    chart: nginx-ingress-<version>    # <version> is the add-on version.
    component: default-backend
    heritage: Helm
    release: cceaddon-nginx-ingress-<controller-name>
  annotations:
    meta.helm.sh/release-name: cceaddon-nginx-ingress-<controller-name>
    meta.helm.sh/release-namespace: kube-system    # Namespace where the add-on is installed
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: http
  selector:
    app: nginx-ingress-<controller-name>
    component: default-backend
    release: cceaddon-nginx-ingress-<controller-name>
  type: ClusterIP
  sessionAffinity: None
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  internalTrafficPolicy: Cluster
```

2.6.5.58 Upgrade of Cloud Native Cluster Monitoring

Check Items

During a cluster upgrade, compatibility issues occur when the Cloud Native Cluster Monitoring add-on is upgraded from a version earlier than 3.9.0 to a version later than 3.9.0. Check whether Grafana is enabled for this add-on.

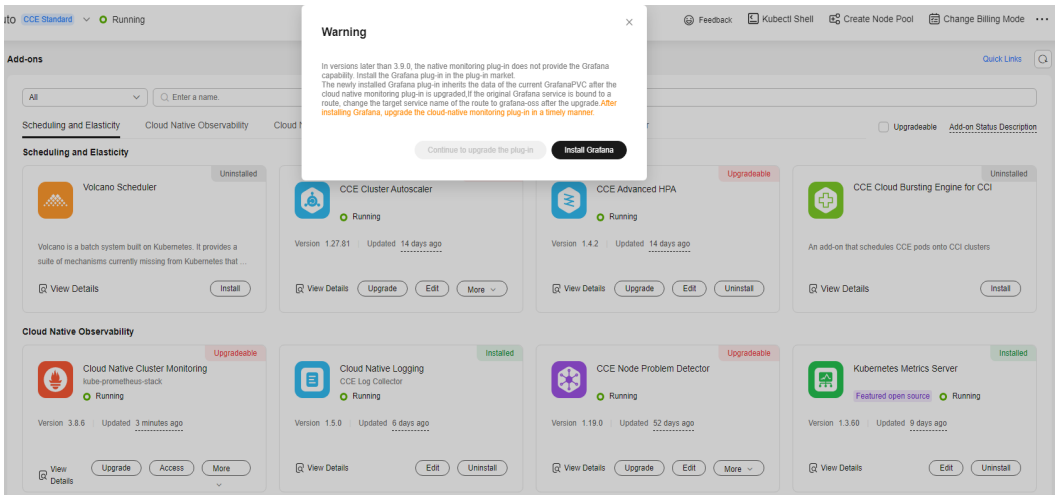
Solution

In versions later than 3.9.0, Cloud Native Cluster Monitoring is not built in with Grafana capabilities anymore. Therefore, reinstall an open-source Grafana version before upgrading Cloud Native Cluster Monitoring.

NOTE

- Reinstalling Grafana does not affect existing data.
- Manually created Grafana Services and ingresses cannot be directly associate with the newly installed Grafana. To resolve this issue, manually change the selector for the affected Services and ingresses.

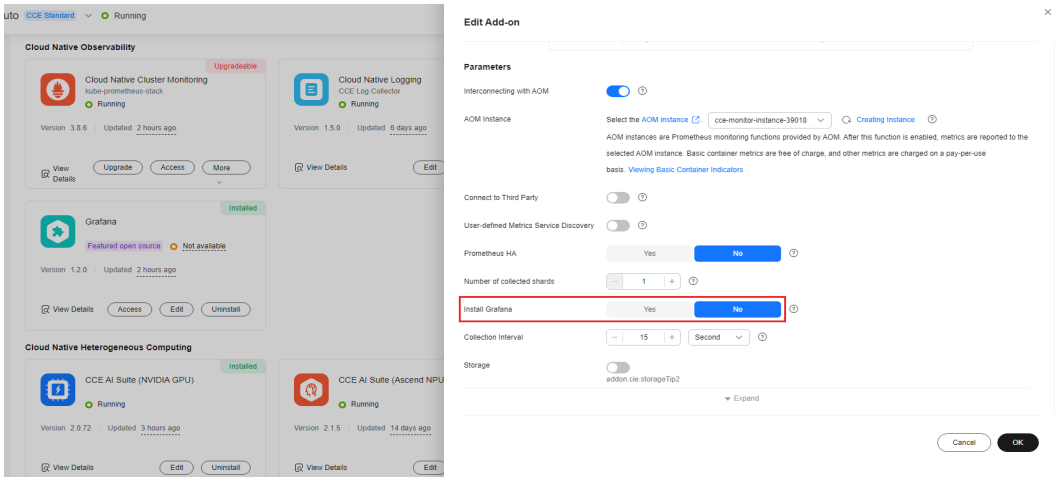
Solution 1: If Cloud Native Cluster Monitoring can be upgraded to 3.9.0 or later, upgrade it on the **Add-ons** page. In the displayed dialog box, click **Install Grafana**. After the request is submitted, continue to upgrade the add-on to the latest version.



Solution 2: If Cloud Native Cluster Monitoring cannot be upgraded to 3.9.0 or later, perform the following operations to manually migrate Grafana:

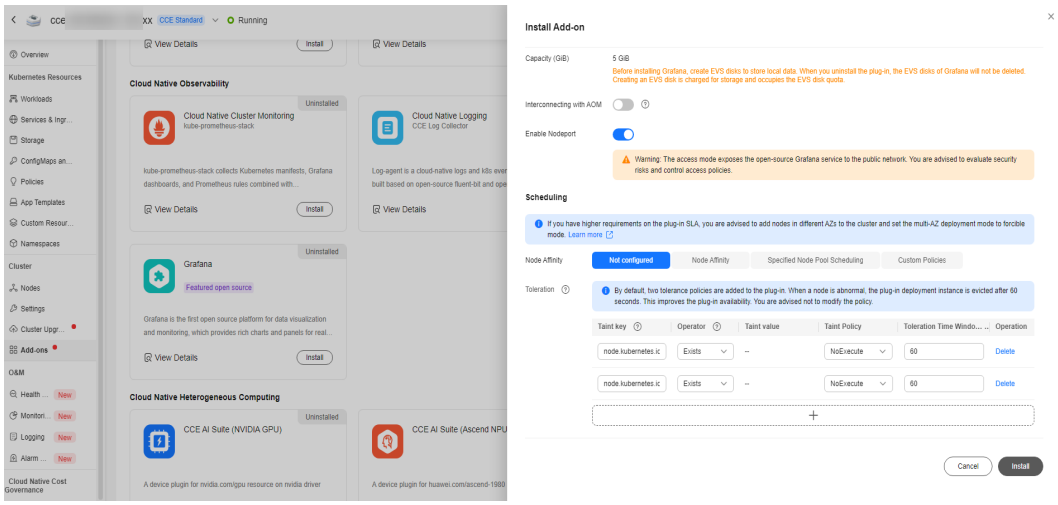
Step 1 Disable Grafana on Cloud Native Cluster Monitoring.

Go to the add-on list, click **Edit** of Cloud Native Cluster Monitoring. On the **Edit Add-on** page, disable Grafana.



Step 2 Install open-source Grafana.

On the **Add-ons** page, install Grafana.



----End

2.6.5.59 containerd Pod Restart Risks

Check Items

Check whether the service pods running on a containerd node are restarted when containerd is upgraded.

Solution

containerd on your node may need to be restarted. To minimize the impact on service containers, upgrade the cluster during controllable times, such as off-peak hours.

If you need help, submit a service ticket to contact O&M personnel.

2.6.5.60 Key GPU Add-on Parameters

Check Items

Check whether the configuration of the CCE AI Suite add-on in a cluster has been intrusively modified. If so, upgrading the cluster may fail.

Solution

Step 1 Use kubectl to access the cluster.

Step 2 Run the following command to obtain the add-on instance details:

```
kubectl get ds nvidia-driver-installer -nkube-system -oyaml
```

Step 3 Check whether the **UpdateStrategy** value is changed to **OnDelete**. If so, change it back to **RollingUpdate**.

Step 4 Check whether the **NVIDIA_DRIVER_DOWNLOAD_URL** value is the same as the GPU driver version on the add-on page. If not, correct the version on the add-on page.

----End

2.6.5.61 GPU or NPU Pod Rebuild Risks

Check Items

Check whether GPU or NPU service pods are rebuilt in a cluster when kubelet is restarted during the upgrade of the cluster.

Solution

Upgrade the cluster when the impact on services is controllable (for example, during off-peak hours) to minimize the impact.

If you need help, submit a service ticket to contact O&M personnel.

2.6.5.62 ELB Listener Access Control

Check Items

Check whether ELB listener access control has been configured using annotations for the Services in the current cluster.

If so, check whether their configurations are correct.

Solution

If the configurations are incorrect, reconfigure them by following the operations provided in **Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Service**.

2.6.5.63 Master Node Flavor

Check Items

Check whether the flavor of the master nodes in the cluster is the same as the actual flavor of these nodes.

Solution

This issue is typically caused by modifications made to the master node. This upgrade may reset the node.

If you are unsure about the impact, submit a service ticket to contact O&M personnel.

2.6.5.64 Subnet Quota of Master Nodes

Check Items

Check whether the number of available IP addresses in the cluster subnet supports rolling upgrade.

Solution

Rolling upgrade is not supported if there are not enough IP addresses in the selected cluster subnet.

Move nodes out of the target subnet and check again. If you are unsure about the impact of migration, submit a service ticket to contact O&M personnel.

2.6.5.65 Node Runtime

Check Items

Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.

Solution

If you plan to create and use Docker nodes in clusters of versions later than v1.27, ignore this alarm. However, you are advised to switch to containerd as soon as possible for a better user experience and more powerful functions.

2.6.5.66 Node Pool Runtime

Check Items

Check whether an alarm is generated when a cluster is upgraded to v1.27 or later. Do not use Docker in clusters of versions later than 1.27 because CCE is going to stop the support for Docker.

Solution

If you plan to create and use Docker node pools in clusters of versions later than v1.27, ignore this alarm. However, you are advised to switch to containerd as soon as possible for a better user experience and more powerful functions.

2.6.5.67 Number of Node Images

Check Items

Check the number of images on your node. If there are more than 1000 images, it takes a long time for Docker to start, affecting the standard Docker output and functions such as Nginx.

Solution

Manually delete residual images.

Perform the pre-upgrade check again.

2.6.5.68 OpenKruise Compatibility Check

Check Items

Check whether the OpenKruise add-on is compatible before upgrading a cluster.

Solution

Since version 1.24, the Kubernetes community no longer supports Dockershim. CCE uses cri-dockerd as an alternative to Dockershim in clusters of v1.25 or later to accommodate users' Docker habits. However, the OpenKruise community does not support cri-dockerd. For details, see **issue**.

If you install OpenKruise v1.0.3 in a cluster of v1.25 or later, kruise-daemon cannot run on a Docker node. In this case, use a containerd node.

Use either of the following methods to resolve this issue:

- Method 1: Disable OpenKruise kruise-daemon and upgrade the cluster.
- Method 2: Change the node runtime from Docker to containerd. For details, see **Migrating Nodes from Docker to containerd**.

2.6.5.69 Compatibility Check of Secret Encryption

Check Items

Check whether the target version supports secret encryption. If it does not, clusters that have this feature enabled cannot be upgraded to the target version.

Solution

Secret encryption is supported in CCE clusters of v1.27 or later versions. The feature is available in the following versions:

- v1.27: v1.27.10-r0 or later
- v1.28: v1.28.8-r0 or later
- v1.29: v1.29.4-r0 or later
- Other clusters of later versions

If secret encryption has been enabled in the cluster before the upgrade, the target cluster version must also support this feature. Select a target version that supports secret encryption for the upgrade.

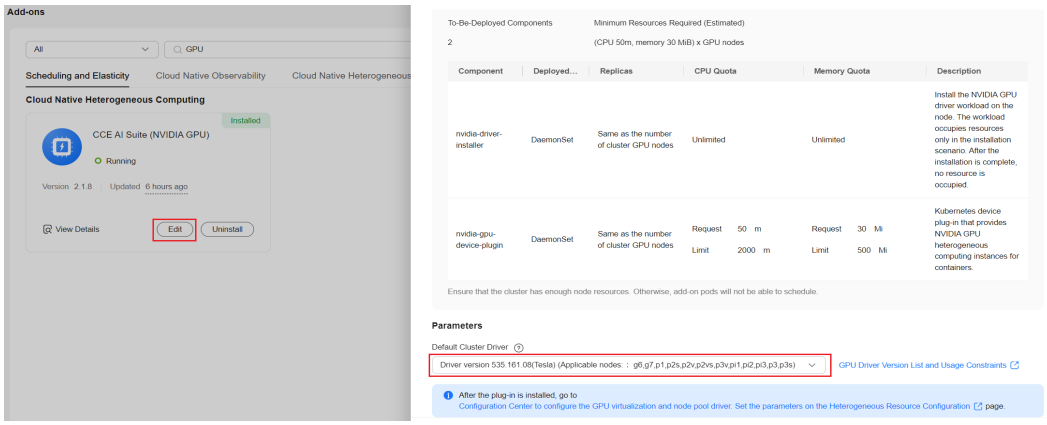
2.6.5.70 Compatibility Between the Ubuntu Kernel and GPU Driver

Check Items

Make sure that the GPU add-on and Ubuntu nodes are compatible before using them in a cluster. If the Ubuntu kernel is 5.15.0-113-generic, the driver of the GPU add-on must be 535.161.08 or later.

Solution

This issue may occur when you create or reset an Ubuntu node after an upgrade. In this case, upgrade the GPU add-on driver to version 535.161.08 or later, and restart the node.



2.6.5.71 Drainage Tasks

Check Items

An unfinished drainage task is detected in the cluster, which may resume after the upgrade. If this happens, running pods will be evicted, which could impact your services.

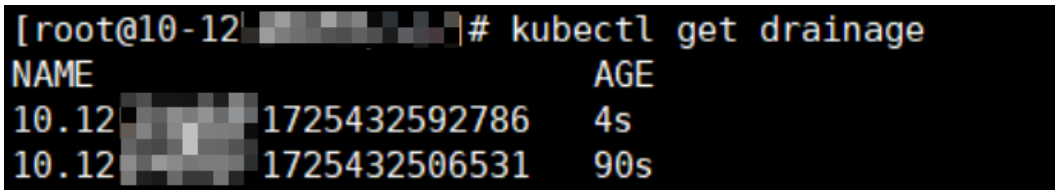
Solution

Step 1 Configure the **kubectl** command. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Check if there are any drainage tasks in progress.

```
kubectl get drainage
```

Figure 2-56 An in-progress drainage task



Delete the drainage resources and perform the pre-upgrade check again.

Step 3 Run the following command to delete a drainage task:

```
kubectl delete drainage {Name of the drainage task}
```

----End

2.6.5.72 Image Layers on a Node

Check Items

Check the number of image layers on your node. If there are more than 5000 layers, it will take a long time for Docker or containerd to start, affecting the stdout of Docker or containerd.

If Nginx is used in your cluster, data forwarding may be slow.

Solution

Log in to the node and manually delete unnecessary images.

2.6.5.73 Cluster Rolling Upgrade

Check Items

Check whether your cluster is eligible for a rolling upgrade. The result shows that the rolling upgrade is not supported.

Solution

Rolling upgrades cannot be performed if the tenant's resource quotas are insufficient.

Contact O&M personnel to expand resources and perform the check again.

2.6.5.74 Rotation Certificates

Check Items

Check whether the number of certificates on your node is greater than 1000. During an upgrade, certificate files will be processed in batches. An excessive number of certificate files will lead to a slow node upgrade and result in pod eviction from the node.

Solution

Solution 1 (preferred): Reset the node. For details, see **Resetting a Node**.

Solution 2: Fix the certificate rotation issue on the node.

1. Log in to the node and go to the node certificate directory.
   ```
   cd /opt/cloud/cce/kubernetes/kubelet/pki/
   ```
2. Back up the certificate file **kubelet-client-current.pem** on the node.
3. Delete the residual **kubelet-server-*** certificate file and certificate soft link file from the node.
   ```
   find -maxdepth 1 -type f -name 'kubelet-server-*.pem' -delete
   rm -f ./kubelet-server-current.pem
   ```


2.6.5.75 Ingress and ELB Configuration Consistency

Check Items

Check whether any modifications have been made to the listener, forwarding policy, forwarding rule, backend cloud server group, backend cloud server, or certificate configurations that were automatically generated by the ingress on the ELB console.

Any modifications made to the ELB will be overwritten during the upgrade process. Verify the configuration consistency before upgrading the cluster.

Solution

Based on the diagnostic analysis logs, determine which configurations require correction. Typically, this issue occurs when different forwarding policies are configured for the listener associated with the target ingress. This causes a mismatch with the forwarding policies configured for the cluster ingress. To resolve this, either transfer the additional forwarding policies to a different listener or add them to the ingress.

2.6.5.76 Network Policies of Cluster Network Components

Check Items

Check the network policy settings on the master nodes in your cluster. If any manual modifications have been made, they will be reset during the upgrade.

Solution

Check whether network policies need to be disabled on the network component canal-controller based on the diagnostic analysis logs. For example, if a cluster uses a Direct Connect connection to access an external address, the external switch does not support **ip-option**. Enabling network policies in this scenario could result in network access failure. In such cases, manually disable the network policies.

If you need to modify the network policy setting, skip this check. After the upgrade, you can modify the network policy setting on the **Network** tab in **Settings**.

2.6.5.77 Cluster and Node Pool Configurations

Check Items

Check whether the **nic-max-above-warm-target** value configured for the network component of the current cluster exceeds the maximum value allowed.

Solution

Determine the scope of impact based on the error information. For example:

configuration check failed: [**nodepool id(1786cd55-xxxx-xxxx-aac8-0255ac100095)**: [eni/nic-max-above-warm-target] should be less than or equal to 256, **nodepool id(master)**: [eni/nic-max-above-warm-target] should be less than or equal to 256]. Please make corrections at settings

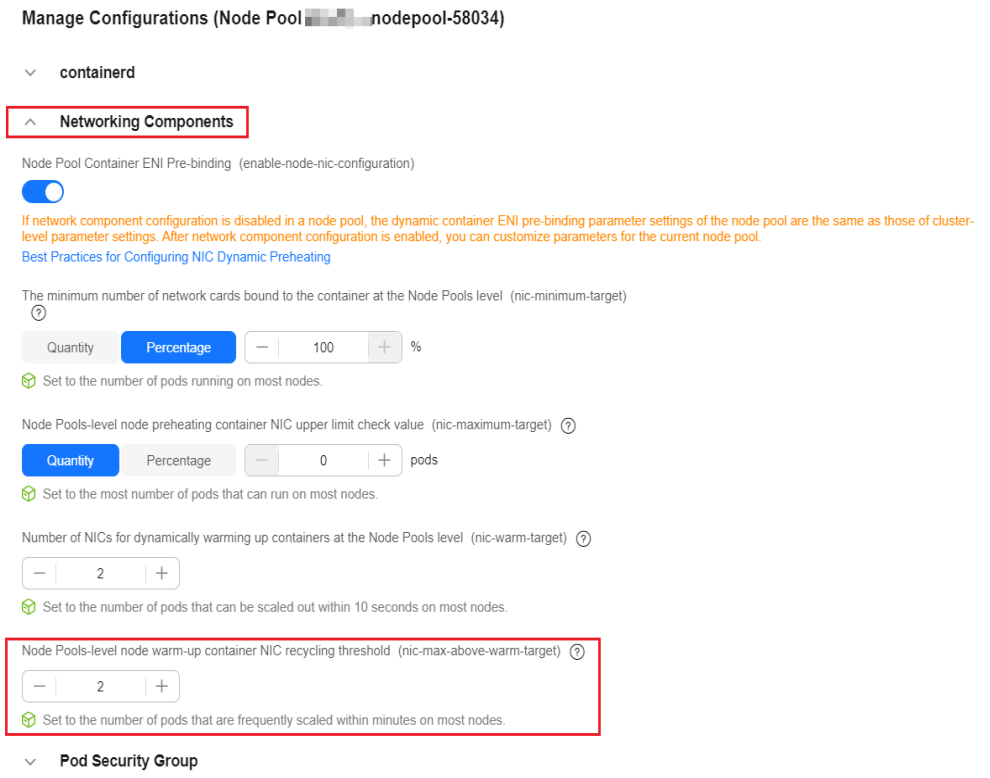
This error information indicates that there are two abnormal **nic-max-above-warm-target** settings:

- **nodepool id(1786cd55-xxxx-xxxx-aac8-0255ac100095)** indicates that the node pool with the ID **1786cd55-xxxx-xxxx-aac8-0255ac100095** has an abnormal configuration.
- **nodepool id(master)** indicates that the cluster has an abnormal configuration.

Scenario 1: Abnormal node pool configuration

To fix this issue, perform the following operations:

1. Log in to the CCE console. In the navigation pane, choose **Nodes**.
2. Locate the affected node pool and choose **More** > **Manage**.
3. In **Networking Components**, set the **nic-max-above-warm-target** value to a maximum of 256.

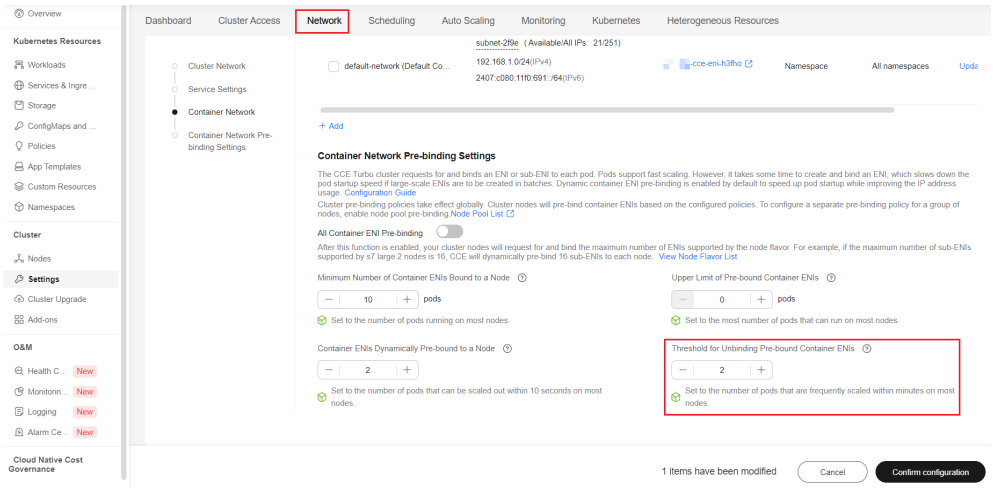


4. Click **OK**.

Scenario 2: Abnormal cluster configuration

To fix this issue, perform the following operations:

1. Log in to the CCE console. In the navigation pane, choose **Settings**.
2. Click the **Network** tab and set the **Threshold for Unbinding Pre-bound Container ENIs** value to a maximum of 256.



3. Click **Confirm Configuration**.

2.6.5.78 Time Zone of Master Nodes

Check Items

Check whether the time zone of the master nodes matches the cluster's time zone. If they are different, the master nodes will be updated to match the cluster's time zone during a rolling upgrade.

If there is a CronJob in your cluster, it may unexpectedly trigger an execution after the upgrade.

Solution

Before the upgrade, disable the CronJob and perform the pre-upgrade check. After the upgrade is complete, enable the CronJob.

3 Nodes

3.1 Node Overview

Introduction

A container cluster consists of a set of worker machines, called nodes, that run containerized applications. A node can be a virtual machine (VM) or a physical machine (PM), depending on your service requirements. The components on a node include kubelet, container runtime, and kube-proxy.

NOTE

A Kubernetes cluster consists of master nodes and worker nodes. The nodes described in this section refer to **worker nodes**, which are computing nodes of a cluster that run containerized applications.

CCE uses high-performance Elastic Cloud Servers (ECSs) or Bare Metal Servers (BMSs) as nodes to build highly available Kubernetes clusters.

Supported Node Specifications

Different regions support different node flavors, and node flavors may be changed or sold out. Log in to the CCE console and check whether the required node flavors are supported on the page for creating nodes.

Underlying File Storage System of Containers

Docker

- In clusters of v1.15.6 or earlier, the underlying Docker file storage system is in XFS format.
- In clusters of v1.15.11 or later, after a node is created or reset, the underlying Docker file storage system changes to the ExtFS format.

For containerized applications that use the XFS format, pay attention to the impact of the underlying file storage format change. (The sequence of files in different file systems is different. For example, some Java applications reference a JAR package, but the directory contains multiple versions of the JAR package. If

the version is not specified, the actual referenced package is determined by the system file.)

Run the **docker info | grep "Backing Filesystem"** command to check the format of the underlying Docker storage file used by the current node.

containerd

Nodes running on containerd use the ext4 file storage system.

paas User and User Group

When you create a node in a cluster, the paas user or a user group will be created on the node by default. CCE components and CCE add-ons on a node run as a non-root user (user **paas** or a user group) to minimize the running permission. If the paas user or user group is modified, CCE components and pods may fail to run properly.

NOTICE

The normal running of CCE components depends on the paas user or user group. Pay attention to the following requirements:

- Do not modify the directory permission and container directory permission on a node.
- Do not change the GID and UID of the paas user or user group.
- Do not directly use the paas user or user group to set the user and group to which the service file belongs.

Node Lifecycle

A lifecycle indicates the node statuses recorded from the time when the node is created through the time when the node is deleted or released.

Table 3-1 Node statuses

Status	Status Attribute	Description
Running	Stable state	The node is running properly and is connected to the cluster. Nodes in this state can provide services.
Unavailable	Stable state	The node is not functioning correctly, which includes being in a stopped state. Instances in this state cannot provide services.
Creating	Intermediate state	The node has been created but is not running.
Installing	Intermediate state	The Kubernetes software is being installed on the node.

Status	Status Attribute	Description
Upgrading	Intermediate state	The node is being upgraded.
Deleting	Intermediate state	The node is being deleted. If this state stays for a long time, an exception occurred.
Error	Stable state	The node is abnormal. Instances in this state no longer provide services. In this case, perform the operations in **Resetting a Node**.

3.2 Container Engines

Introduction to Container Engines

Container engines, one of the most important components of Kubernetes, manage the lifecycle of images and containers. The kubelet interacts with a container runtime through the Container Runtime Interface (CRI).

CCE supports containerd and Docker. **containerd is recommended for its shorter traces, fewer components, higher stability, and less consumption of node resources**.

Kubernetes has removed dockershim from v1.24 and does not support Docker by default. For details, see **Kubernetes is Moving on From Dockershim: Commitments and Next Steps**. CCE is going to stop the support for Docker. Change the node container engine from Docker to containerd. For details, see **Migrating Nodes from Docker to containerd**.

Table 3-2 Comparison between container engines

Item	containerd	Docker
Tracing	kubelet --> CRI plugin (in the containerd process) --> containerd	• Docker (Kubernetes v1.23 and earlier versions): kubelet --> dockershim (in the kubelet process) --> docker --> containerd • Docker (community solution for Kubernetes v1.24 or later): kubelet --> cri-dockerd (kubelet uses CRI to connect to cri-dockerd) --> docker--> containerd
Command	crictl/ctr	docker

Item	containerd	Docker
Kubernetes CRI	Native support	Support through dockershim or cri-dockerd
Pod delayed startup	Minor	High
kubelet CPU/ memory usage	Minor	High
Runtime's CPU/memory usage	Minor	High

Mapping Between Node OSs and Container Engines

NOTE

VPC network clusters of v1.23 or later versions support containerd. Tunnel network clusters of v1.23.2-r0 or later versions support containerd.

Table 3-3 Node OSs and container engines in CCE clusters

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
CentOS 7.6	3.x	Docker Clusters of v1.23 and later support containerd.	Clusters of v1.19.16 and earlier use Device Mapper. Clusters of v1.19.16 and later use OverlayFS.	runC
EulerOS 2.3	3.x	Docker	Device Mapper	runC
EulerOS 2.5	3.x	Docker	Device Mapper	runC
EulerOS 2.9	4.x	Docker Clusters of v1.23 and later support containerd.	OverlayFS	runC
Ubuntu 18.04	4.x	Docker Clusters of v1.23 and later support containerd.	OverlayFS	runC

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
Ubuntu 22.04	4.x	Docker Clusters of v1.23 and later support containerd.	OverlayFS	runC
Huawei Cloud EulerOS 1.1	3.x	Docker containerd	OverlayFS	runC
Huawei Cloud EulerOS 2.0	5.x	Docker containerd	OverlayFS	runC

Table 3-4 Node OSs and container engines in CCE Turbo clusters

Node Type	OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
ECS (VM)	CentOS 7.6	3.x	Docker containerd	OverlayFS	runC
	Ubuntu 18.04	4.x			
	EulerOS 2.9	4.x			
	Huawei Cloud EulerOS 1.1	3.x			
	Huawei Cloud EulerOS 2.0	5.x			
ECS (PM)	EulerOS 2.9	4.x	containerd	Device Mapper	Kata

Table 3-5 Kunpeng Node OSs and container engines in CCE

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
Huawei Cloud EulerOS 2.0	5.x	Docker containerd	OverlayFS	runC
EulerOS 2.9	4.x	Docker containerd	OverlayFS	runC
EulerOS 2.8	4.x	Docker	OverlayFS	runC

Common Commands of containerd and Docker

containerd does not support Docker APIs and Docker CLI, but you can run crictl commands to implement similar functions.

Table 3-6 Image-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List local images.	docker images	crictl images	ctr -n k8s.io i ls
Pull images.	docker pull	crictl pull	ctr -n k8s.io i pull
Push images.	docker push	None	ctr -n k8s.io i push
Delete a local image.	docker rmi	crictl rmi	ctr -n k8s.io i rm
Check images.	docker inspect	crictl inspecti	None

Table 3-7 Container-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List containers.	docker ps	crictl ps	ctr -n k8s.io c ls
Create a container.	docker create	crictl create	ctr -n k8s.io c create
Start a container.	docker start	crictl start	ctr -n k8s.io run
Stop a container.	docker stop	crictl stop	None
Delete a container.	docker rm	crictl rm	ctr -n k8s.io c del
Connect to a container.	docker attach	crictl attach	None
Access the container.	docker exec	crictl exec	None
Query container details.	docker inspect	crictl inspect	ctr -n k8s.io c info
View container logs.	docker logs	crictl logs	None

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
Check the resource usage of the container.	docker stats	crictl stats	None
Update container resource limits.	docker update	crictl update	None

Table 3-8 Pod-related commands

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
List pods.	None	crictl pods	None
View pod details.	None	crictl inspectp	None
Start a pod.	None	crictl start	None
Run a pod.	None	crictl runp	None
Stop a pod.	None	crictl stopp	None
Delete a pod.	None	crictl rmp	None

NOTE

Containers created and started by containerd are immediately deleted by kubelet. containerd does not support suspending, resuming, restarting, renaming, and waiting for containers, nor Docker image build, import, export, comparison, push, search, and labeling. containerd does not support file copy. You can log in to the image repository by modifying the configuration file of containerd.

Differences in Tracing

- Docker (Kubernetes v1.23 and earlier versions):

  kubelet --> dockershim (in the kubelet process) --> docker --> containerd
- Docker (community solution for Kubernetes v1.24 or later):

  kubelet --> cri-dockerd (kubelet uses CRI to connect to cri-dockerd) --> docker--> containerd
- containerd:

  kubelet --> CRI plugin (in the containerd process) --> containerd

Although Docker has added functions such as swarm cluster, docker build, and Docker APIs, it also introduces bugs. Compared with containerd, Docker has one more layer of calling. **Therefore, containerd is more resource-saving and secure.**

Container Engine Versions

NOTE

Table 3-9 lists only the runtime versions for the OSs in the latest cluster versions. For historical cluster versions, log in to the target node and check their runtime versions.

- Docker
```
# Huawei Cloud EulerOS, EulerOS, and CentOS
rpm -q docker-engine
# Ubuntu
dpkg -s docker-ce
```
- containerd
```
# Huawei Cloud EulerOS, EulerOS, and CentOS
rpm -q containerd
# Ubuntu
dpkg -s containerd
```

Table 3-9 Latest runtime versions for each OS

Cluster Version	Architecture	OS	Latest Docker Version	Latest Docker Version
v1.30	x86	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.x86_64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.x86_64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.x86_64	
		Huawei Cloud EulerOS 1.1	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		CentOS 7.6	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		Ubuntu 22.04	5:27.3.1-1, Ubuntu 22.04, Jammy	1.7.16-24.11.ubuntu.96477737.git187568c8
	Arm	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.aarch64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.aarch64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.aarch64	

Cluster Version	Architecture	OS	Latest Docker Version	Latest Docker Version
v1.29	x86	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.x86_64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.x86_64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.x86_64	
		Huawei Cloud EulerOS 1.1	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		CentOS 7.6	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		Ubuntu 22.04	5:27.3.1-1, Ubuntu 22.04, Jammy	1.7.16-24.11.ubuntu.96477737.git187568c8
	Arm	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.aarch64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.aarch64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.aarch64	
v1.28	x86	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.x86_64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.x86_64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.x86_64	
		Huawei Cloud EulerOS 1.1	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		CentOS 7.6	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	

Cluster Version	Architecture	OS	Latest Docker Version	Latest Docker Version
		Ubuntu 22.04	5:27.3.1-1, Ubuntu 22.04, Jammy	1.7.16-24.11.ubuntu.96477737.git187568c8
	Arm	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.aarch64	containerd-1.7.16-24.11.eulerosv2r9.96465557.git6dc92bbb.aarch64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.aarch64	
v1.27	x86	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.x86_64	containerd-1.6.14-24.08.eulerosv2r9.89023346.git7ed2b28f.x86_64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.x86_64	
		Huawei Cloud EulerOS 1.1	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		CentOS 7.6	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		Ubuntu 22.04	5:27.3.1-1, Ubuntu 22.04, Jammy	1.6.14-24.08.ubuntu.89022072.git7ed2b28f
	Arm	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.aarch64	containerd-1.6.14-24.08.eulerosv2r9.89023346.git7ed2b28f.aarch64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.aarch64	
v1.25	x86	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.x86_64	containerd-1.6.14-24.08.eulerosv2r9.89023346.git7ed2b28f.x86_64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.x86_64	

Cluster Version	Architecture	OS	Latest Docker Version	Latest Docker Version
		Huawei Cloud EulerOS 1.1	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		CentOS 7.6	docker-engine-18.09.0.101-1.h70.28.24.eulerosv2r7.x86_64	
		Ubuntu 22.04	5:27.3.1-1, Ubuntu 22.04, Jammy	1.6.14-24.08.ubuntu.89022072.git7ed2b28f
		Ubuntu 18.04	5:18.09.9~3-0~ubuntu-bionic	
	Arm	Huawei Cloud EulerOS 2.0	docker-engine-18.09.0-311.r50.46.43.hce2.aarch64	containerd-1.6.14-24.08.eulerosv2r9.89023346.git7ed2b28f.aarch64
		EulerOS 2.9	docker-engine-18.09.0.129-1.h91.43.27.eulerosv2r9.aarch64	
		EulerOS 2.8	docker-engine-18.09.0.101-1.h69.28.24.eulerosv2r8.aarch64	

3.3 Node OSs

This section describes the mappings between released cluster versions and OS versions.

ECS (VM)

Table 3-10 ECS (VM) OSs

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
Huawei Cloud EulerOS 2.0	v1.31	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.x86_64
	v1.30	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.x86_64
	v1.29	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.x86_64
	v1.28	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.x86_64
	v1.27	√	Supported in v1.27.3-r0 or later.	√	5.10.0-182.0.0.95.r2220_156.hce2.x86_64
	v1.25	√	Supported in v1.25.6-r0 or later.	√	• Clusters of v1.25.3-r0 or later 5.10.0-182.0.0.95.r2220_156.hce2.x86_64 • Clusters of versions earlier than v1.25.3-r0 5.10.0-60.18.0.50.r865_35.hce2.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.23 (end of maintenance)	√	Supported in v1.23.11-r0 or later.	√	• Clusters of v1.23.8-r0 or later 5.10.0-182.0.0.95.r2220_156.hce2.aarch64 • Clusters of versions earlier than v1.23.8-r0 5.10.0-60.18.0.50.r865_35.hce2.x86_64
Huawei Cloud EulerOS 2.0 (Arm)	v1.31	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.aarch64
	v1.30	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.aarch64
	v1.29	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.aarch64
	v1.28	√	√	√	5.10.0-182.0.0.95.r2220_156.hce2.aarch64
	v1.27	√	Supported in v1.27.3-r0 or later.	√	5.10.0-182.0.0.95.r2220_156.hce2.aarch64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.25	√	Supported in v1.25.6-r0 or later.	√	• Clusters of v1.25.3-r0 or later 5.10.0-182.0.0.95.r2220_156.hce2.aarch64 • Clusters of versions earlier than v1.25.3-r0 5.10.0-60.18.0.50.r865_35.hce2.aarch64
	v1.23 (end of maintenance)	√	Supported in v1.23.11-r0 or later.	√	• Clusters of v1.23.8-r0 or later 5.10.0-182.0.0.95.r2220_156.hce2.aarch64 • Clusters of versions earlier than v1.23.8-r0 5.10.0-60.18.0.50.r865_35.hce2.aarch64
Ubuntu 22.04	v1.31	√	x	√	5.15.0-126-generic
	v1.30	√	x	√	5.15.0-126-generic
	v1.29	√	x	√	5.15.0-126-generic
	v1.28	√	x	√	5.15.0-126-generic

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.27	√	x	√	• Clusters of v1.27.3-r0 or later 5.15.0-126-generic • Clusters of versions earlier than v1.27.3-r0 5.15.0-86-generic
	v1.25	√	x	√	• Clusters of v1.25.6-r0 or later 5.15.0-126-generic • Clusters of versions earlier than v1.25.6-r0 5.15.0-86-generic
	v1.23 (end of maintenance)	√	x	√	• Clusters of v1.23.11-r0 or later 5.15.0-126-generic • Clusters of versions earlier than v1.23.11-r0 5.15.0-86-generic
Huawei Cloud EulerOS 1.1	v1.31	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.30	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.29	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.28	√	√	√	3.10.0-1160.76.2.hce1c.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.27	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.25	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.23 (end of maintenance)	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.21 (end of maintenance)	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
CentOS Linux release 7.6	v1.31	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.30	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.29	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.28	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.27	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.25	√	√	√	3.10.0-1160.119.1.el7.x86 64
	v1.23 (end of maintenance)	√	√	√	• Clusters of v1.23.3-r0 or later 3.10.0-1160.119.1.el7.x86 64 • Clusters of versions earlier than v1.23.3-r0 3.10.0-1160.66.1.el7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.21 (end of maintenance)	√	√	√	• Clusters of v1.21.5-r0 or later 3.10.0-1160.119.1.el7.x86_64 • v1.21.4-r0 clusters: 3.10.0-1160.66.1.el7.x86_64 • Clusters of versions earlier than v1.21.4-r0 3.10.0-1160.25.1.el7.x86_64
	v1.19 (end of maintenance)	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.17.17 (end of maintenance)	√	√	√	3.10.0-1160.15.2.el7.x86_64
	v1.17.9 (end of maintenance)	√	√	√	3.10.0-1062.12.1.el7.x86_64
	v1.15.11 (end of maintenance)	√	√	√	3.10.0-1062.12.1.el7.x86_64
	v1.15.6-r1 (end of maintenance)	√	√	√	3.10.0-1062.1.1.el7.x86_64
	v1.13.10-r1 (end of maintenance)	√	√	√	3.10.0-957.21.3.el7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.13.7-r0 (end of maintenance)	√	√	√	3.10.0-957.21.3.el7.x86_64
EulerOS release 2.9	v1.31	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.30	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.29	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.28	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.27	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.25	√	√	√	4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64
	v1.23 (end of maintenance)	√	√	√	• Clusters of v1.23.5-r0 or later 4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64 • Clusters of versions earlier than v1.23.5-r0 4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.21 (end of maintenance)	√	√	√	• Clusters of v1.21.7-r0 or later 4.18.0-147.5.1.6.h1447.eulerosv2r9.x86_64 • Clusters of versions earlier than v1.21.7-r0 4.18.0-147.5.1.6.h1017.eulerosv2r9.x86_64
	v1.19 (end of maintenance)	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
EulerOS release 2.9 (Arm)	v1.31	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64
	v1.30	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64
	v1.29	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64
	v1.28	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64
	v1.27	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.25	√	√	√	4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64
	v1.23 (end of maintenance)	√	√	√	• Clusters of v1.23.5-r0 or later 4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64 • Clusters of versions earlier than v1.23.5-r0 4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64
	v1.21 (end of maintenance)	√	√	√	• Clusters of v1.21.7-r0 or later 4.19.90-vhulk2103.1.0.h1380.eulerosv2r9.aarch64 • Clusters of versions earlier than v1.21.7-r0 4.19.90-vhulk2103.1.0.h990.eulerosv2r9.aarch64
	v1.19 (end of maintenance)	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
EulerOS release 2.8 (Arm, end of maintenance)	v1.27 or later	x	x	x	None
	v1.25 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.23 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.21 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.16 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.10 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h962.eulerosv2r8.aarch64
	v1.17.17 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h962.eulerosv2r8.aarch64
	v1.15.11 (end of maintenance)	√	√	√	4.19.36-vhulk1907.1.0.h702.eulerosv2r8.aarch64
EulerOS release 2.5 (end of maintenance)	v1.27 or later	x	x	x	None
	v1.25 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.23 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.21 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.19.16 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.19.10 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64
	v1.19.8 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64
	v1.17.17 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h470.eulerosv2r7.x86_64
	v1.17.9 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h428.eulerosv2r7.x86_64
	v1.15.11 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h428.eulerosv2r7.x86_64
	v1.15.6-r1 (end of maintenance)	√	√	√	3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.13.10-r1 (end of maintenance)	√	√	√	3.10.0-862.14.1.2.h249.eulerosv2r7.x86_64
	v1.13.7-r0 (end of maintenance)	√	√	√	3.10.0-862.14.1.0.h197.eulerosv2r7.x86_64
Ubuntu 18.04 server 64-bit (end of maintenance)	v1.27 or later	x	x	x	None
	v1.25 (end of maintenance)	√	x	√	4.15.0-171-generic
	v1.23 (end of maintenance)	√	x	√	4.15.0-171-generic
	v1.21 (end of maintenance)	√	x	√	4.15.0-171-generic
	v1.19.16 (end of maintenance)	√	x	√	4.15.0-171-generic
	v1.19.8 (end of maintenance)	√	x	√	4.15.0-136-generic
	v1.17.17 (end of maintenance)	√	x	√	4.15.0-136-generic

ECS (PM)

Table 3-11 ECS (PM) OSs

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
EulerOS release 2.10	v1.31	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.30	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.29	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.28	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.27	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.25	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.23	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.21 (end of maintenance)	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64
	v1.19.16 (end of maintenance)	√	√	√	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64

BMS

For details about the specifications of the server, see **Instance Family**.

Table 3-12 BMS OSs

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
Huawei Cloud EulerOS 2.0 (supported by certain regions and server models. For details, see the console.)	v1.31	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.30	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.29	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.28	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.27	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.25	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.23	√	√	x	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
EulerOS release 2.9 (restricted use. Submit a service ticket for confirmation.)	v1.31	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.30	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.29	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.28	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.27	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.25	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.23	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.21 (end of maintenance)	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.19 (end of maintenance)	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
EulerOS release 2.3 (end of maintenance)	v1.27 or later	x	x	x	None
	v1.25 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.23 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.21 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.19 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native Network 2.0	
	v1.17 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.15.11 (end of maintenance)	√	√	x	3.10.0-514.41.4.28.h62.x86_64

3.4 Node Specifications

This section describes the node specifications supported by CCE and typical node features.

NOTE

Node flavors vary depending on regions, and some flavors may be unavailable due to insufficient resources. To obtain available node flavors, log in to the CCE console and switch to the page for creating nodes.

Node Type	Description	Specifications
ECS (VM)	ECS running on a KVM or QingTian hypervisor. Multiple flavors are offered to meet your compute and storage needs for various applications.	x86: • **General Computing-plus ECSs** • **General Computing ECSs** • **Memory-optimized ECSs** • **General Computing-Basic ECSs** • **Disk-intensive ECSs** • **Ultra-high I/O ECSs** • **FlexusX ECSs** Arm (Kunpeng): • **Kunpeng General Computing-plus ECSs** • **Kunpeng Memory-optimized ECSs** • **Kunpeng Ultra-high I/O ECSs** Heterogeneous: • **GPU-accelerated ECSs** • **AI-accelerated ECSs**
ECS (PM)	ECS running on a QingTian-backed bare-metal hypervisor. PM resources and VM resources of such an ECS are in the same resource pool and can be dynamically scheduled.	**General Computing-plus ECSs**
BMS	BMS offering high-performance computing at low latency for containers.	**Instance Family**

General Computing-plus ECSs

General computing-plus ECSs use dedicated vCPUs and next-generation network acceleration engines to provide powerful compute and network performance.

Table 3-13 General computing-plus ECS features

ECS Type	Compute	Network	Supported Cluster Type
aC7	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 232 • Base/Turbo frequency: 2.45 GHz/3.5 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 20,000,000 • Maximum intranet bandwidth: 100 Gbit/s	CCE standard cluster CCE Turbo cluster
C7	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 128 • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.5 GHz and 2.8 GHz/3.5 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 12,000,000 • Maximum intranet bandwidth: 42 Gbit/s	CCE standard cluster CCE Turbo cluster
C6ne	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 64 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 10,000,000 • Maximum intranet bandwidth: 40 Gbit/s	CCE Turbo cluster
C6s	• vCPU to memory ratio: 1:2 • Number of vCPUs: 2 to 64 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 8,500,000 • Maximum intranet bandwidth: 30 Gbit/s	CCE standard cluster

ECS Type	Compute	Network	Supported Cluster Type
C6	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 88 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 12,000,000 • Maximum intranet bandwidth: 44 Gbit/s	CCE standard cluster
C3	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 60 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 5,000,000 • Maximum intranet bandwidth: 16 Gbit/s	CCE standard cluster

Table 3-14 aC7 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Virtualization
ac7.large.2	2	4	2/1	40	2	2	16	KVM
ac7.xlarge.2	4	8	3/1.5	60	2	3	32	
ac7.2xlarge.2	8	16	4/2.5	100	4	4	64	
ac7.3xlarge.2	12	24	6/4	150	4	6	96	
ac7.4xlarge.2	16	32	8/5	200	8	8	128	

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Virtualization
ac7.6xlarge.2	24	48	12/6	250	8	8	192	
ac7.8xlarge.2	32	64	15/8	300	16	8	256	
ac7.12xlarge.2	48	96	22/12	400	16	8	256	
ac7.16xlarge.2	64	128	28/16	550	24	12	256	
ac7.24xlarge.2	96	192	40/25	800	24	12	256	
ac7.29xlarge.2	116	216	50/30	950	32	16	256	
ac7.32xlarge.2	128	256	55/35	1,000	32	16	256	
ac7.48xlarge.2	192	384	100/80	1,600	32	16	256	
ac7.58xlarge.2	232	432	120/100	2,000	32	16	256	
ac7.large.4	2	8	2/1	40	2	2	16	
ac7.xlarge.4	4	16	3/1.5	60	2	3	32	
ac7.2xlarge.4	8	32	4/2.5	100	4	4	64	
ac7.3xlarge.4	12	48	6/4	150	4	6	96	
ac7.4xlarge.4	16	64	8/5	200	8	8	128	
ac7.6xlarge.4	24	96	12/6	250	8	8	192	
ac7.8xlarge.4	32	128	15/8	300	16	8	256	
ac7.12xlarge.4	48	192	22/12	400	16	8	256	

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Virtualization
ac7.16xlarge.4	64	256	28/16	550	24	12	256	
ac7.24xlarge.4	96	384	40/25	800	24	12	256	
ac7.29xlarge.4	116	464	50/30	950	32	16	256	
ac7.32xlarge.4	128	512	55/35	1,000	32	16	256	
ac7.48xlarge.4	192	768	100/80	1,600	32	16	256	
ac7.58xlarge.4	232	928	120/100	2,000	32	16	256	

Table 3-15 C7 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	EVS Basic Bandwidth/ Burst Bandwidth (Gbit/s)	Virtualization
c7.large.2	2	4	4/0.64	40	50	2	2	16	1/6	QingTian
c7.xlarge.2	4	8	8/1.28	80	50	2	3	32	1.5/6	
c7.2xlarge.2	8	16	15/2.4	150	100	4	4	64	2/6	
c7.3xlarge.2	12	24	17/4	200	150	4	6	96	3/6	

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	EVS Basic Bandwidth/ Burst Bandwidth (Gbit/s)	Virtualization
c7.4xlarge.2	16	32	20/4.8	280	150	8	8	128	4/6	
c7.6xlarge.2	24	48	25/7.2	400	200	8	8	192	5/6	
c7.8xlarge.2	32	64	30/9.6	550	300	16	8	256	6/ None	
c7.12xlarge.2	48	96	35/14.4	750	400	16	8	256	8/ None	
c7.16xlarge.2	64	128	36/19.2	1,000	500	28	8	256	12/ None	
c7.24xlarge.2	96	192	40/28.8	1,100	800	32	8	256	16/ None	
c7.32xlarge.2	128	256	42/32	1,200	1,000	32	8	256	24/ None	
c7.large.4	2	8	4/0.64	40	50	2	2	16	1/6	
c7.xlarge.4	4	16	8/1.28	80	50	2	3	32	1.5/6	
c7.2xlarge.4	8	32	15/2.4	150	100	4	4	64	2/6	
c7.3xlarge.4	12	48	17/4	200	150	4	6	96	3/6	
c7.4xlarge.4	16	64	20/4.8	280	150	8	8	128	4/6	

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	EVS Basic Bandwidth/ Burst Bandwidth (Gbit/s)	Virtualization
c7.6xlarge.4	24	96	25/7.2	400	200	8	8	192	5/6	
c7.8xlarge.4	32	128	30/9.6	550	300	16	8	256	6/ None	
c7.12xlarge.4	48	192	35/14.4	750	400	16	8	256	8/ None	
c7.16xlarge.4	64	256	36/19.2	1,000	500	28	8	256	12/ None	
c7.24xlarge.4	96	384	40/28.8	1,100	800	32	8	256	16/ None	
c7.32xlarge.4	128	512	42/32	1,200	1,000	32	8	256	24/ None	

Table 3-16 C6ne ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
c6ne.large.2	2	4	4/1.2	40	2	2	16
c6ne.large.4	2	8	4/1.2	40	2	2	16

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
c6ne.xlarge.2	4	8	8/2.4	80	2	3	32
c6ne.xlarge.4	4	16	8/2.4	80	2	3	32
c6ne.2xlarge.2	8	16	15/4.5	150	4	4	64
c6ne.2xlarge.4	8	32	15/4.5	150	4	4	64
c6ne.3xlarge.2	12	24	17/7	200	4	6	96
c6ne.3xlarge.4	12	48	17/7	200	4	6	96
c6ne.4xlarge.2	16	32	20/9	280	8	8	128
c6ne.4xlarge.4	16	64	20/9	280	8	8	128
c6ne.6xlarge.2	24	48	25/14	400	8	8	192
c6ne.6xlarge.4	24	96	25/14	400	8	8	192
c6ne.8xlarge.2	32	64	30/18	550	16	8	256
c6ne.8xlarge.4	32	128	30/18	550	16	8	256
c6ne.12xlarge.2	48	96	35/27	750	16	8	256
c6ne.12xlarge.4	48	192	35/27	750	16	8	256
c6ne.16xlarge.2	64	128	40/36	1,000	32	8	256
c6ne.16xlarge.4	64	256	40/36	1,000	32	8	256

Table 3-17 C6s ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Virtualization
c6s.large.2	2	4	1/1	30	50	2	2	KVM
c6s.xlarge.2	4	8	2/2	60	50	2	3	KVM
c6s.2xlarge.2	8	16	4/4	120	100	4	4	KVM
c6s.3xlarge.2	12	24	5.5/5.5	180	150	4	6	KVM
c6s.4xlarge.2	16	32	7.5/7.5	240	150	8	8	KVM
c6s.6xlarge.2	24	48	11/11	350	200	8	8	KVM
c6s.8xlarge.2	32	64	15/15	450	300	16	8	KVM
c6s.12xlarge.2	48	96	22/22	650	400	16	8	KVM
c6s.16xlarge.2	64	128	30/30	850	500	32	8	KVM
c6s.large.4	2	8	1/1	30	50	2	2	KVM
c6s.xlarge.4	4	16	2/2	60	50	2	3	KVM
c6s.2xlarge.4	8	32	4/4	120	100	4	4	KVM
c6s.3xlarge.4	12	48	5.5/5.5	180	150	4	6	KVM
c6s.4xlarge.4	16	64	7.5/7.5	240	150	8	8	KVM
c6s.6xlarge.4	24	96	11/11	350	200	8	8	KVM

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Virtualization
c6s.8xlarge.4	32	128	15/15	450	300	16	8	KVM
c6s.12xlarge.4	48	192	22/22	650	400	16	8	KVM
c6s.16xlarge.4	64	256	30/30	850	500	32	8	KVM

Table 3-18 C3 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	EVS Basic Bandwidth (Gbit/s)	Virtualization
c3.large.2	2	4	1.5/0.6	30	2	1	KVM
c3.xlarge.2	4	8	3/1	50	2	1.5	KVM
c3.2xlarge.2	8	16	5/2	90	4	2	KVM
c3.3xlarge.2	12	24	7/3	110	4	2.5	KVM
c3.4xlarge.2	16	32	10/4	130	4	3	KVM
c3.6xlarge.2	24	48	12/6	200	8	3.5	KVM
c3.8xlarge.2	32	64	15/8	260	8	4	KVM
c3.15xlarge.2	60	128	16/16	500	16	8	KVM

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	EVS Basic Bandwidth (Gbit/s)	Virtualization
c3.large.4	2	8	1.5/0.6	30	2	1	KVM
c3.xlarge.4	4	16	3/1	50	2	1.5	KVM
c3.2xlarge.4	8	32	5/2	90	4	2	KVM
c3.3xlarge.4	12	48	7/3	110	4	2.5	KVM
c3.4xlarge.4	16	64	10/4	130	4	3	KVM
c3.6xlarge.4	24	96	12/6	200	8	3.5	KVM
c3.8xlarge.4	32	128	15/8	260	8	4	KVM
c3.15xlarge.4	60	256	16/16	500	16	8	KVM

General Computing ECSs

General computing ECSs provide a balance of compute, memory, and networking resources and a baseline level of vCPU performance with the ability to burst above the baseline.

Table 3-19 General computing ECS features

ECS Type	Compute	Network	Supported Cluster Type
S7	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 8 • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.8 GHz/3.5 GHz	• IPv6 access • Higher compute specifications for better network performance • Maximum PPS: 500,000 • Maximum intranet bandwidth: 3 Gbit/s	CCE standard cluster CCE Turbo cluster

ECS Type	Compute	Network	Supported Cluster Type
S6	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 8 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• IPv6 access • Higher compute specifications for better network performance • Maximum PPS: 500,000 • Maximum intranet bandwidth: 3 Gbit/s	CCE standard cluster
Sn3	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 16 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.2 GHz/3.0 GHz	• IPv6 access • Higher compute specifications for better network performance • Maximum PPS: 500,000 • Maximum intranet bandwidth: 3 Gbit/s	CCE standard cluster
S3	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 16 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.2 GHz/3.0 GHz	• Higher compute specifications for better network performance • Maximum PPS: 300,000 • Maximum intranet bandwidth: 4 Gbit/s	CCE standard cluster
S2	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 32 • Intel® Xeon® Processor E5 v4 Family • Base/Turbo frequency: 2.4 GHz/3.3 GHz	• Higher compute specifications for better network performance • Maximum PPS: 500,000 • Maximum intranet bandwidth: 6 Gbit/s	CCE standard cluster

Table 3-20 S7 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs	Virtualization
s7.large.2	2	4	1.5/0.2	15	1	2	8	KVM
s7.xlarge.2	4	8	2/0.35	25	1	2	16	KVM
s7.2xlarge.2	8	16	3/0.75	50	2	2	32	KVM
s7.large.4	2	8	1.5/0.2	15	1	2	8	KVM
s7.xlarge.4	4	16	2/0.35	25	1	2	16	KVM
s7.2xlarge.4	8	32	3/0.75	50	2	2	32	KVM

Table 3-21 S6 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Virtualization
s6.large.2	2	4	1.5/0.2	15	1	2	KVM
s6.xlarge.2	4	8	2/0.35	25	1	2	KVM
s6.2xlarge.2	8	16	3/0.75	50	2	2	KVM
s6.large.4	2	8	1.5/0.2	15	1	2	KVM
s6.xlarge.4	4	16	2/0.35	25	1	2	KVM
s6.2xlarge.4	8	32	3/0.75	50	2	2	KVM

Table 3-22 Sn3 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Virtualization
sn3.large.2	2	4	1.5/0.35	15	1	2	KVM
sn3.xlarge.2	4	8	2/0.7	25	1	2	KVM
sn3.2xlarge.2	8	16	3/1.3	50	2	2	KVM
sn3.4xlarge.2	16	32	6/2.5	100	4	2	KVM
sn3.large.4	2	8	1.5/0.35	15	1	2	KVM
sn3.xlarge.4	4	16	2/0.7	25	1	2	KVM
sn3.2xlarge.4	8	32	3/1.3	50	2	2	KVM
sn3.4xlarge.4	16	64	6/2.5	100	4	2	KVM

Table 3-23 S3 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Virtualization
s3.large.2	2	4	0.8/0.2	10	1	KVM
s3.xlarge.2	4	8	1.5/0.4	15	1	KVM
s3.2xlarge.2	8	16	3/0.8	20	2	KVM
s3.4xlarge.2	16	32	4/1.5	30	4	KVM
s3.large.4	2	8	0.8/0.2	10	1	KVM
s3.xlarge.4	4	16	1.5/0.4	15	1	KVM

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Virtualization
s3.2xlarge.4	8	32	3/0.8	20	2	KVM
s3.4xlarge.4	16	64	4/1.5	30	4	KVM

Table 3-24 S2 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Virtualization
s2.large.2	2	4	0.8/0.2	10	1	KVM
s2.xlarge.2	4	8	1.5/0.4	15	1	KVM
s2.2xlarge.2	8	16	3/0.8	20	2	KVM
s2.4xlarge.2	16	32	4/1.5	30	4	KVM
s2.8xlarge.2	32	64	6/3	50	8	KVM
s2.large.4	2	8	0.8/0.2	10	1	KVM
s2.xlarge.4	4	16	1.5/0.4	15	1	KVM
s2.2xlarge.4	8	32	3/0.8	20	2	KVM
s2.4xlarge.4	16	64	4/1.5	30	4	KVM
s2.8xlarge.4	32	128	6/3	50	8	KVM

Memory-optimized ECSs

Memory-optimized ECSs have a large memory size and provide high memory performance. They are designed for memory-intensive applications that process a large amount of data.

Table 3-25 Memory-optimized ECS features

ECS Type	Compute	Network	Supported Cluster Type
M7	• vCPU to memory ratio: 1:8 • Number of vCPUs: 2 to 128 • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.5 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 12,000,000 • Maximum intranet bandwidth: 42 Gbit/s	CCE standard cluster CCE Turbo cluster
M6ne	• vCPU to memory ratio: 1:8 • Number of vCPUs: 2 to 64 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 10,000,000 • Maximum intranet bandwidth: 40 Gbit/s	CCE Turbo cluster
M6	• vCPU to memory ratio: 1:8 • Number of vCPUs: 2 to 64 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 10,000,000 • Maximum intranet bandwidth: 40 Gbit/s	CCE standard cluster
M3	• vCPU to memory ratio: 1:8 • Number of vCPUs: 2 to 60 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 5,000,000 • Maximum intranet bandwidth: 17 Gbit/s	CCE standard cluster

Table 3-26 M6ne ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
m6ne.large.8	2	16	4/1.2	40	2	2	16
m6ne.xlarge.8	4	32	8/2.4	80	2	3	32
m6ne.2xlarge.8	8	64	15/4.5	150	4	4	64
m6ne.3xlarge.8	12	96	17/7	200	4	6	96
m6ne.4xlarge.8	16	128	20/9	280	8	8	128
m6ne.6xlarge.8	24	192	25/14	400	8	8	192
m6ne.8xlarge.8	32	256	30/18	550	16	8	256
m6ne.16xlarge.8	64	512	40/36	1000	32	8	256

Table 3-27 M6 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	EVS Basic Bandwidth/ Burst Bandwidth (Gbit/s)	Virtualization
m6.large.8	2	16	4/1.2	40	50	2	2	1/5	KVM
m6.xlarge.8	4	32	8/2.4	80	50	2	3	1.5/5	KVM
m6.2xlarge.8	8	64	15/4.5	150	100	4	4	2/5	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	EVS Basic Bandwidth/Burst Bandwidth (Gbit/s)	Virtualization
m6.3xlarge.8	12	96	17/7	200	150	4	6	2.5/5	KVM
m6.4xlarge.8	16	128	20/9	280	150	8	8	3.5/5	KVM
m6.6xlarge.8	24	192	25/14	400	200	8	8	4/5	KVM
m6.8xlarge.8	32	256	30/18	550	300	16	8	7/10	KVM
m6.12xlarge.8	48	384	35/27	750	400	16	8	10/15	KVM
m6.16xlarge.8	64	512	40/36	1,000	500	32	8	20/None	KVM
m6.22xlarge.8.physical	88	768	40/40	1,000	1,000	16	33	20/None	Bare metal

Table 3-28 M3 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	EVS Basic Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	EVS Basic Bandwidth (Gbit/s)	Virtualization
m3.large.8	2	16	1.5/0.6	1	30	2	1	KVM
m3.xlarge.8	4	32	3/1.1	1.5	50	2	1.5	KVM
m3.2xlarge.8	8	64	5/2	2	90	4	2	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	EVS Basic Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	EVS Basic Bandwidth (Gbit/s)	Virtualization
m3.3xlarge.8	12	96	8/3.5	2.5	110	4	2.5	KVM
m3.4xlarge.8	16	128	10/4.5	3	130	4	3	KVM
m3.6xlarge.8	24	192	12/6.5	3.5	200	8	3.5	KVM
m3.8xlarge.8	32	256	15/9	4	260	8	4	KVM
m3.15xlarge.8	60	512	17/17	8	500	16	8	KVM

General Computing-Basic ECSs

General computing-basic ECSs are suitable for scenarios that require moderate CPU performance generally but burstable high performance occasionally while keeping costs low. The performance of such ECSs is constrained by the baseline performance and CPU credits. There are no additional charges for using CPU credits. For details, see **CPU Credits**.

Table 3-29 General computing-basic ECS features

ECS Type	Compute	Network	Supported Cluster Type
T6	• vCPU to memory ratio: 1:1, 1:2, or 1:4 • Number of vCPUs: 2 to 16 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.2 GHz/3.0 GHz	• Higher compute specifications for better network performance • Maximum PPS: 600,000 • Maximum intranet bandwidth: 3 Gbit/s	CCE standard cluster

Table 3-30 T6 ECS specifications

Flavor	vCPUs	Memory (GiB)	CPU Baseline (%)	Average CPU Baseline (%)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NICs	Virtualization
t6.xlarge.1	4	4	80	20	1/0.2	20	2	KVM
t6.2xlarge.1	8	8	120	15	2/0.4	40	2	KVM
t6.4xlarge.1	16	16	240	15	3/0.8	60	2	KVM
t6.large.2	2	4	40	20	0.5/0.1	10	1	KVM
t6.xlarge.2	4	8	80	20	1/0.2	20	2	KVM
t6.2xlarge.2	8	16	120	15	2/0.4	40	2	KVM
t6.4xlarge.2	16	32	240	15	3/0.8	60	2	KVM
t6.large.4	2	8	40	20	0.5/0.1	10	1	KVM
t6.xlarge.4	4	16	80	20	1/0.2	20	2	KVM
t6.2xlarge.4	8	32	120	15	2/0.4	40	2	KVM

GPU-accelerated ECSs

GPU-accelerated ECSs provide outstanding floating-point computing capabilities. They are suitable for applications that require real-time, highly concurrent massive computing.

GPU-accelerated ECSs are classified as G series and P series ECSs.

- G series: Graphics-accelerated ECSs, which are suitable for 3D animation rendering and CAD
- P series: Computing-accelerated or inference-accelerated ECSs, which are suitable for deep learning, scientific computing, and CAE

Table 3-31 GPU-accelerated ECSs

Classification	ECS Type	Graphics Card	CUDA Cores per GPU	Single-GPU Performance	Application	Supported Cluster Type
Graphics-accelerated	G6ne	NVIDIA T4 (GPU passthrough)	2560	• 8.1 TFLOPS of single-precision floating-point computing • 130 INT8 TOPS • 260 INT4 TOPS	Cloud desktop, image rendering, 3D visualization, and heavy-load graphics design	CCE Turbo cluster
Graphics-accelerated	G6	NVIDIA T4 (GPU passthrough)	2560	• 8.1 TFLOPS of single-precision floating-point computing • 130 INT8 TOPS • 260 INT4 TOPS	Cloud desktop, image rendering, 3D visualization, and heavy-load graphics design	CCE standard cluster
Graphics-accelerated	G5	NVIDIA V100 (GPU passthrough)	5120	• 14 TFLOPS of single-precision floating-point computing • 7 TFLOPS of double-precision floating-point computing • 112 TFLOPS Tensor Cores for deep learning acceleration	Cloud desktop, image rendering, 3D visualization, and heavy-load graphics design	CCE standard cluster

Classification	ECS Type	Graphics Card	CUDA Cores per GPU	Single-GPU Performance	Application	Supported Cluster Type
Computing-accelerated	P2s	NVIDIA V100	5120	• 14 TFLOPS of single-precision floating-point computing • 7 TFLOPS of double-precision floating-point computing • 112 TFLOPS Tensor Cores for deep learning acceleration	AI deep learning training, scientific computing, computational fluid dynamics, computational finance, seismic analysis, molecular modeling, and genomics	CCE standard cluster
Computing-accelerated	P2sne	NVIDIA V100	5120	• 14 TFLOPS of single-precision floating-point computing • 7 TFLOPS of double-precision floating-point computing • 112 TFLOPS Tensor Cores for deep learning acceleration	AI deep learning training, scientific computing, computational fluid dynamics, computational finance, seismic analysis, molecular modeling, and genomics	CCE Turbo cluster

Classification	ECS Type	Graphics Card	CUDA Cores per GPU	Single-GPU Performance	Application	Supported Cluster Type
Computing-accelerated	P2v	NVIDIA V100 NVLink (GPU passthrough)	5120	• 15.7 TFLOPS of single-precision floating-point computing • 7.8 TFLOPS of double-precision floating-point computing • 125 TFLOPS Tensor Cores for deep learning acceleration • 300 GiB/s NVLink	Machine learning, deep learning, inference training, scientific computing, seismic analysis, computational finance, rendering, multimedia encoding and decoding	CCE standard cluster
Inference-accelerated	Pi2ne	NVIDIA T4 (GPU passthrough)	2560	• 8.1 TFLOPS of single-precision floating-point computing • 130 INT8 TOPS • 260 INT4 TOPS	Machine learning, deep learning, inference training, scientific computing, seismic analysis, computational finance, rendering, multimedia encoding and decoding	CCE Turbo cluster

Classification	ECS Type	Graphics Card	CUDA Cores per GPU	Single-GPU Performance	Application	Supported Cluster Type
Inference-accelerated	Pi2	NVIDIA T4 (GPU passthrough)	2560	• 8.1 TFLOPS of single-precision floating-point computing • 130 INT8 TOPS • 260 INT4 TOPS	Machine learning, deep learning, inference training, scientific computing, seismic analysis, computational finance, rendering, multimedia encoding and decoding	CCE standard cluster
Inference-accelerated	Pi1	NVIDIA P4 (GPU passthrough)	2560	5.5 TFLOPS of single-precision floating-point computing	Machine learning, deep learning, inference training, scientific computing, seismic analysis, computational finance, rendering, multimedia encoding and decoding	CCE standard cluster

Table 3-32 P2v ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	GPUs	GPU Connection	GPU Memory (GiB)	Virtualization	Hardware
p2v.2xlarge.8	8	64	10/4	50	4	4	1 × V100	N/A	1 × 16 GiB	KVM	CPU: Intel® Xeon® Skylake-SP Gold 6151 v5
p2v.4xlarge.8	16	128	15/8	100	8	8	2 × V100	NVLink	2 × 16 GiB	KVM	
p2v.8xlarge.8	32	256	25/15	200	16	8	4 × V100	NVLink	4 × 16 GiB	KVM	
p2v.16xlarge.8	64	512	30/30	400	32	8	8 × V100	NVLink	8 × 16 GiB	KVM	

Disk-intensive ECSs

Disk-intensive ECSs are delivered with local disks for high storage bandwidth and IOPS. In addition, local disks are more cost-effective in massive data storage scenarios.

Table 3-33 Disk-intensive ECS features

ECS Type	Compute	Network	Supported Cluster Type
D7	• vCPU to memory ratio: 1:4 • Number of vCPUs: 4 to 72 • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 8,500,000 Maximum intranet bandwidth: 42 Gbit/s	CCE standard cluster CCE Turbo cluster

ECS Type	Compute	Network	Supported Cluster Type
D6	• vCPU to memory ratio: 1:4 • Number of vCPUs: 4 to 72 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 9,000,000 • Maximum intranet bandwidth: 44 Gbit/s	CCE standard cluster CCE Turbo cluster

Table 3-34 D7 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Local Disks (GiB)	Virtualization
d7.xlarge.4	4	16	5/1.7	60	50	2	3	32	2 × 3,600	KVM
d7.2xlarge.4	8	32	10/3.5	120	100	4	4	64	4 × 3,600	KVM
d7.4xlarge.4	16	64	20/6.7	240	150	4	6	96	8 × 3,600	KVM
d7.6xlarge.4	24	96	25/10	350	200	8	8	128	12 × 3,600	KVM
d7.8xlarge.4	32	128	30/13.5	450	300	8	8	192	16 × 3,600	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Local Disks (GiB)	Virtualization
d7.12xlarge.4	48	192	40/20	650	400	16	8	256	24 × 3,600	KVM
d7.16xlarge.4	64	256	42/27	850	500	16	8	256	32 × 3,600	KVM

Table 3-35 D6 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Local Disks (GiB)	Virtualization
d6.xlarge.4	4	16	5/2	60	50	2	3	2 × 3,600	KVM
d6.2xlarge.4	8	32	10/4	120	100	4	4	4 × 3,600	KVM
d6.4xlarge.4	16	64	20/7.5	240	150	8	8	8 × 3,600	KVM
d6.6xlarge.4	24	96	25/11	350	200	8	8	12 × 3,600	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Local Disks (GiB)	Virtualization
d6.8xlarge.4	32	128	30/15	450	300	16	8	16 × 3,600	KVM
d6.12xlarge.4	48	192	40/22	650	400	16	8	24 × 3,600	KVM
d6.16xlarge.4	64	256	42/30	850	500	32	8	32 × 3,600	KVM
d6.18xlarge.4	72	288	44/34	900	700	32	8	36 × 3,600	KVM

Kunpeng General Computing-plus ECSs

Kunpeng general computing-plus ECSs use Kunpeng processors to provide powerful compute and high-performance networks, meeting enterprise requirements for cost-effective, secure, reliable cloud services.

Table 3-36 Kunpeng general computing-plus ECS features

ECS Type	Compute	Network	Supported Cluster Type
kC1	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 60 • Kunpeng 920 processor • Base frequency: 2.6 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 4,000,000 • Maximum intranet bandwidth: 30 Gbit/s	CCE standard cluster
kC1n	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 8 to 48 • Kunpeng 920 processor • Base frequency: 2.6 GHz	• IPv6 access • Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 3,500,000 • Maximum intranet bandwidth: 25 Gbit/s	CCE standard cluster CCE Turbo cluster

Kunpeng Memory-optimized ECSs

Kunpeng memory-optimized ECSs use Kunpeng 920 processors and 25GE high-speed intelligent NICs to provide up to 480 GiB DDR4-based memory with high network performance for large in-memory datasets.

Table 3-37 Kunpeng memory-optimized ECS features

ECS Type	Compute	Network	Supported Cluster Type
kM1	• vCPU to memory ratio: 1:8 • Number of vCPUs: 2 to 60 • Kunpeng 920 processor • Base frequency: 2.6 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 4,000,000 • Maximum intranet bandwidth: 30 Gbit/s	CCE standard cluster

Table 3-38 kM1 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Virtualization
km1.large.8	2	16	3/0.8	30	2	2	KVM
km1.xlarge.8	4	32	5/1.5	50	2	3	KVM
km1.2xlarge.8	8	64	7/3	80	4	4	KVM
km1.3xlarge.8	12	96	9/4.5	110	4	5	KVM
km1.4xlarge.8	16	128	12/6	140	4	6	KVM
km1.6xlarge.8	24	192	15/8	200	8	6	KVM
km1.8xlarge.8	32	256	18/10	260	8	6	KVM
km1.12xlarge.8	48	384	25/16	350	16	8	KVM
km1.15xlarge.8	60	480	30/20	400	16	8	KVM

Kunpeng Ultra-high I/O ECSs

Kunpeng ultra-high I/O ECSs use Kunpeng 920 processors and 25GE high-speed intelligent NICs to provide up to 480 GiB DDR4-based memory with high network performance for large in-memory datasets.

Table 3-39 Kunpeng ultra-high I/O ECS features

ECS Type	Compute	Network	Supported Cluster Type
kI1	• vCPU to memory ratio: 1:4 • Number of vCPUs: 8 to 64 • Kunpeng 920 processor • Base frequency: 2.6 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 4,000,000 • Maximum intranet bandwidth: 30 Gbit/s	CCE standard cluster

Table 3-40 kI1 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NICs	Max. NIC Queues	Local Disks	Virtualization
ki1.2xlarge.4	8	32	7/3	80	4	4	1 × 3,200 GiB	KVM
ki1.4xlarge.4	16	64	12/6	140	6	4	2 × 3,200 GiB	KVM
ki1.6xlarge.4	24	96	15/8.5	200	6	8	3 × 3,200 GiB	KVM
ki1.8xlarge.4	32	128	18/10	260	6	8	4 × 3,200 GiB	KVM
ki1.12xlarge.4	48	192	25/16	350	6	16	6 × 3,200 GiB	KVM
ki1.16xlarge.4	64	228	30/20	400	6	16	8 × 3,200 GiB	KVM

Ultra-high I/O ECSs

Ultra-high I/O ECSs use high-performance local NVMe SSDs to provide high storage IOPS and low read/write latency.

Table 3-41 Ultra-high I/O ECS features

ECS Type	Compute	Network	Supported Cluster Type
Ir7	• vCPU to memory ratio: 1:4 • Number of vCPUs: 2 to 64 • 3rd Generation Intel® Xeon® Platinum Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 6,000,000 • Maximum intranet bandwidth: 40 Gbit/s	CCE standard cluster CCE Turbo cluster
I7	• vCPU to memory ratio: 1:4 • Number of vCPUs: 8 to 96 • 3rd Generation Intel® Xeon® Platinum Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 8,000,000 • Maximum intranet bandwidth: 44 Gbit/s	CCE standard cluster CCE Turbo cluster
Ir3	• vCPU to memory ratio: 1:4 • Number of vCPUs: 2 to 32 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 5,500,000 • Maximum intranet bandwidth: 30 Gbit/s	CCE standard cluster
I3	• vCPU to memory ratio: 1:8 • Number of vCPUs: 8 to 64 • Intel® Xeon® Scalable Processor • Base/Turbo frequency: 3.0 GHz/3.4 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 5,000,000 • Maximum intranet bandwidth: 25 Gbit/s	CCE standard cluster

Table 3-42 I7 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Local Disks (GiB)	Virtualization
i7.2xlarge.4	8	32	10/3	120	100	4	4	64	1 × 1,600 GiB NVMe	KVM
i7.4xlarge.4	16	64	15/6	200	150	4	6	96	2 × 1,600 GiB NVMe	KVM
i7.8xlarge.4	32	128	25/12	400	300	8	8	192	4 × 1,600 GiB NVMe	KVM
i7.12xlarge.4	48	192	30/18	500	400	16	8	256	6 × 1,600 GiB NVMe	KVM
i7.16xlarge.4	64	256	35/24	600	500	16	8	256	8 × 1,600 GiB NVMe	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Local Disks (GiB)	Virtualization
i7.24xlarge.4	96	384	44/36	800	800	32	8	256	12 × 1,600 GiB NVMe	KVM
i7.2xlarge.8	8	64	10/3	120	100	4	4	64	1 × 1,600 GiB NVMe	KVM
i7.4xlarge.8	16	128	15/6	200	150	4	6	96	2 × 1,600 GiB NVMe	KVM
i7.8xlarge.8	32	256	25/12	400	300	8	8	192	4 × 1,600 GiB NVMe	KVM
i7.12xlarge.8	48	384	30/18	500	400	16	8	256	6 × 1,600 GiB NVMe	KVM
i7.16xlarge.8	64	512	35/24	600	500	16	8	256	8 × 1,600 GiB NVMe	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth (Gbit/s)	Max. PPS (10,000)	Network Connections (10,000)	Max. NIC Queues	Max. NICs	Max. Supplementary NICs	Local Disks (GiB)	Virtualization
i7.24xlarge.8	96	768	44/36	800	800	32	8	256	12 × 1,600 GiB NVMe	KVM

FlexusX ECSs

FlexusX ECSs are the next-generation cloud servers that offer flexible computing for small and medium-sized enterprises and developers. These ECSs allow for flexible customization of vCPU to memory ratios, enabling you to choose the appropriate flavors based on your service resource needs and minimize resource overhead.

Table 3-43 FlexusX ECS features

Flavor	Compute	Supported Cluster Type
FlexusX1	• vCPU to memory ratio: These ECSs allow for flexible customization of vCPU to memory ratios, enabling you to choose the appropriate flavors based on your service resource needs and minimize resource overhead. • Maximum vCPUs or memory: 1 to 16 for vCPUs, and 1 GiB to 128 GiB for memory • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.8 GHz/3.5 GHz	CCE standard cluster CCE Turbo cluster

Flavor	Compute	Supported Cluster Type
FlexusX1e	• vCPU to memory ratio: These ECSs allow for flexible customization of vCPU to memory ratios, enabling you to choose the appropriate flavors based on your service resource needs and minimize resource overhead. • Maximum vCPUs or memory: 2 to 32 for vCPUs, and 2 GiB to 256 GiB for memory • Base/Turbo frequency: 2.45 GHz/3.5 GHz	CCE standard cluster CCE Turbo cluster

Table 3-44 FlexusX1 ECS specifications

vCPUs	Assured/Max. Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
2	0.2/2	30	2	2	8
4	0.4/3	50	2	2	16
6	0.6/4	60	2	2	24
8	0.8/6	80	2	2	32
12	1.2/8	90	4	3	48
16	1.6/12	100	4	3	64

Table 3-45 FlexusX1e ECS specifications

vCPUs	Assured/Max. Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
2	1/2	40	2	2	16
4	1.5/3	60	2	3	32
8	2.5/6	100	4	4	64
12	4/8	150	4	6	96
16	5/12	200	8	8	128
20	5/13	220	8	8	128
24	6/14	250	8	8	192

vCPUs	Assured/Max. Bandwidth (Gbit/s)	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Max. ENIs
28	6/15	280	8	8	192
32	8/16	300	16	8	256

AI-accelerated ECSs

AI-accelerated ECSs are dedicated for accelerating AI applications. These ECSs use Ascend processors that feature low power consumption, high compute capabilities, and significantly improved energy efficiency for AI inference and video encoding and decoding.

AI-accelerated ECSs are classified as kAi series and Ai series ECSs.

- kAi series: Arm ECSs, which use Kunpeng 920 processors
- AI series: x86 ECSs, which use Intel Xeon processors

Table 3-46 AI-accelerated ECS features

ECS Type	Compute	Network	Supported Cluster Type
kAi1s	• vCPU to memory ratio: 1:1 or 1:2 • Number of vCPUs: 4 to 48 • Kunpeng 920 processor • Base frequency: 2.6 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 4,000,000 • Maximum intranet bandwidth: 20 Gbit/s	CCE standard cluster
Ai1s	• vCPU to memory ratio: 1:2 or 1:4 • Number of vCPUs: 2 to 32 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 2,000,000 • Maximum intranet bandwidth: 25 Gbit/s	CCE standard cluster

ECS Type	Compute	Network	Supported Cluster Type
Ai2	• vCPU to memory ratio: 1:4 • Number of vCPUs: 4 to 96 • 3rd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 8,500,000 • Maximum intranet bandwidth: 40 Gbit/s	CCE standard cluster CCE Turbo cluster
Ai1	• vCPU to memory ratio: 1:4 • Number of vCPUs: 2 to 32 • 2nd Generation Intel® Xeon® Scalable Processor • Base/Turbo frequency: 2.6 GHz/3.5 GHz	• Ultra-high PPS throughput • Higher compute specifications for better network performance • Maximum PPS: 2,000,000 • Maximum intranet bandwidth: 25 Gbit/s	CCE standard cluster

Table 3-47 kAi1s ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Ascend 310	Virtualization
kai1s.xlarge.1	4	4	3/0.8	20	2	2	1	KVM
kai1s.2xlarge.1	8	8	4/1.5	40	2	3	2	KVM
kai1s.4xlarge.1	16	16	6/3	80	4	4	4	KVM
kai1s.3xlarge.2	12	24	8/4	100	4	4	4	KVM

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth	Max. PPS (10,000)	Max. NIC Queues	Max. NICs	Ascend 310	Virtualization
kai1s.4xlarge.2	16	32	10/6	140	4	5	6	KVM
kai1s.6xlarge.2	24	48	12/8	200	8	6	8	KVM
kai1s.9xlarge.2	36	72	16/12	280	8	6	12	KVM
kai1s.12xlarge.2	48	96	20/16	400	16	6	12	KVM

Table 3-48 Ai1s ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./ Assured Bandwidth	Max. PPS (10,000)	Ascend 310 Processors	Ascend RAM (GiB)	Max. NIC Queues	Max. NICs	Virtualization
ai1s.large.4	2	8	4/1.3	20	1	8	2	2	KVM
ai1s.xlarge.4	4	16	6/2	35	2	16	2	3	KVM
ai1s.2xlarge.4	8	32	10/4	50	4	32	4	4	KVM
ai1s.4xlarge.4	16	64	15/8	100	8	64	8	8	KVM
ai1s.8xlarge.4	32	128	25/15	200	16	128	8	8	KVM

Table 3-49 Ai2 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth	Max. PPS (10,000)	Ascend Chip	Ascend RAM (GiB)	Max. NIC Queues	Max. NICs	Virtualization
ai2.xlarge.4	4	16	8/1.6	80	1	24	3	3	KVM
ai2.2xlarge.4	8	32	15/3	150	1	24	4	4	KVM
ai2.4xlarge.4	16	64	20/6	280	1	24	8	8	KVM
ai2.8xlarge.4	32	128	30/12	550	2	48	8	8	KVM
ai2.16xlarge.4	64	256	36/24	800	4	96	16	8	KVM
ai2.24xlarge.4	96	384	40/36	850	6	144	32	8	KVM

Table 3-50 Ai1 ECS specifications

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth	Max. PPS (10,000)	Ascend 310 Processors	Ascend RAM (GiB)	Max. NIC Queues	Max. NICs	Virtualization
ai1.large.4	2	8	4/1.3	20	1	8	2	2	KVM
ai1.xlarge.4	4	16	6/2	35	2	16	2	3	KVM
ai1.2xlarge.4	8	32	10/4	50	4	32	4	4	KVM

Flavor	vCPUs	Memory (GiB)	Max./Assured Bandwidth	Max. PPS (10,000)	Ascend 310 Processors	Ascend RAM (GiB)	Max. NIC Queues	Max. NICs	Virtualization
ai1.4xlarge.4	16	64	15/8	100	8	64	8	8	KVM
ai1.8xlarge.4	32	128	25/15	200	16	128	8	8	KVM

3.5 Creating a Node

Prerequisites

- At least one cluster has been created.
- A key pair has been created for identity authentication upon remote node login.

  If you use a password to log in to a node, skip this step. For details, see **Creating a Key Pair**.

Notes and Constraints

- The DNS configuration of a subnet where a node is located cannot be modified because OBS and other dependent services are necessary for creating nodes.
- When IPv4/IPv6 dual stack is enabled, DHCP unlimited lease cannot be enabled for the selected node subnet.

Precautions

- To maintain node stability, Kubernetes components such as kubelet, kube-proxy, and docker, as well as the Kubernetes system, will be allocated a specific number of node resources based on the node flavor. Therefore, the total number of node resources and the number of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components. For details, see **Node Resource Reservation Policy**.
- Networks including VM networks and container networks of nodes are all managed by CCE. Do not add or delete ENIs, or change routes and IP addresses. Otherwise, services may be unavailable. For example, the ENI named **gw_11cbf51a@eth0** on the node is the container network gateway and cannot be modified.

- **Pay-per-use** nodes in a cluster will be deleted directly after you perform certain operations on the **Nodes** page of the CCE console, while **yearly/monthly** nodes cannot be deleted directly on the CCE console. Instead, you can click **Billing** in the upper right corner, choose **Orders** > **Unsubscriptions and Returns/Exchanges**, and unsubscribe from the corresponding resources.

Procedure

After a cluster is created, you can create nodes for the cluster.

Step 1 Log in to the **CCE console**.

Step 2 In the navigation pane of the CCE console, choose **Clusters**. Click the target cluster name to access its details page.

Step 3 In the navigation pane, choose **Nodes**. On the page displayed, click the **Nodes** tab and then **Create Node** in the upper right corner. Configure node parameters.

Configurations

You can configure the flavor and OS of a cloud server, on which your containerized applications run.

Table 3-51 Node configuration parameters

Parameter	Description
Billing Mode	The following billing modes are supported: • Yearly/Monthly You must specify the required duration if **Yearly/Monthly** is selected. You can choose whether to select **Auto-renew** based on site requirements. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. • Pay-per-use Resources will be billed based on usage duration. You can provision or delete resources at any time. • Spot price A postpaid billing mode, in which a spot cloud server will be billed based on the service duration at a lower price than that of a pay-per-use ECS cloud server with the same specifications. For details, see **Spot Pricing ECSs**. **NOTE** – If you purchase both a data disk and an EIP when creating a spot pricing ECS, they will be automatically released when the ECS is released. If you attach data disks or bind EIPs to an existing spot pricing ECS, these resources must be manually released after the ECS is deleted. – Spot pricing ECSs cannot be reset, accepted for management, removed, migrated, changed to yearly/monthly, or upgraded by resetting the cluster. Before resetting a cluster for an upgrade, delete spot pricing nodes and set the number of pods in the spot pricing node pool to 0. – Spot pricing ECSs may be released if the quoted price is lower than the market price or if there are not enough resources available. Install the latest NPD add-on in the cluster so that you can receive a notification 5 minutes before a spot pricing ECS is released. This add-on generates a ReceivedReclaimNodeNotification event, adds taint **node-problem-controller.cce.io/SpotPriceNodeReclaimNotification: NoExecute** to the node (spot pricing ECS), and evicts pods on the node. This allows pods to be migrated to other nodes before the spot pricing ECS is deleted.
AZ	AZ where the node is located. Nodes in a cluster can be created in different AZs for higher reliability. The value cannot be changed after the node is created. Select **Random** to deploy your node in a random AZ based on the selected node flavor. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. To enhance workload availability, create nodes in different AZs.

Parameter	Description
Node Type	Select a node type based on service requirements. Then, the available node flavors will be automatically displayed in the **Specifications** area for you to select. CCE standard clusters support the following node types: • ECS (VM): A virtualized ECS is used as a cluster node. • ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node. • BMS: A traditional bare metal server is used as a cluster node. Local disks of bare metal servers can be used as data disks. CCE Turbo clusters support the following node types: • ECS (VM): A virtualized ECS is used as a cluster node. A CCE Turbo cluster supports only the cloud servers that allow multiple ENIs. Select a server type displayed on the CCE console. • ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node.
Specifications	Select node flavors as needed. A node needs at least two vCPU cores and 4 GiB of memory. The available node flavors vary depending on AZs. Obtain the flavors displayed on the console. **NOTE** Kunpeng (Arm) nodes can be added to CCE clusters. The specifications displayed on the node creation can be used.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see **Mapping Between Node OSs and Container Engines**.
OS	Select an OS type. Different types of nodes support different OSs. • **Public image**: Select a public image for the node. • **Private image**: Select a private image for the node. For details about how to create a private image, see **Creating a Custom CCE Node Image**. **NOTE** Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS.

Parameter	Description
Node Name	Name of the node. When nodes (ECSs) are created in batches, the value of this parameter is used as the name prefix for each ECS. The system generates a default name for you, which can be modified. Enter 1 to 56 characters. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. The name must start with a lowercase letter and cannot end with a hyphen (-). Only lowercase letters or digits are allowed before and after periods (.).
Enterprise Project	This parameter is available only for enterprise users who have enabled an enterprise project, and the cluster version must be v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. After an enterprise project is selected, nodes will be created in it. To manage clusters and other resources like nodes, load balancers, and node security groups, you can use the Enterprise Project Management Service (EPS). For more details, see **Enterprise Management**.
Login Mode	• **Password** The default username is **root**. Enter the password for logging in to the node and confirm the password. Be sure to remember the password as you will need it when you log in to the node. • **Key Pair** Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**. • **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images) Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image.

Storage Settings

Configure storage resources on a node for the containers running on it. Select a disk type and configure its size based on service requirements. For details about EVS disks, see **Disk Types and Performance**.

Table 3-52 Configuration parameters

Parameter	Description
System Disk	System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB. **NOTE** General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**. General-purpose SSD V2 EVS disks are available only in clusters of v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **System Disk Encryption**: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. Only the nodes of the **Elastic Cloud Server (VM)** type in certain regions support system disk encryption. For details, see the console. • **Not encrypted** is selected by default. • If you select **Enabled (key)** for **System Disk Encryption**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box. • If you select **Enabled (KMS key ID)** for **System Disk Encryption**, enter a KMS key (which can be shared by others) in the current region.
System Component Storage	Select a disk for storing system components. • **Data Disk**: added for storing container runtime and kubelet components by default. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. This data disk cannot be deleted or detached. Otherwise, the node will be unavailable. • **System Disk**: stores CCE resources such as downloaded images, ephemeral storage for containers, and container stdout logs. If the system disk is fully occupied, it will negatively affect the stability of the node. **NOTE** In clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later, you can select a disk for storing system components. If **CCE Node Problem Detector** is used, ensure that its version is 1.19.2 or later.

Parameter	Description
Data Disk	• **At least one default data disk must be added** for storing container runtime and kubelet components if **System Component Storage** is set to **Data Disk**. **This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.** This function is available for clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, or v1.29.4-r0. – Default data disk: used for container runtime and kubelet components. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. – Other common data disks: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. • If **System Component Storage** is set to **System Disk**, you do not need to add a default data disk. In this case, all data disks are common ones: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. This function is available for clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later versions. **NOTE** • If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk. • Local disks may break down and do not ensure data reliability. Store your service data in EVS disks, which are more reliable than local disks. • General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**. General-purpose SSD V2 EVS disks are available only in clusters of v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **Advanced Settings** Expand the area and configure the following parameters: • **Data Disk Space Allocation**: allocates space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. • **Data Disk Encryption**: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **BMS nodes do not support data disk encryption, and this option is available only in certain regions. For details, see the console.** – **Not encrypted** is selected by default. – If you select **Enabled (key)** for **Data Disk Encryption**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box.

Parameter	Description
	– If you select **Enabled (KMS key ID)** for **Data Disk Encryption**, enter a KMS key (which can be shared by others) in the current region. **Adding data disks** A maximum of 16 data disks can be attached to an ECS and 10 to a BMS. By default, a raw disk is created without any processing. You can also click **Expand** and select any of the following options: • **Default**: By default, a raw disk is created without any processing. • **Mount Disk**: The data disk is attached to a specified directory. • **Use as PV**: applicable when there is a high performance requirement on PVs. The **node.kubernetes.io/local-storage-persistent** label is added to the node with PV configured. The value is **linear** or **striped**. • **Use as ephemeral volume**: applicable when there is a high performance requirement on emptyDir. **PVs** and **EVs** support the following write modes: • **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up. • **Striped**: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out. This option can be selected only when there are multiple volumes. **NOTE** • Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended. • Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 3-53 Configuration parameters

Parameter	Description
VPC/Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead.

Parameter	Description
Node IP Address	IP address of the specified node. By default, the value is randomly allocated.
EIP	An ECS without a bound EIP cannot access the Internet or be accessed by public networks. The default value is **Do not use**. **Use existing** and **Auto create** are supported.

Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

Table 3-54 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. A maximum of eight resource tags can be added. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag.
Kubernetes Label	A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click **Add Label** for more. A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see **Labels and Selectors**.

Parameter	Description
Taint	This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters: • **Key**: A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. • **Value**: A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. For details, see **Managing Node Taints**. **NOTE** For a cluster of v1.19 or earlier, the workload may have been scheduled to a node before the taint is added. To avoid such a situation, select a cluster of v1.19 or later.
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. Value range: 16 to 256 This limit prevents the node from being overloaded with pods. This number is also decided by other factors. For details, see **Maximum Number of Pods That Can Be Created on a Node**.
ECS Group	An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group. **Anti-affinity**: ECSs in an ECS group are deployed on different physical hosts to improve service reliability. Select an existing ECS group, or click **Add ECS Group** to create one. After the ECS group is created, click the refresh icon.
Pre-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.

Parameter	Description
Post-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation. **NOTE** Do not run the **reboot** command in the post-installation script to restart the system immediately. To restart the system, run the **shutdown -r 1** command to restart with a delay of one minute.
Agency	An agency is created by the tenant administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. If no agency is available, click **Create Agency** on the right to create one.
Kubernetes Node Name	The Kubernetes node name is the value of **metadata.labels.kubernetes.io/hostname** in the YAML file of the node. The following two values are supported: • **Node private IP** (default) • **Cloud server name**: Use the custom cloud server name configured in node settings. Cloud server names may be duplicate. To prevent name conflicts, CCE randomly adds a five-digit random suffix to the end of each cloud server name. **NOTICE** – This function is available only when the cluster version is v1.23.4-r0 or later. – The name of a cloud server can be specified as the name of a Kubernetes node only when the cloud server is created or managed. After the cloud server is created or managed, the Kubernetes node name cannot be changed. For details, see **ECS Names, Node Names, and Kubernetes Node Names**. – Existing nodes in the cluster still use the private IP address as the Kubernetes node name. Newly created or accepted nodes can use cloud server names. In this scenario, some Kubernetes node names may be inconsistent with node private IP addresses, and adaptation is required. For example, when configuring node affinity, you cannot use the node private IP address as the node name to configure a scheduling policy. To change the Kubernetes node name of the existing nodes to the cloud server name, remove these nodes from the cluster and accept them again. Before doing so, learn about the possible impacts on services when **removing** or **accepting a node**.

Step 4 Configure the number of nodes to be purchased. Then, click **Next: Confirm**. Confirm the configured parameters, price, and specifications. Ensure that you have read and understood the **Image Management Service Statement**.

Step 5 Click **Submit**.

If the node will be billed on a yearly/monthly basis, click **Pay Now** and follow on-screen prompts to pay the order.

The node list page is displayed. If the node status is **Running**, the node is created successfully. It takes about 6 to 10 minutes to create a node.

Step 6 Click **Back to Node List**. The node is created successfully if it changes to the **Running** state.

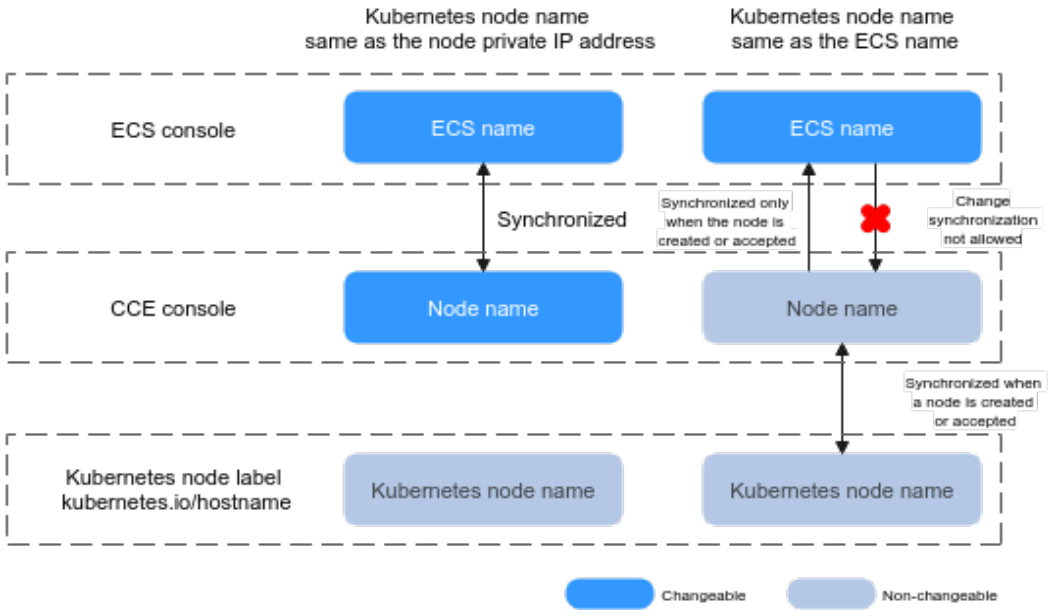
----End

ECS Names, Node Names, and Kubernetes Node Names

- ECS name: the name of an ECS on the ECS page. You can customize an ECS name when creating the ECS (node).
- Node name: the name of a node on the CCE console, which can be synchronized with the ECS name on the ECS console. However, after the Kubernetes node name is specified, the change of the ECS name cannot be synchronized to the node name.
- Kubernetes node name: value of **metadata.labels.kubernetes.io/hostname** in the YAML file of the node. The name of a cloud server can be specified as the name of a Kubernetes node only when the cloud server is created or managed. After the cloud server is created or managed, the Kubernetes node name cannot be changed.

Figure 3-1 shows the synchronization relationship between ECS names, node names, and Kubernetes node names.

Figure 3-1 Relationship between node names



Related Operations

Create a node injection script.

3.6 Accepting Nodes for Management

Scenario

In CCE, you can create a node (**Creating a Node**) or add existing nodes (ECSs or BMSs) to your cluster for management. These nodes can be billed on a yearly/monthly or pay-per-use basis.

NOTICE

- When accepting an ECS, you can reset the ECS OS to a standard public image offered by CCE. If you choose to do so, you need to reset the password or key pair, and the previous password or key pair will become invalid.
- LVM information, including volume groups (VGs), logical volumes (LVs), and physical volumes (PVs), will be deleted from the system disks and data disks attached to the selected ECSs during acceptance. Ensure that the information has been backed up.
- During the acceptance of an ECS, do not perform any operation on the ECS through the ECS console.

Notes and Constraints

- BMSs and ECSs, as well as DeHs can be managed.

Prerequisites

The cloud servers to be managed must meet the following requirements:

- The node to be accepted must be in the **Running** state and not used by other clusters. In addition, the node to be accepted does not carry the CCE-Dynamic-Provisioning-Node tag.
- The node to be accepted and the cluster must be in the same VPC. (If the cluster version is earlier than v1.13.10, the node to be accepted and the CCE cluster must be in the same subnet.)
- Data disks must be attached to the nodes to be managed if the system components of these nodes are separately stored. A local disk (disk-intensive disk) or a data disk of at least 20 GiB can be attached to the node, and any data disks already attached cannot be smaller than 10 GiB. For details about how to attach a data disk, see **Adding a Disk to an ECS**.
- The node to be accepted has 2-core or higher CPU, 4 GiB or larger memory, and only one NIC.
- If an enterprise project is used, the node to be accepted and the cluster must be in the same enterprise project. Otherwise, resources cannot be identified during management. As a result, the node cannot be accepted.
- Only cloud servers with the same data disk configuration can be accepted in batches for management.

- If IPv6 is enabled for a cluster, only nodes in a subnet with IPv6 enabled can be accepted and managed. If IPv6 is not enabled for the cluster, only nodes in a subnet without IPv6 enabled can be accepted.
- Nodes in a CCE Turbo cluster must support sub-ENIs or be bound to at least 16 ENIs. For details about the node flavors, see the options provided on the console when you create a node.
- Data disks that have been partitioned will be ignored during node management. Ensure that there is at least one unpartitioned data disk meeting the specifications is attached to the node.

Procedure

Step 1 Log in to the CCE console and go to the cluster where the node to be accepted resides.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab and then **Accept Node** in the upper right corner.

Step 3 Specify node parameters.

Configurations

Table 3-55 Node configuration parameters

Parameter	Description
Node Pool	• Default node pool: You can add nodes that meet the management requirements to the default node pool of the cluster. • Custom node pool: You only need to select the nodes that meet the management requirements. These nodes will use the basic, network, and advanced settings of the custom node pool. You do not need to configure other parameters. For details, see **Accepting Nodes in a Node Pool**.
Specifications	Click **Select Cloud Server** and select the servers to be accepted. You can select multiple cloud servers for batch management. However, only the cloud servers with the same specifications, AZ, and data disk configuration can be added in batches. If a cloud server contains multiple data disks, select one of them for the container runtime and kubelet.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see **Mapping Between Node OSs and Container Engines**.

Parameter	Description
OS	Select an OS type. Different types of nodes support different OSs. • **Public image**: Select a public image for the node. • **Private image**: Select a private image for the node. For details about how to create a private image, see **Creating a Custom CCE Node Image**. **NOTE** Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS.
Login Mode	• **Password** The default username is **root**. Enter the password for logging in to the node and confirm the password. Be sure to remember the password as you will need it when you log in to the node. • **Key Pair** Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**. • **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images) Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 3-56 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.

Parameter	Description
System Component Storage	Select a disk for storing system components. • **Data Disk**: added for storing container runtime and kubelet components by default. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. This data disk cannot be deleted or detached. Otherwise, the node will be unavailable. • **System Disk**: stores CCE resources such as downloaded images, ephemeral storage for containers, and container stdout logs. If the system disk is fully occupied, it will negatively affect the stability of the node. **NOTE** In clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later, you can select a disk for storing system components. If **CCE Node Problem Detector** is used, ensure that its version is 1.19.2 or later.
Data Disk	• **At least one default data disk must be added** for storing container runtime and kubelet components if **System Component Storage** is set to **Data Disk**. **This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.** This function is available for clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, or v1.29.4-r0. • If **System Component Storage** is set to **System Disk**, you do not need to add a default data disk. In this case, all data disks are common ones: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. This function is available for clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later versions. Click **Expand** to configure **Data Disk Space Allocation**, which is used to allocate space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. For other data disks, a raw disk is created without any processing by default. You can also click **Expand** and select **Mount Disk** to mount the data disk to a specified directory. Data disks can also be used as **local PVs** or **local EVs**.

Advanced Settings

Table 3-57 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. A maximum of eight resource tags can be added. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag.
Kubernetes Label	Click **Add Label** to set the key-value pair attached to the Kubernetes objects (such as pods). A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see **Labels and Selectors**.
Taint	This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters: • **Key**: A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. • **Value**: A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. **NOTICE** • If taints are used, you must configure tolerations of pods. Otherwise, a scale-out may fail or pods cannot be scheduled onto the added nodes. • After a node pool is created, you can click **Edit** to modify its configuration. The modification will be synchronized to all nodes in the node pool.
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. Value range: 16 to 256 This limit prevents the node from being overloaded with pods.

Parameter	Description
Pre-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.
Post-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation.

Step 4 Click **Next: Confirm**. Ensure that you have read and understood the **Image Management Service Statement**. Click **Submit**.

----End

3.7 Logging In to a Node

Prerequisites

- Before you log in to a node using SSH, ensure that the SSH port (22 by default) is enabled in the security group of the node. For details, see **Configuring Security Group Rules**.
- Before you log in to a node (an ECS) using SSH through the Internet, ensure that the ECS already has an EIP bound.
- Only login to a running ECS is allowed.
- Only the user **root** can log in to a Linux server.

Login Modes

You can log in to an ECS in either of the following modes:

- Management console (VNC)

  If an ECS has no EIP, log in to the ECS console and click **Remote Login** in the same row as the ECS.

  For details, see **Login Using VNC**.

- SSH

  This mode applies only to ECSs running Linux. Usually, you can use a remote login tool, such as PuTTY, Xshell, and SecureCRT, to log in to your ECS. If none of the remote login tools can be used, log in to the ECS console and click **Remote Login** in the same row as the ECS to view the connection status and running status of the ECS.

NOTE

- You can use either an SSH key or SSH password for login. For details, see **Login Using an SSH Key** and **Login Using an SSH Password**
- When you use the Windows OS to log in to a Linux node, set **Auto-login username** to **root**.
- The CCE console does not support node OS upgrade. Do not upgrade the node OS using the **yum update** command. Otherwise, the container networking components will be unavailable. For details on how to manually restore the container network, see **What Can I Do If the Container Network Becomes Unavailable After yum update Is Used to Upgrade the OS?**

Table 3-58 Linux ECS login modes

EIP Binding	On-Premises OS	Connection Method
Yes	Windows	Use a remote login tool, such as PuTTY or Xshell. • SSH password authentication: **Login Using an SSH Password** • SSH key authentication: **Login Using an SSH Key**
Yes	Linux	Run commands. • SSH password authentication: **Login Using an SSH Password** • SSH key authentication: **Login Using an SSH Key**
Yes/No	Windows/ Linux	Remote login using the management console: **Login Using VNC**

3.8 Management Nodes

3.8.1 Managing Node Labels

You can add different labels to nodes and define different attributes for labels. By using these node labels, you can quickly understand the characteristics of each node.

Node Label Usage Scenario

Node labels are mainly used in the following scenarios:

- Node management: Node labels are used to classify nodes.
- Node affinity or anti-affinity for workloads: By adding labels to nodes, you can schedule pods to specific nodes through node affinity or prevent pods from being scheduled to specific nodes through node anti-affinity. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.

Inherent Label of a Node

After a node is created, some fixed labels exist and cannot be deleted. For details about these labels, see **Table 3-59**.

NOTE

Do not manually change the inherent labels that are automatically added to a node. If the manually changed value conflicts with the system value, the system value is used.

Table 3-59 Inherent labels of a node

Key	Description
New: topology.kubernetes.io/region Old: failure-domain.beta.kubernetes.io/region	Region where the node is located
New: topology.kubernetes.io/zone Old: failure-domain.beta.kubernetes.io/zone	AZ where the node is located
New: node.kubernetes.io/baremetal Old: failure-domain.beta.kubernetes.io/is-baremetal	Whether the node is a bare metal node **false** indicates that the node is not a bare metal node.
node.kubernetes.io/container-engine	Container engine Example: **docker** or **containerd**
node.kubernetes.io/instance-type	Node specifications
kubernetes.io/arch	Node processor architecture
kubernetes.io/hostname	Node name
kubernetes.io/os	Node OS type
node.kubernetes.io/subnetid	ID of the subnet where the node is located
os.architecture	Node processor architecture For example, **amd64** indicates a AMD64-bit processor.
os.name	Node OS name
os.version	Node OS kernel version
accelerator/huawei-npu	NPU node labels
accelerator	GPU node labels
cce.cloud.com/cce-nodepool	The dedicated label of a node in a node pool

Adding or Deleting a Node Label

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

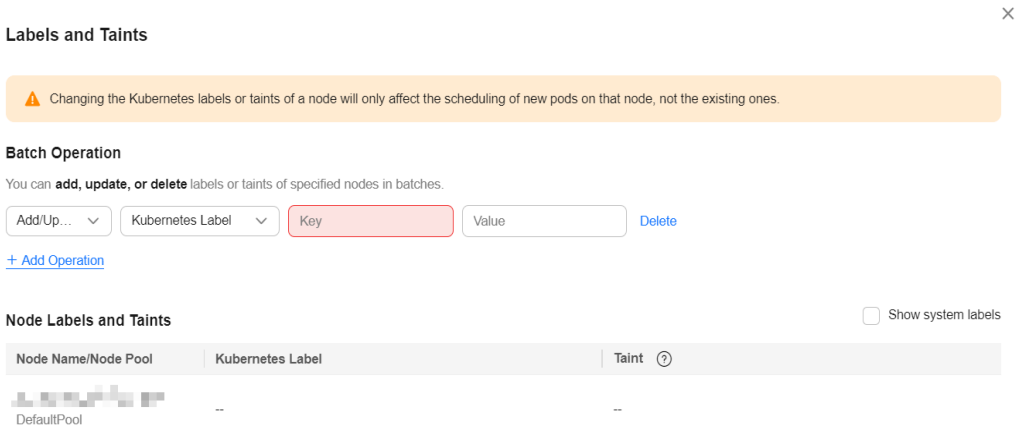
Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab, select the target node and click **Labels and Taints** in the upper left corner.

Step 3 In the displayed dialog box, click **Add operation** under **Batch Operation**, and then choose **Add/Update** or **Delete**.

Enter the key and value of the label to be added or deleted, and click **OK**.

For example, the key is **deploy_qa** and the value is **true**, indicating that the node is used to deploy the QA (test) environment.

Figure 3-2 Adding a node label



Step 4 After the label is added, check the added label in node data.

----End

3.8.2 Managing Node Taints

Taints enable a node to repel specific pods to prevent these pods from being scheduled to the node.

Procedure for Operations Performed on the Console

On the CCE console, you can also batch manage nodes' taints.

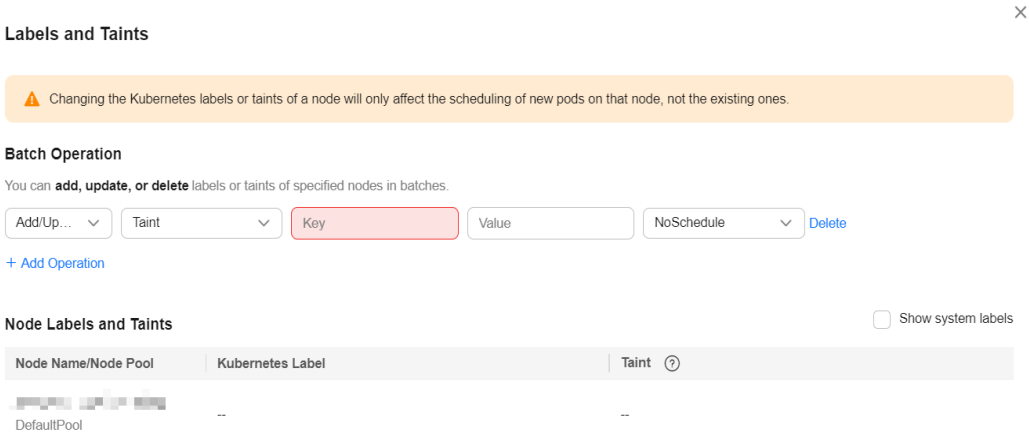
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab, select the target node and click **Labels and Taints** in the upper left corner.

Step 3 In the displayed dialog box, click **Add Operation** under **Batch Operation**, and then choose **Add/Update** or **Delete** as well as **Taint**.

Enter the key and value of the taint to be operated, choose a taint effect, and click **OK**.

Figure 3-3 Adding a taint



Step 4 After the taint is added, check the added taint in node data.

----End

Procedure for Operations Performed Through kubectl

A taint is a key-value pair associated with an effect. The following effects are available:

- NoSchedule: No pod will be scheduled onto the node unless it has a matching toleration. Existing pods will not be evicted from the node.
- PreferNoSchedule: Kubernetes prevents pods that cannot tolerate this taint from being scheduled onto the node.
- NoExecute: If the pod has been running on a node, the pod will be evicted from the node. If the pod has not been running on a node, the pod will not be scheduled onto the node.

To add a taint to a node, run the **kubectl taint node** *nodename* command as follows:

```
$ kubectl get node
NAME            STATUS   ROLES    AGE    VERSION
192.168.10.170   Ready    <none>   73d    v1.19.8-r1-CCE21.4.1.B003
192.168.10.240   Ready    <none>   4h8m   v1.19.8-r1-CCE21.6.1.2.B001
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule
node/192.168.10.240 tainted
```

To view the taint configuration, run the **describe** and **get** commands as follows:

```
$ kubectl describe node 192.168.10.240
Name:               192.168.10.240
...
Taints:             key1=value1:NoSchedule
...
$ kubectl get node 192.168.10.240 -oyaml
apiVersion: v1
...
spec:
  providerID: 06a5ea3a-0482-11ec-8e1a-0255ac101dc2
  taints:
  - effect: NoSchedule
    key: key1
    value: value1
...
```


To remove a taint, add a hyphen (-) at the end of the command for adding a taint, as shown in the following example:

```
$ kubectl taint node 192.168.10.240 key1=value1:NoSchedule-
node/192.168.10.240 untainted
$ kubectl describe node 192.168.10.240
Name:               192.168.10.240
...
Taints:             <none>
...
```

Configuring a Node Scheduling Policy in One-Click Mode

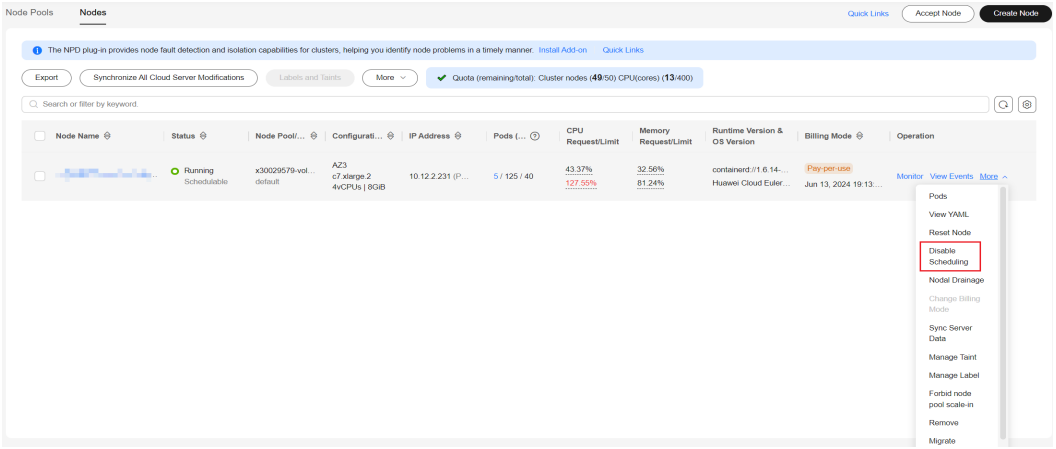
You can configure a node to be unschedulable on the console. Then, CCE will add a taint with key **node.kubernetes.io/unschedulable** and the **NoSchedule** setting to the node. After a node is set to be unschedulable, new pods cannot be scheduled to this node, but pods running on the node are not affected.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

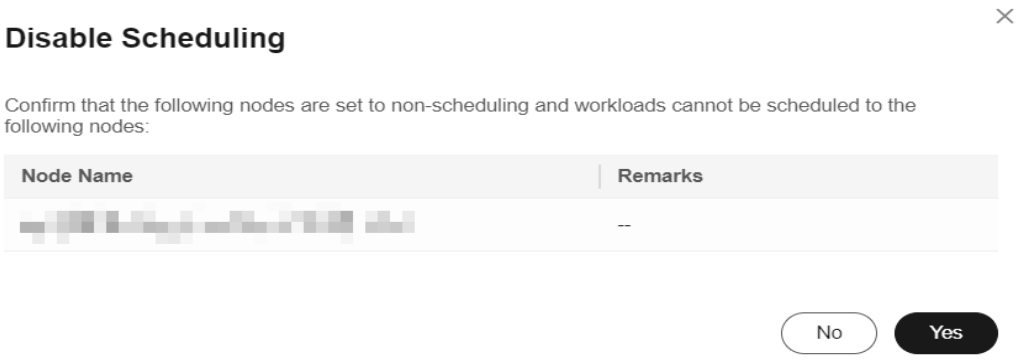
Step 3 In the node list, locate the target node and choose **More** > **Disable Scheduling** in the **Operation** column.

Figure 3-4 Disable Scheduling



Step 4 In the dialog box that is displayed, click **Yes** to configure the node to be unschedulable.

Figure 3-5 Disable Scheduling



This operation will add a taint to the node. You can use kubectl to view the content of the taint.

```
$ kubectl describe node 192.168.10.240
...
Taints:             node.kubernetes.io/unschedulable:NoSchedule
...
```

Step 5 Go back to the node list, locate the target node, and choose **More** > **Enable Scheduling**. Then, the node changes to be schedulable.

----End

System Taints

When some issues occurred on a node, Kubernetes automatically adds a taint to the node. The built-in taints are as follows:

- node.kubernetes.io/not-ready: The node is not ready. The node **Ready** value is **False**.
- node.kubernetes.io/unreachable: The node controller cannot access the node. The node **Ready** value is **Unknown**.
- node.kubernetes.io/memory-pressure: The node memory is approaching the upper limit.
- node.kubernetes.io/disk-pressure: The node disk space is approaching the upper limit.
- node.kubernetes.io/pid-pressure: The node PIDs are approaching the upper limit.
- node.kubernetes.io/network-unavailable: The node network is unavailable.
- node.kubernetes.io/unschedulable: The node cannot be scheduled.
- node.cloudprovider.kubernetes.io/uninitialized: If an external cloud platform driver is specified when kubelet is started, kubelet adds a taint to the current node and marks it as unavailable. After a controller of **cloud-controller-manager** initializes the node, kubelet will delete the taint.

Related Operations (Tolerations)

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node. This marks that the node should not accept any pods that do not tolerate the taints.

Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
```



```
tolerations:
- key: "key1"
  operator: "Equal"
  value: "value1"
  effect: "NoSchedule"
```

In the preceding example, the toleration label of the pod is key1=value1 and the taint effect is NoSchedule. Therefore, the pod can be scheduled onto the corresponding node.

You can also configure tolerations similar to the following information, which indicates that the pod can be scheduled onto a node when the node has the taint key1:

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

3.8.3 Resetting a Node

Scenario

You can reset a node to modify the node configuration, such as the node OS and login mode.

Resetting a node will reinstall the node OS and the Kubernetes software on the node. If a node is unavailable because you modify the node configuration, you can reset the node to rectify the fault.

Notes and Constraints

- To enable node resetting in CCE standard clusters or CCE Turbo clusters, the version must be v1.13 or later.
- For Kunpeng clusters, the version must be v1.15 or later to support node resetting.

Precautions

- Only worker nodes can be reset. If the node is still unavailable after the resetting, delete the node and create a new one.
- **After a node is reset, the node OS will be reinstalled. Before resetting a node, drain the node to gracefully evict the pods running on the node to other available nodes. Perform this operation during off-peak hours.**
- **After a node is reset, its system disk and data disks will be cleared. Back up important data before resetting a node.**
- **If you reset a worker node that has an additional data disk attached on the ECS console, the attachment will be removed. To keep the data, you need to reattach the disk.**
- The IP addresses of the workload pods on the node will change, but the container network access is not affected.
- There is remaining EVS disk quota.
- When a node is reset, the backend will make it unschedulable.
- Resetting a node will clear the Kubernetes labels and taints you added (those added by editing a node pool will not be lost). As a result, node-specific

resources (such as local storage and workloads scheduled to this node) may be unavailable.

- Resetting a node will cause PVC/PV data loss for the **local PV** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod is created and stays in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled. After the node is reset, the pod may be scheduled to the reset node. In this case, the pod remains in the creating state because the underlying logical volume corresponding to the PVC does not exist.

Resetting Nodes in the Default Pool

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 In the node list of the default pool, select one or more nodes to be reset and choose **More** > **Reset Node** in the **Operation** column.

Step 4 In the displayed dialog box, click **Next**.

Step 5 Specify node parameters.

Compute Settings

Table 3-60 Configuration parameters

Parameter	Description
Specifications	Specifications cannot be modified when you reset a node.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see **Mapping Between Node OSs and Container Engines**.
OS	Select an OS type. Different types of nodes support different OSs. • **Public image**: Select a public image for the node. • **Private image**: Select a private image for the node. For details about how to create a private image, see **Creating a Custom CCE Node Image**. **NOTE** Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS.

Parameter	Description
Login Mode	• **Password** The default username is **root**. Enter the password for logging in to the node and confirm the password. Be sure to remember the password as you will need it when you log in to the node. • **Key Pair** Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**. • **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images) Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 3-61 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.
System Component Storage	Select a disk for storing system components. • **Data Disk**: added for storing container runtime and kubelet components by default. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. This data disk cannot be deleted or detached. Otherwise, the node will be unavailable. • **System Disk**: stores CCE resources such as downloaded images, ephemeral storage for containers, and container stdout logs. If the system disk is fully occupied, it will negatively affect the stability of the node. **NOTE** In clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later, you can select a disk for storing system components. If **CCE Node Problem Detector** is used, ensure that its version is 1.19.2 or later.

Parameter	Description
Data Disk	• **At least one default data disk must be added** for storing container runtime and kubelet components if **System Component Storage** is set to **Data Disk**. **This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.** This function is available for clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, or v1.29.4-r0. • If **System Component Storage** is set to **System Disk**, you do not need to add a default data disk. In this case, all data disks are common ones: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. This function is available for clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later versions. Click **Expand** to configure **Data Disk Space Allocation**, which is used to allocate space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. For other data disks, a raw disk is created without any processing by default. You can also click **Expand** and select **Mount Disk** to mount the data disk to a specified directory. Data disks can also be used as **local PVs** or **local EVs**.

Advanced Settings

Table 3-62 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. A maximum of eight resource tags can be added. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE will automatically create the **CCE-Dynamic-Provisioning-Node=***Node ID* tag.
Kubernetes Label	Click **Add Label** to set the key-value pair attached to the Kubernetes objects (such as pods). A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see **Labels and Selectors**.

Parameter	Description
Taint	This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters: • **Key**: A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. • **Value**: A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. **NOTICE** • If taints are used, you must configure tolerations of pods. Otherwise, a scale-out may fail or pods cannot be scheduled onto the added nodes. • After a node pool is created, you can click **Edit** to modify its configuration. The modification will be synchronized to all nodes in the node pool.
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. Value range: 16 to 256 This limit prevents the node from being overloaded with pods.
Pre-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.
Post-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation.

Step 6 Click **Next: Confirm**. Ensure that you have read and understood the **Image Management Service Statement** .

Step 7 Click **Submit**.

----End

Resetting Nodes in a Node Pool

NOTE

- When resetting a node in a node pool, you can only change its storage configuration. All other configurations will follow the settings of the node pool.
- Resetting a node will execute the pre- and post-installation scripts in the current node pool and update the security group configurations to those of the node pool.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 In the node list of the target node pool, select a node to be reset and choose **More** > **Reset Node** in the **Operation** column.

Step 4 Modify the node storage parameters.

Table 3-63 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.
Default Data Disk	Select a data disk for container runtime and kubelet.
Data Disk	Configure advanced settings for each data disk. For the **default data disk**, click **Expand** to configure **Data Disk Space Allocation**, which is used to allocate space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. For a **common data disk**, click **Expand** and select attachment settings. • **Default**: The data disk is attached as a raw disk without any settings. • **Mount Disk**: The data disk is attached to the service directory path. This parameter cannot be left blank or set to a key OS path such as the root directory. • **Use as PV**: The data disk is used as persistent storage volumes for PVCs. For details, see **Local PVs**. • **Use as ephemeral volume**: The data disk is used as ephemeral storage volumes for PVCs. For details, see **Using a Local EV**.

Step 5 Click **OK**.

----End

Resetting Nodes in a Batch

Resetting nodes in a batch varies depending on application scenarios.

Scenario	Supported or Not	Description
Resetting nodes in the default pool in a batch	Conditionally supported	This operation can be performed only if the node flavor, AZ, and disk configurations of all nodes are the same.
Resetting nodes in a node pool in a batch	Conditionally supported	This operation can be performed only if the disk configurations of all nodes are the same.
Resetting nodes in different node pools in a batch	Not supported	Only the nodes in the same node pool can be reset in a batch.

3.8.4 Removing a Node

Scenario

Removing a node from a cluster will re-install the node OS and clear CCE components on the node.

Removing a node will not delete the server corresponding to the node. You are advised to remove nodes at off-peak hours to avoid impacts on your services.

After a node is removed from the cluster, the node is still running and incurs fees.

Notes and Constraints

- If the OS fails to be re-installed after the node is removed, manually re-install the OS. After the re-installation, log in to the node and run the clearance script to clear CCE components. For details, see **Handling Failed OS Reinstallation**.
- Removing a node will cause PVC/PV data loss for the **local PV** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod is created and stays in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled.

Precautions

- When you remove a node, the pods running on it will need to migrate to another node, which could potentially impact your services. To avoid any disruptions, drain the node so that pods can be gracefully evicted to other available nodes. Additionally, perform this task during off-peak hours.
- Unexpected risks may occur during the operation. Back up data beforehand.
- When a node is removed, the backend will make it unschedulable.
- After you remove the node and re-install the OS, the original LVM partitions will be cleared and the data managed by LVM will be cleared. Therefore, back up data in advance.

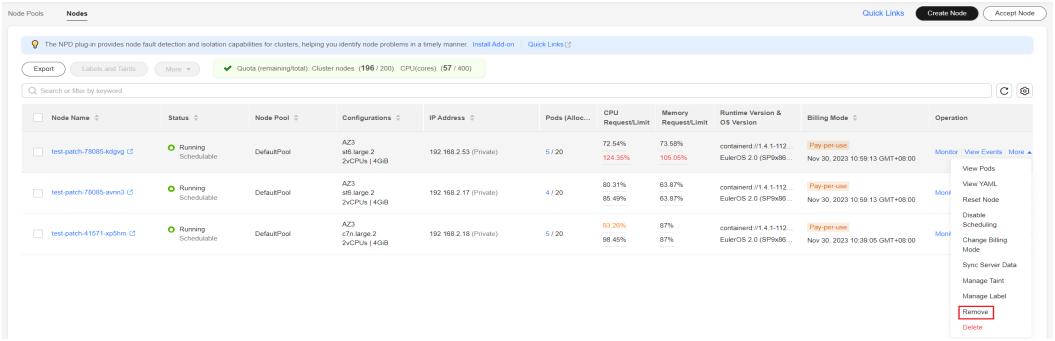
Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

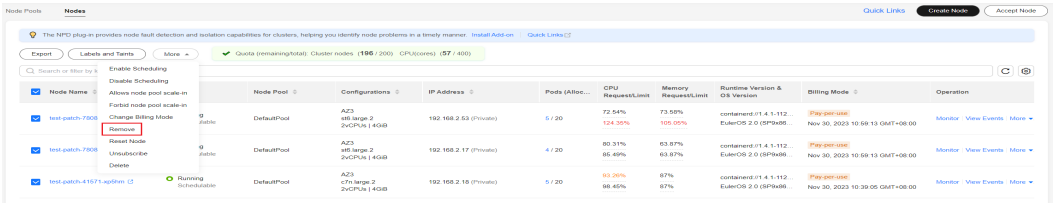
Step 3 Locate the target node and choose **More** > **Remove** in the **Operation** column.

Figure 3-6 Removing a node



You can also select multiple nodes and remove them at a time.

Figure 3-7 Removing multiple nodes at a time



Step 4 In the dialog box displayed, configure the login information required for re-installing the OS and click **Yes**. Wait until the node is removed.

After the node is removed, workload pods on the node are automatically migrated to other available nodes.

----End

Handling Failed OS Reinstallation

You can perform the following steps to re-install the OS and clear the CCE components on the node if previous attempts fail:

Step 1 Log in to the management console of the server and re-install the OS. For details, see **Changing the OS**.

Step 2 Log in to the server and run the following commands to clear the CCE components and LVM data:

Write the following scripts to the **clean.sh** file:

```
lsblk
vgs --noheadings | awk '{print $1}' | xargs vgremove -f
pvs --noheadings | awk '{print $1}' | xargs pvremove -f
lvs --noheadings | awk '{print $1}' | xargs -i lvremove -f --select {}
function init_data_disk() {
```



```
    all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')
    for device in ${all_devices[@]}; do
        isRootDisk=$(lsblk -o KNAME,MOUNTPOINT $device 2>/dev/null| grep -E '[[:space:]]/$' | wc -l )
        if [[ ${isRootDisk} != 0 ]]; then
            continue
        fi
        dd if=/dev/urandom of=${device} bs=512 count=64
        return
    done
    exit 1
}
init_data_disk
lsblk
```

Run the following command:

bash clean.sh

----End

3.8.5 Synchronizing the Data of Cloud Servers

Scenario

Each node in a cluster is a cloud server or physical machine. After a cluster node is created, you can change the cloud server name or specifications as required. Modifying node specifications will affect services. Perform the operation on nodes one by one.

Some information of CCE nodes is maintained independently from the ECS console. After you change the name, EIP, billing mode, or specifications of an ECS on the ECS console, synchronize the ECS with the target node on the CCE console. After the synchronization, information on both consoles is consistent.

Common ECS information to be modified:

- ECS (node) name: **Changing an ECS Name**
- Node specifications: **General Operations for Modifying Specifications**

Notes and Constraints

- Data, including the VM status, ECS names, number of CPUs, size of memory, ECS specifications, and public IP addresses, can be synchronized.

  If an ECS name is specified as the Kubernetes node name, the change of the ECS name cannot be synchronized to the CCE console. For details, see **ECS Names, Node Names, and Kubernetes Node Names**.
- The following data cannot be synchronized: OS, image ID, and disk configuration.

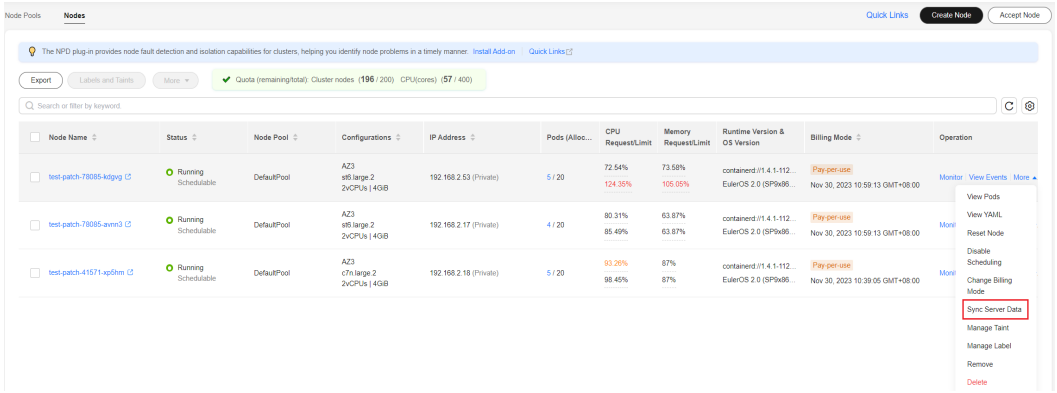
Synchronizing the Data of a Cloud Server

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the target node and choose **More** > **Sync Server Data** in the **Operation** column.

Figure 3-8 Synchronizing server data



After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

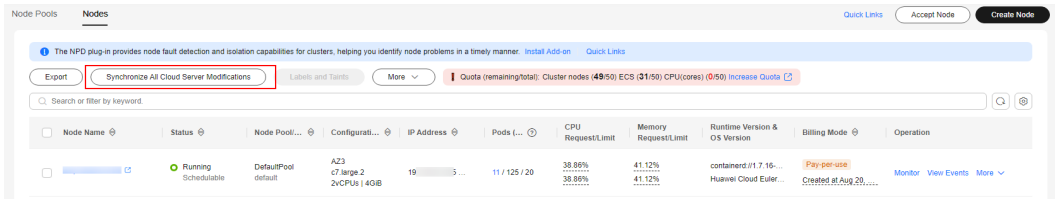
Synchronizing the Data of All Cloud Servers

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Click **Synchronize All Cloud Server Modifications** and click **OK**.

Figure 3-9 Synchronize All Cloud Server Modifications



After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

3.8.6 Draining a Node

Scenario

After you enable nodal drainage on the console, CCE configures the node to be non-schedulable and securely evicts all pods that comply with **Rules for Draining Nodes** on the node. Subsequent new pods will not be scheduled to this node.

When a node becomes faulty, nodal drainage quickly isolates the faulty node. The pods evicted from the faulty node will be scheduled by the workload controller to other nodes that are running properly.

NOTICE

To ensure service availability during drainage, **specify a disruption budget** for your application. Otherwise, the application may become unavailable during pod rescheduling.

Prerequisites

- A cluster is available and the cluster version meets the following requirements:
  - v1.21: v1.21.10-r0 or later
  - v1.23: v1.23.8-r0 or later
  - v1.25: v1.25.3-r0 or later
  - Versions later than v1.25
- To drain a node as an IAM user, you must have at least one of the following permissions (for details, see **Namespace Permissions (Kubernetes RBAC-based)**):
  - cluster-admin (administrator): read and write permissions on all resources in all namespaces.
  - drainage-editor: drain a node.
  - drainage-viewer: view the nodal drainage status but cannot drain a node.

Rules for Draining Nodes

When a node is drained, all pods on the node will be safely evicted. However, CCE will take specific actions for pods that meet certain filtering criteria.

Filter Criterion	Forced Drainage Enabled	Forced Drainage Disabled
The **status.phase** field of the pod is **Succeeded** or **Failed**.	Deletion	Deletion
The pod is not managed by the workload controller.	Deletion	Drainage cancellation
The pod is managed by DaemonSet.	None	Drainage cancellation **NOTE** For clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later, if forcible drainage is not used for DaemonSet pods, the default operation is **None**.

Filter Criterion	Forced Drainage Enabled	Forced Drainage Disabled
A volume of the emptyDir type is mounted to the pod.	Eviction	Drainage cancellation
The pod is a **static pod** directly managed by kubelet	None	None

NOTE

The following operations may be performed on pods during node drainage:

- Deletion: The pod is deleted from the current node and will not be scheduled to other nodes.
- Eviction: The pod is deleted from the current node and rescheduled to another node.
- None: The pod will not be evicted or deleted.
- Drainage cancellation: If a pod on a node cancels drainage, the drainage process of the node is terminated and no pod is evicted or deleted.

Procedure

This section describes how to drain nodes.

Through the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the target node and choose **More** > **Nodal Drainage** in the **Operation** column.

Step 4 In the **Nodal Drainage** window displayed, set parameters.

- **Timeout (s)**: Node drain tasks automatically fail after the specified timeout expires. A value of 0 indicates that task will not time out.
- **Forced Drainage**: If this function is enabled, pods managed by DaemonSet will be ignored, and pods with emptyDir volumes and pods not managed by controllers will be deleted. For details, see **Rules for Draining Nodes**.

Step 5 Click **OK** and wait until the node drainage is complete.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Edit the YAML file for drainage.

The following is an example of **Drainage-test.yaml**:


```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  name: 192.168.1.67-1721616409999   # Drainage resource name
spec:
  nodeName: 192.168.1.67     # Kubernetes name of the node to be drained, which can be obtained by running the kubectl get node command
  force: true
  timeout: 0
```

- **nodeName**: node to be drained. The parameter value is the node name in Kubernetes, not the name displayed on the console.

  You can run the **kubectl get node** command to obtain a node name in Kubernetes.
- **force**: whether to forcibly drain a node. Value **true** means that the drainage is forced, while **false** means it is not.
- **timeout**: timeout measured in seconds. Node drain tasks automatically fail after the specified timeout expires. A value of 0 indicates that task will not time out.

Step 3 Create drainage resources.

```
kubectl create -f Drainage-test.yaml
```

If information similar to the following is displayed, the drainage resources have been created:

```
drainage.node.cce.io/192.168.1.67-1721616409999 created
```

Step 4 Check the result.

```
kubectl get drainages 192.168.1.67-1721616409999 -o yaml
```

If **phase** is **Succeeded**, the operation is successful.

```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  creationTimestamp: "2024-07-22T03:12:56Z"
  generation: 1
  name: 192.168.1.67-1721616409999
  resourceVersion: "2683143"
  uid: 3ec131e4-0505-4c88-8255-ef9d0eb02712
spec:
  force: true
  nodeName: 192.168.1.67
  timeout: 0
status:
  conditions:
  - lastTransitionTime: "2024-07-22T03:12:56Z"
    message: start to drain node
    reason: Started
    status: "True"
    type: Started
  - lastTransitionTime: "2024-07-22T03:13:26Z"
    message: node has been drained
    reason: Succeeded
    status: "True"
    type: Finished
  phase: Succeeded
```

----End

Through APIs

Step 1 Obtain the token in the region where the cluster is located. For details, see **Authentication**.

Step 2 Based on the API format, find the URL for the node drainage API.

URL of the API for draining a node:

```
https://{clusterid}.Endpoint/apis/node.cce.io/v1/drainages
```

- **{clusterid}**: cluster ID, which can be obtained on the **Overview** page of the CCE console.
- **Endpoint**: endpoint of CCE in the region where the cluster is located.

  For details about its value, see **Regions and Endpoints**.

  For example, the endpoint of CCE in the **AP-Singapore** region is **cce.ap-southeast-3.myhuaweicloud.com**.

Step 3 Use the **POST** request method and configure request header parameters.

```
curl --location --request POST 'https://{clusterid}.Endpoint/apis/node.cce.io/v1/drainages' \
--header 'Content-Type: application/json' \
--header 'X-Auth-Token: MIIWww******' \
--data @Drainage.json
```

The following table lists the header parameters contained in the request.

Table 3-64 Request header parameters

Parameter	Mandatory	Type	Description
Content-Type	Yes	String	Message body type (format), for example, application/json.
X-Auth-Token	Yes	String	Use a token to call the API. For details about how to obtain a token, see **Authentication**.

Drainage.json is located in the current directory and contains the following content:

```
{
   "apiVersion": "node.cce.io/v1",
   "kind": "Drainage",
   "metadata": {
      "name": "192.168.1.67-1721616404940"
   },
   "spec": {
      "nodeName": "192.168.1.67",
      "force": true,
      "timeout": 0
   }
}
```

- **nodeName**: node to be drained. The parameter value is the node name in Kubernetes, not the name displayed on the console.

  You can run the **kubectl get node** command to obtain a node name in Kubernetes.
- **force**: whether to forcibly drain a node. Value **true** means that the drainage is forced, while **false** means it is not.

- **timeout**: timeout measured in seconds. Node drain tasks automatically fail after the specified timeout expires. A value of 0 indicates that task will not time out.

----End

Cancelling Node Drainage

This section describes how to stop a node drainage that is currently in progress.

NOTE

In clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions, node drainage can be canceled.

This operation will abort drainage on nodes, but workloads that have been evicted from these nodes will not be automatically migrated back.

Through the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the node that is being drained and click **Cancel Drainage**.

Step 4 In the displayed dialog box, click **OK**. The node status changes to **Drainage cancelled**. You can click **Enable Scheduling** to restore the node to the schedulable state.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Check drainage resources.

```
kubectl get drainages
```

Command output:

```
NAME                      AGE
192.168.1.67-1721616409999   13s
```

Step 3 Cancel drainage.

```
kubectl annotate drainages 192.168.1.67-1721616409999 node.cce.io/drainage-disable=true
```

Step 4 Check the result.

```
kubectl get drainages 192.168.1.67-1721616409999 -o yaml
```

If the command output, **phase** is changed to **Cancelled**.

```
apiVersion: node.cce.io/v1
kind: Drainage
metadata:
  annotations:
    node.cce.io/drainage-disable: "true"
  creationTimestamp: "2024-07-22T03:12:56Z"
  generation: 1
  name: 192.168.1.67-1721616409999
  resourceVersion: "2689858"
```



```
  uid: 3ec131e4-0505-4c88-8255-ef9d0eb02712
spec:
  force: true
  nodeName: 192.168.1.67
  timeout: 0
status:
  conditions:
  - lastTransitionTime: "2024-07-22T03:12:56Z"
    message: start to drain node
    reason: Started
    status: "True"
    type: Started
  - lastTransitionTime: "2024-07-22T03:13:26Z"
    message: node has been drained
    reason: Succeeded
    status: "True"
    type: Finished
  - lastTransitionTime: "2024-07-22T03:37:48Z"
    message: node drainage has been cancelled
    reason: Cancelled
    status: "True"
    type: Cancelled
  phase: Cancelled
```

----End

Through APIs

Step 1 Obtain the token in the region where the cluster is located. For details, see **Authentication**.

Step 2 Based on the API format, find the URL for the node drainage API.

URL of the API for canceling node drainage:

```
https://{clusterid}.Endpoint/apis/node.cce.io/v1/drainages/{drainageName}
```

- **{clusterid}**: cluster ID, which can be obtained on the **Overview** page of the CCE console.
- **Endpoint**: endpoint of CCE in the region where the cluster is located.

  For details about its value, see **Regions and Endpoints**.

  For example, the endpoint of CCE in the **AP-Singapore** region is **cce.ap-southeast-3.myhuaweicloud.com**.
- **{drainageName}**: name of the drainage resource, which can be obtained by running the **kubectl get drainages** command.

Step 3 Use the **PATCH** request method and configure request header parameters.

```
curl --location --request PATCH 'https://{clusterid}.Endpoint/apis/node.cce.io/v1/drainages/{drainageName}' \
--header 'Content-Type: application/merge-patch+json' \
--header 'X-Auth-Token: MIIWww******' \
--data @Drainage-cancel.json
```

The following table lists the header parameters contained in the request.

Table 3-65 Request header parameters

Parameter	Mandatory	Type	Description
Content-Type	Yes	String	Message body type. The value is **application/merge-patch+json** in PATCH format.

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	Use a token to call the API. For details about how to obtain a token, see **Authentication**.

Drainage-cancel.json is located in the current directory and contains the following content:

```
{
   "metadata": {
      "annotations": {
         "node.cce.io/drainage-disable": "true"
      }
   }
}
```

----End

3.8.7 Deleting or Unsubscribing from a Node

Scenario

You can delete a pay-per-use node or unsubscribe from a yearly/monthly-billed node that is not needed from the node list.

Deleting or unsubscribing from a node in a CCE cluster will release the node and services running on the node. Drain the node and back up data before the deletion or unsubscription to prevent services running on the node from being affected.

Precautions

- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours. It is a good practice to drain the node before deletion. For details, see **Draining a Node**.
- Unexpected risks may occur during the operation. Back up data beforehand.
- It takes about 5 to 10 minutes to unsubscribe from a yearly/monthly node. During this period, the node will be unavailable. After that, the node will be automatically cleared.
- Deleting a node will cause PVC/PV data loss for the **local PV** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod is created and stays in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled.

Deleting a Pay-per-Use Node

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the target node and choose **More** > **Delete** in the **Operation** column.

Step 4 In the **Delete Node** dialog box, enter **DELETE** and click **Yes**.

NOTE

- After the node is deleted, workload pods on the node are automatically migrated to other available nodes.
- If the disks and EIPs bound to the node are important resources, unbind them first. Otherwise, they will be deleted with the node.

----End

Unsubscribing from a Yearly/Monthly Node

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the target node and choose **More** > **Unsubscribe** in the **Operation** column. In the displayed dialog box, drain the node.

After a node is selected for draining, CCE sets the node to be unschedulable so that no new pods can be scheduled to this node and then safely evicts all the pods meeting node draining requirements from the node. The time required for draining depends on the pods on the node. If it takes an excessively long time, go back to the node list and check events.

Step 4 After you confirm the unsubscription, the order page will be displayed. Confirm the unsubscription type and order amount.

NOTE

- It takes about 5 to 10 minutes to process the unsubscription. During this period, the node will be unavailable.
- When a yearly/monthly node is unsubscribed from, the node is automatically labeled with **node.cce.io/unsubscribe: true**.

----End

3.8.8 Changing the Billing Mode of a Node to Yearly/Monthly

Nodes in CCE clusters can be billed on a pay-per-use or yearly/monthly basis. A pay-per-use node can be changed to yearly/monthly-billed.

Notes and Constraints

- To change the billing mode of a node in a pay-per-use node pool to yearly/monthly, you need to upgrade the cluster to v1.19.16-r40, v1.21.11-r0, v1.23.0-r0, v1.25.4-r0, or later.
- After a node in a pay-per-use node pool is changed to a yearly/monthly node, the node does not support elastic scale-in.

Changing the Billing Mode of a Node

NOTICE

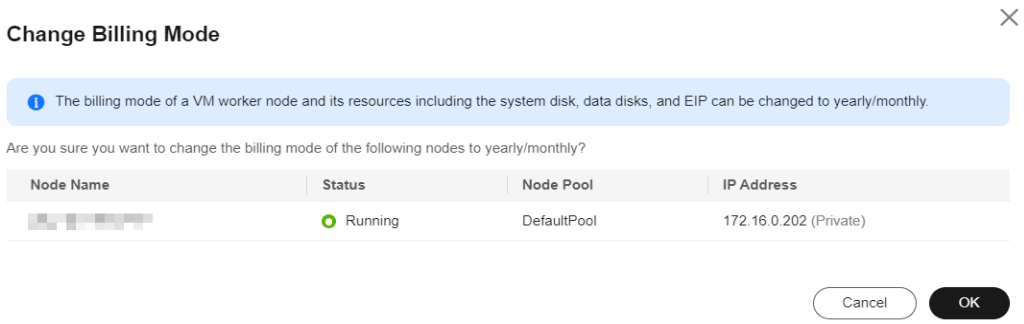
- The billing modes of resources like EVS disks and EIPs used by pay-per-use nodes cannot be changed simultaneously. For details, see **Pay-per-Use to Yearly/Monthly**.
- To change a pay-per-use node in a node pool to a yearly/monthly one, locate the target node in the node list, choose **More** > **Forbid node pool scale-in** above the list, and change the billing mode to yearly/monthly.

To change the billing mode of a pay-per-use node to yearly/monthly, perform the following steps:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. Click the **Nodes** tab, locate the row containing the target node, and choose **More** > **Change Billing Mode** in the **Operation** column.

Figure 3-10 Changing the billing mode of a node to yearly/monthly



Step 3 Click **OK**. Wait until the order is processed and the payment is complete.

----End

3.8.9 Modifying the Auto-Renewal Configuration of a Yearly/Monthly Node

After purchasing yearly/monthly nodes, you can enable auto-renewal for them or modify the existing auto-renewal configuration as needed.

Enabling Auto-Renewal

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

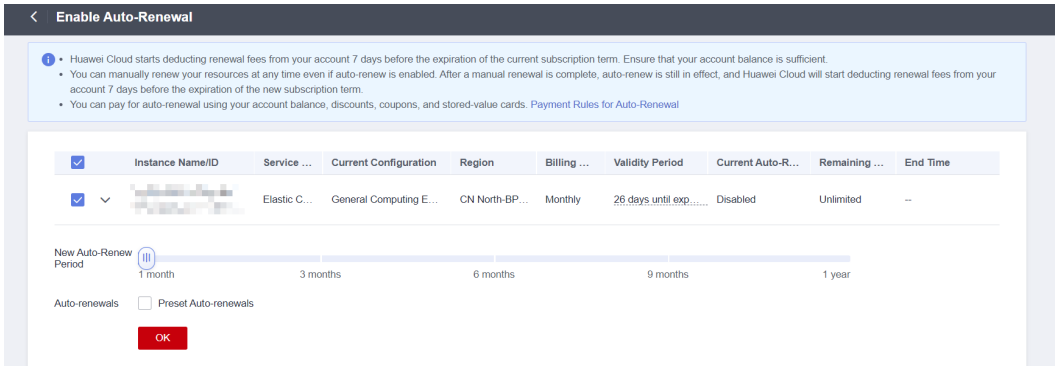
Step 2 In the navigation pane, choose **Nodes** and switch to the **Nodes** tab page.

Step 3 Choose **More** > **Enable Auto-Renewal** in the **Operation** column of a yearly/monthly node. You can also select multiple nodes at a time. To do so, choose **More** > **Enable Auto-Renewal** in the function bar on the top of the page. In the displayed dialog box, confirm the nodes for which auto-renewal is to be enabled.

Step 4 On the **Enable Auto-Renewal** page, configure auto-renewal parameters.

- **New Auto-Renew Period**: Configure the duration of each renewal. The minimum duration of each renewal is one month and the maximum duration is one year.
- **Auto-renewals**: After selecting this option, you can customize the number of renewals. If the number of renewals exceeds the value, auto-renewal will not take effect. If you do not configure the number of auto-renewals, the number of auto-renewals is not limited by default.

Figure 3-11 Configuring auto-renewal



NOTE

If resources such as EIPs are bound to a node, you can select the resources to determine whether to enable auto-renewal with the node.

Step 5 Click **OK**.

----End

Modifying Auto-Renewal

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

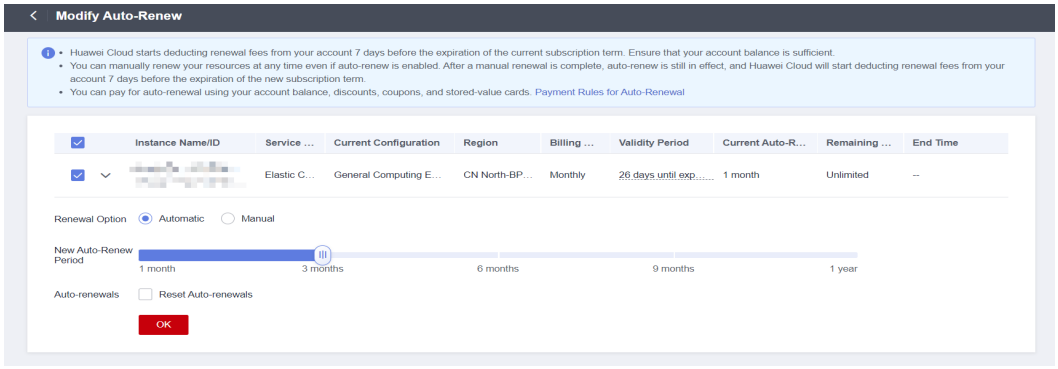
Step 2 In the navigation pane, choose **Nodes** and switch to the **Nodes** tab page.

Step 3 Choose **More** > **Change Auto-Renewal** in the **Operation** column of a yearly/monthly node. You can also select multiple nodes at a time. To do so, choose **More** > **Change Auto-Renewal** in the function bar on the top of the page. In the displayed dialog box, confirm the nodes for which auto-renewal is to be modified.

Step 4 On the **Modify Auto-Renew** page, modify auto-renewal parameters.

- **Renewal Option**: If auto-renewal is enabled for a node, **Automatic** is selected by default. To disable auto-renewal, select **Manual**.
- **New Auto-Renew Period**: Configure the duration of each renewal. The minimum duration of each renewal is one month and the maximum duration is one year.
- **Auto-renewals**: After selecting this option, you can customize the number of renewals. If the number of renewals exceeds the value, auto-renewal will not take effect. If you do not configure the number of auto-renewals, the number of auto-renewals is not limited by default.

Figure 3-12 Modifying auto-renewal



Step 5 Click **OK**.

----End

3.8.10 Stopping a Node

Scenario

When a node in the cluster is stopped, all services on that node will also be stopped, and the node will no longer be available for scheduling. Check if your services will be affected before stopping a node.

Most nodes are no longer billed after they are stopped, excluding certain types of ECSs (ones with local disks attached, such as disk-intensive and ultra-high I/O ECSs). For details, see **ECS Billing**.

Precautions

- Stopping a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours.
- Unexpected risks may occur during the operation. Back up data beforehand.

Procedure

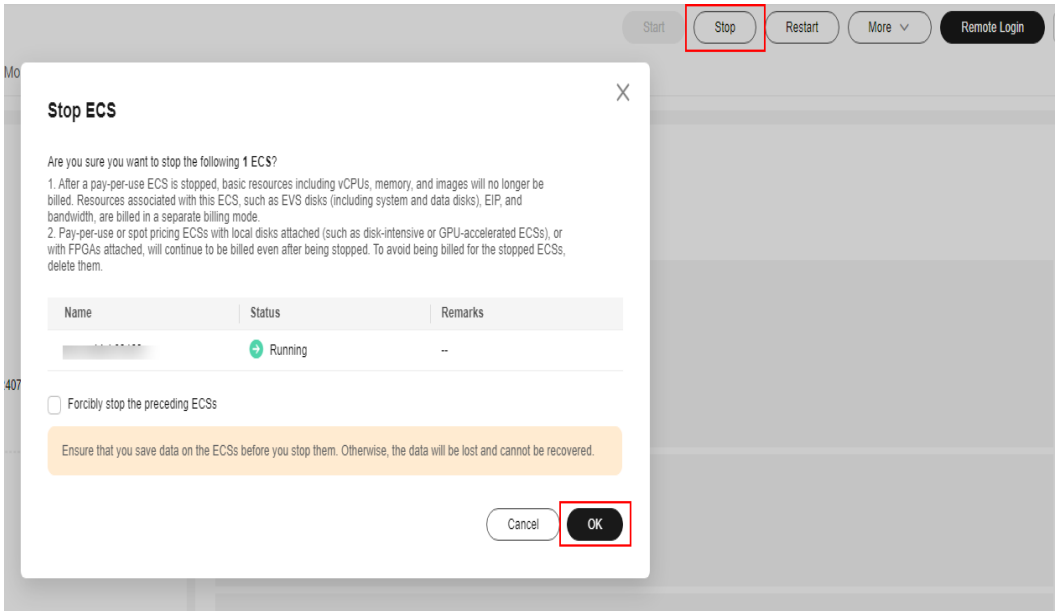
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 Locate the target node and click its name.

Step 4 In the upper right corner of the ECS details page, click **Stop**. In the displayed dialog box, click **OK**.

Figure 3-13 ECS details page



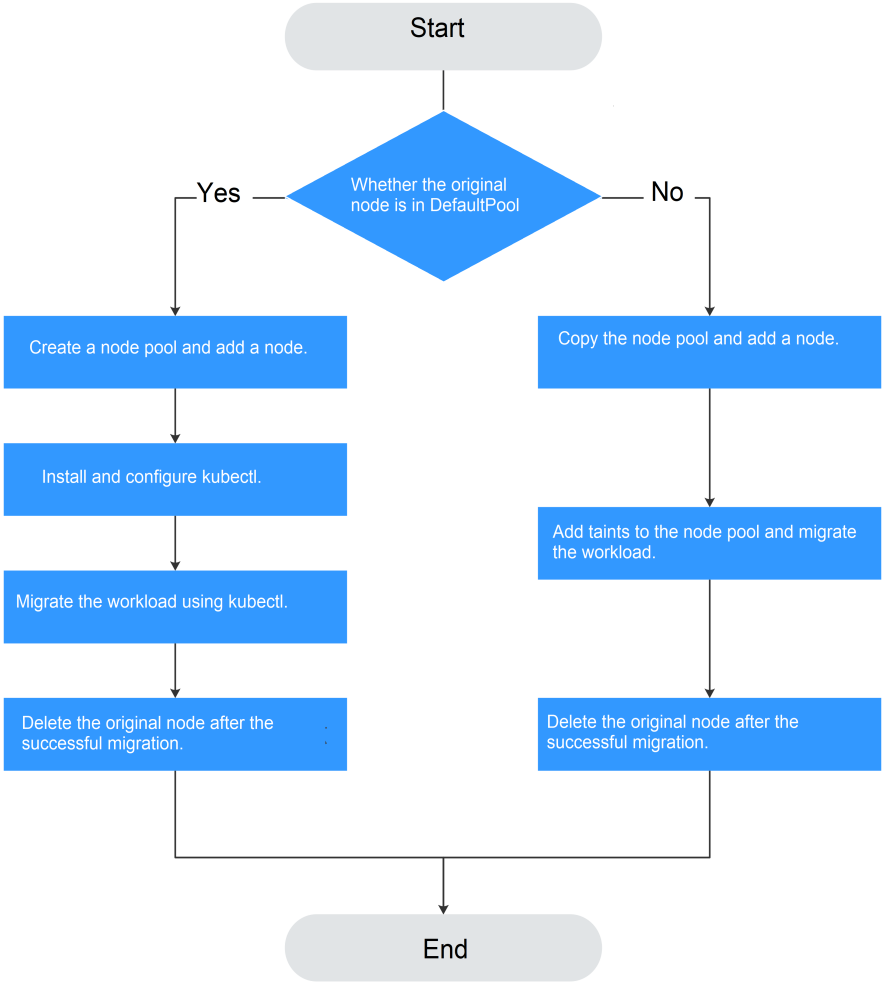
----End

3.8.11 Performing Rolling Upgrade for Nodes

Scenario

In a rolling upgrade, a new node is created, existing workloads are migrated to the new node, and then the old node is deleted. **Figure 3-14** shows the migration process.

Figure 3-14 Workload migration



Notes and Constraints

- The original node and the target node to which the workload is to be migrated must be in the same cluster.
- The cluster must be of v1.13.10 or later.
- The default node pool does not support this configuration.

Procedure If the Original Node Is in the Default Pool

Step 1 Create a node pool. For details, see **Creating a Node Pool**.

Step 2 On the node pool list page, click **View Node** in the **Operation** column of the target node pool. The IP address of the new node is displayed in the node list.

Step 3 Install and configure kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Step 4 Migrate the workload.

1. Add a taint to the node where the workload needs to be migrated out.

   kubectl taint node *[node]* **key=value:***[effect]*

In the preceding command, *[node]* indicates the IP address of the node where the workload to be migrated is located. The value of *[effect]* can be **NoSchedule**, **PreferNoSchedule**, or **NoExecute**. In this example, set this parameter to **NoSchedule**.

- **NoSchedule**: Pods that do not tolerate this taint are not scheduled on the node; existing pods are not evicted from the node.
- **PreferNoSchedule**: Kubernetes tries to avoid scheduling pods that do not tolerate this taint onto the node.
- **NoExecute**: A pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.

NOTE

To reset a taint, run the **kubectl taint node *[node]* key:*[effect]*- command** to remove the taint.

2. Safely evicts the workload on the node.

   kubectl drain *[node]*

   In the preceding command, *[node]* indicates the IP address of the node where the workload to be migrated is located.

3. In the navigation pane of the CCE console, choose **Workloads** > **Deployments**. In the workload list, the status of the workload to be migrated changes from **Running** to **Unready**. If the workload status changes to **Running** again, the migration is successful.

NOTE

During workload migration, if node affinity is configured for the workload, the workload keeps displaying a message indicating that the workload is not ready. In this case, click the workload name to go to the workload details page. On the **Scheduling Policies** tab page, delete the affinity configuration of the original node and configure the affinity and anti-affinity policies of the new node. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.

After the workload is migrated, you can view that the workload has been migrated to the node created in **Step 1** on the **Pods** tab page of the workload details page.

Step 5 Delete the original node.

After the workload is successfully migrated and runs properly, delete the original node.

----End

Procedure If the Original Node Is Not in the Default Pool

Step 1 Copy the node pool and add nodes to it. For details, see **Copying a Node Pool**.

Step 2 Click **View Node** in the **Operation** column of the node pool. The IP address of the new node is displayed in the node list.

Step 3 Migrate the workload.

1. Click **Edit** on the right of original node pool and configure **Taints**.

2. Enter the key and value of a taint. The options of **Effect** are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. Select **NoExecute** and click **Add**.
   - **NoSchedule**: Pods that do not tolerate this taint are not scheduled on the node; existing pods are not evicted from the node.
   - **PreferNoSchedule**: Kubernetes tries to avoid scheduling pods that do not tolerate this taint onto the node.
   - **NoExecute**: A pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.

   NOTE

   To reset the taint, delete the configured one.
3. Click **OK**.
4. In the navigation pane of the CCE console, choose **Workloads** > **Deployments**. In the workload list, the status of the workload to be migrated changes from **Running** to **Unready**. If the workload status changes to **Running** again, the migration is successful.

NOTE

During workload migration, if node affinity is configured for the workload, the workload keeps displaying a message indicating that the workload is not ready. In this case, click the workload name to go to the workload details page. On the **Scheduling Policies** tab page, delete the affinity configuration of the original node and configure the affinity and anti-affinity policies of the new node. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.

After the workload is migrated, you can view that the workload has been migrated to the node created in **Step 1** on the **Pods** tab page of the workload details page.

Step 4 Delete the original node.

After the workload is successfully migrated and runs properly, delete the original node.

----End

3.9 Node O&M

3.9.1 Node Resource Reservation Policy

Some node resources are used to run mandatory Kubernetes system components and resources to make the node as part of your cluster. Therefore, the total number of node resources and the number of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components.

To ensure node stability, a certain number of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

CCE calculates the resources that can be allocated to user nodes as follows:

Allocatable resources = Total amount - Reserved amount - Eviction threshold

The memory eviction threshold is fixed at 100 MiB.

NOTE

Total amount indicates the available memory of the ECS, excluding the memory used by system components. Therefore, the total amount is slightly less than the memory of the node flavor. For details, see **Why Is the Memory of an ECS Obtained by Running the free Command Inconsistent with the Actual Memory?**

When the memory consumed by all pods on a node increases, the following behaviors may occur:

1. When the available memory of the node is lower than the eviction threshold, kubelet is triggered to evict the pod. For details about the eviction threshold in Kubernetes, see **Node-pressure Eviction**.
2. If a node triggers an OS memory insufficiency event (OOM) before kubelet reclaims memory, the system terminates the container. However, different from pod eviction, kubelet restarts the container based on the RestartPolicy of the pod.

Rules v1 for Reserving Node Memory

NOTE

For clusters of versions earlier than **v1.21.4-r0** and **v1.23.3-r0**, the v1 model is used for node memory reservation. For clusters of **v1.21.4-r0**, **v1.23.3-r0**, or later, the node memory reservation model is optimized to v2. For details, see **Rules for Reserving Node Memory v2**.

Upgrading the cluster to **v1.21.4-r0**, **v1.23.3-r0**, or a later version may result in the eviction of the workload on a node that is already fully utilized. This is because system components will need more reserved resources.

You can use the following formula calculate how much memory you should reserve for running containers on a node:

Total reserved amount = **Reserved memory for system components** + **Reserved memory for kubelet to manage pods**

Table 3-66 Reservation rules for system components

Total Memory (TM)	Reserved Memory for System Components
TM ≤ 8 GiB	0 MiB
8 GiB < TM ≤ 16 GiB	[(TM – 8 GiB) x 1024 x 10%] MiB
16 GiB < TM ≤ 128 GiB	[8 GiB x 1024 x 10% + (TM – 16 GiB) x 1024 x 6%] MiB
TM > 128 GiB	(8 GiB x 1024 x 10% + 112 GiB x 1024 x 6% + (TM – 128 GiB) x 1024 x 2%) MiB

Table 3-67 Reservation rules for kubelet

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
TM ≤ 2 GiB	None	TM x 25%
TM > 2 GiB	0 < Max. pods on a node ≤ 16	700 MiB
	16 < Max. pods on a node ≤ 32	[700 + (Max. pods on a node – 16) x 18.75] MiB
	32 < Max. pods on a node ≤ 64	[1024 + (Max. pods on a node – 32) x 6.25] MiB
	64 < Max. pods on a node ≤ 128	[1230 + (Max. pods on a node – 64) x 7.80] MiB
	Max. pods on a node > 128	[1740 + (Max. pods on a node – 128) x 11.20] MiB

NOTICE

For a small-capacity node, adjust the maximum number of instances based on the site requirements. Alternatively, when creating a node on the CCE console, you can adjust the maximum number of instances for the node based on the node specifications.

Rules for Reserving Node Memory v2

For clusters of **v1.21.4-r0**, **v1.23.3-r0**, or later, the node memory reservation model is optimized to v2 and can be dynamically adjusted using the node pool parameters **kube-reserved-mem** and **system-reserved-mem**. For details, see **Modifying Node Pool Configurations**.

The total reserved node memory of the v2 model is equal to the sum of that reserved for the OS and that reserved for CCE to manage pods.

Reserved memory includes basic and floating parts. For the OS, the floating memory depends on the node specifications. For CCE, the floating memory depends on the number of pods on a node.

Table 3-68 Rules for reserving node memory v2

Reserved for	Basic/Floating	Reservation	Used by
OS	Basic	Fixed at 400 MiB	OS service components such as sshd and systemd-journald.

Reserved for	Basic/Floating	Reservation	Used by
	Floating (depending on the node memory)	25MiB/GiB	Kernel
CCE	Basic	Fixed at 500 MiB	Container engine components, such as kubelet and kube-proxy, when the node is unloaded
	Floating (depending on the number of pods on the node)	Docker: 20 MiB/Pod containerd: 5 MiB/Pod	Container engine components when the number of pods increases **NOTE** When the v2 model reserves memory for a node by default, the default maximum number of pods is estimated based on the memory. For details, see **Table 3-72**.

Rules for Reserving Node CPU

Table 3-69 Node CPU reservation rules

Total CPU Cores (Total)	Reserved CPU Cores
Total ≤ 1 core	Total x 6%
1 core < Total ≤ 2 cores	1 core x 6% + (Total – 1 core) x 1%
2 cores < Total ≤ 4 cores	1 core x 6% + 1 core x 1% + (Total – 2 cores) x 0.5%
Total > 4 cores	1 core x 6% + 1 core x 1% + 2 cores x 0.5% + (Total – 4 cores) x 0.25%

Rules for CCE to Reserve Data Disks on Nodes

CCE uses Logical Volume Manager (LVM) to manage disks. LVM creates a metadata area on a disk to store logical and physical volumes, occupying 4 MiB space. Therefore, the actual available disk space of a node is equal to the disk size minus 4 MiB.

3.9.2 Space Allocation of a Data Disk

This section describes how to allocate data disk space to nodes so that you can configure the data disk space accordingly.

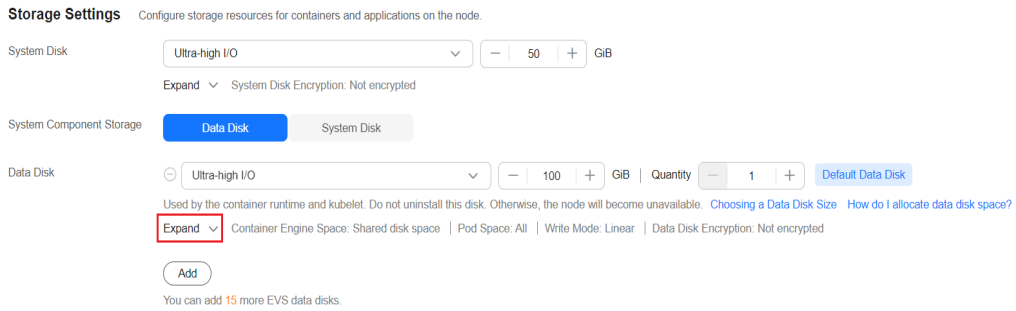
Allocating Default Data Disk Space

NOTE

- In clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.29.4-r0, or v1.28.8-r0, a default data disk will be added to the node for the container runtime and kubelet components. You can customize the space allocation of the default data disk.
- In clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later, a default data disk will be added to the node for the container runtime and kubelet components only if **System Component Storage** is set to **Data Disk**. You can customize the space allocation of the default data disk.

When creating a node, you can customize Data Disk Space Allocation in the expanded area of **Data Disk**.

Figure 3-15 Allocating data disk space



- **Space Allocation for Container Engines**
  - Specified disk space: CCE divides the data disk space for two parts by default. One part is used to store the Docker/containerd working directories, container image data, and image metadata. The other is reserved for kubelet and emptyDir volumes. The available container engine space affects image pulls and container startup and running.
    - Container engine and container image space (90% by default): stores the container runtime working directories, container image data, and image metadata.
    - kubelet and emptyDir space (10% by default): stores pod configuration files, secrets, and mounted storage such as emptyDir volumes.

      NOTE

      If the sum of the container engine and container image space and the kubelet and emptyDir space is less than 100%, the remaining space will be allocated for user data. You can mount the storage volume to a service path. Do not leave the path empty or set it to a key OS path such as the root directory.
  - Shared disk space: In clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions, CCE allows **a container engine (Docker/containerd) and kubelet components to share data disk space**.
- **Space Allocation for Pods**: indicates the basesize of a pod. You can set an upper limit for the disk space occupied by each workload pod (including the space occupied by container images). This setting prevents the pods from

taking all the disk space available, which may cause service exceptions. It is recommended that the value is less than or equal to 80% of the container engine space. This parameter is related to the node OS and container storage Rootfs and is not supported in some scenarios. For details, see **Mapping Between OS and Container Storage Rootfs**.

- Write Mode
  - **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
  - **Striped**: available only if there are at least two data disks. A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out.

Space Allocation for Container Engines

For nodes using a data disk shared between a container engine and kubelet components, the container storage Rootfs is of the **OverlayFS** type. For details about data disk space allocation, see **Data Disk Shared Between a Container Engine and kubelet Components**.

For a node using a non-shared data disk (100 GiB for example), the division of the disk space varies depending on the container storage Rootfs type **Device Mapper** or **OverlayFS**. For details about the container storage Rootfs corresponding to different OSs, see **Mapping Between OS and Container Storage Rootfs**.

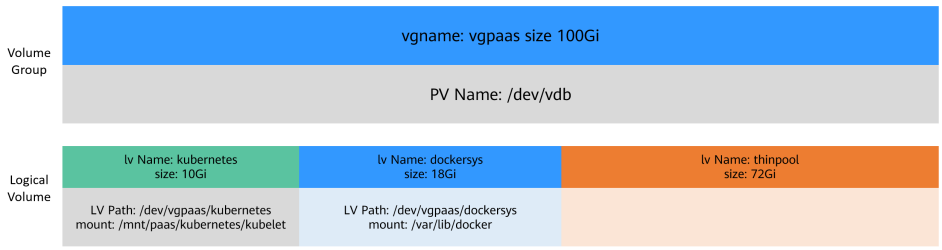
- **Rootfs (Device Mapper)**

  By default, the container engine and image space, occupying 90% of the data disk, can be divided into the following two parts:

  - The **/var/lib/docker** directory is used as the Docker working directory and occupies 20% of the container engine and container image space by default. (Space size of the **/var/lib/docker** directory = **Data disk space x 90% x 20%**)
  - The thin pool is used to store container image data, image metadata, and container data, and occupies 80% of the container engine and container image space by default. (Thin pool space = **Data disk space x 90% x 80%**)

    The thin pool is dynamically mounted. You can view it by running the **lsblk** command on a node, but not the **df -h** command.

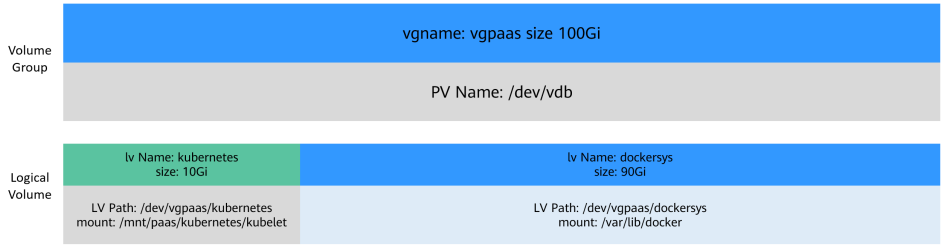
  Figure 3-16 Space allocation for container engines of Device Mapper



- **Rootfs (OverlayFS)**

No separate thin pool. The entire container engine and container image space (90% of the data disk by default) are in the **/var/lib/docker** directory.

Figure 3-17 Space allocation for container engines of OverlayFS



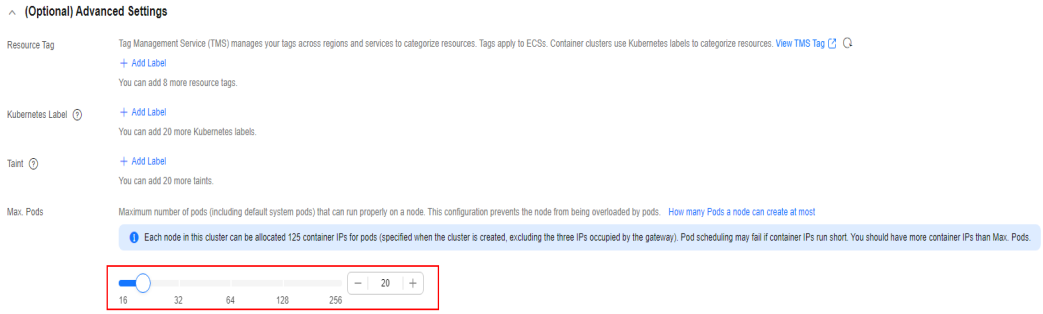
Space Allocation for Pods

The customized pod container space (basesize) is related to the node OS and container storage Rootfs. For details about the container storage Rootfs, see **Mapping Between OS and Container Storage Rootfs**.

- Device Mapper supports custom pod basesize. The default value is 10 GiB.
- In OverlayFS mode, the pod container space is not limited by default.

When configuring **basesize**, consider the maximum number of pods allowed on one node. The container engine space should be greater than the total disk space used by containers. Formula: **Container engine space and container image space (90% by default)** > **Number of containers** x **basesize**. Otherwise, the container engine space allocated to the node may be insufficient and the container cannot be started.

Figure 3-18 Configuring the maximum number of pods during node creation



For nodes that support **basesize**, when Device Mapper is used, although you can limit the size of the **/home** directory of a single container (to 10 GiB by default), all containers on the node still share the thin pool of the node for storage. They are not completely isolated. When the sum of the thin pool space used by certain containers reaches the upper limit, other containers cannot run properly.

In addition, after a file is deleted in the **/home** directory of the container, the thin pool space occupied by the file is not released immediately. Therefore, even if **basesize** is set to 10 GiB, the thin pool space occupied by files keeps increasing until 10 GiB when files are created in the container. The space released after file deletion will be reused but after a while. If **the number of containers on the node multiplied by basesize** is greater than the thin pool space size of the node, there is a possibility that the thin pool space has been used up.

Mapping Between OS and Container Storage Rootfs

Table 3-70 Node OSs and container engines in CCE clusters

OS	Container Storage Rootfs	Custom Basesize
CentOS 7.x	Clusters of v1.19.16 and earlier use Device Mapper. Clusters of v1.19.16 and later use OverlayFS.	Supported when Rootfs is set to Device Mapper and the runtime is Docker. The default value is 10 GiB. If Rootfs is set to OverlayFS, the basesize cannot be customized.
EulerOS 2.3	Device Mapper	Supported only when the runtime is Docker. The default value is 10 GiB.
EulerOS 2.5	Device Mapper	Supported only when the runtime is Docker. The default value is 10 GiB.
EulerOS 2.8	Clusters of v1.19.16-r2 and earlier use Device Mapper. Clusters of v1.19.16-r2 and later use OverlayFS.	Supported when Rootfs is set to Device Mapper and the runtime is Docker. The default value is 10 GiB. Supported when Rootfs is set to OverlayFS and the runtime is Docker. There are no limits by default.
EulerOS 2.9	OverlayFS	Supported by Docker clusters of v1.19.16-r0, v1.21.3-r0, v1.23.3-r0, or later. There are no limits by default. Supported by containerd clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. There are no limits by default. Not supported by clusters of a version earlier than v1.19.16-r0, v1.21.3-r0, or v1.23.3-r0.
EulerOS 2.10	OverlayFS	Supported only by Docker clusters of a version earlier than v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, or v1.28.4-r0. There are no limits by default. Supported by both Docker and containerd clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. There are no limits by default.

OS	Container Storage Rootfs	Custom Basesize
Ubuntu 18.04	OverlayFS	Not supported
Ubuntu 22.04	OverlayFS	Not supported
Huawei Cloud EulerOS 1.1	OverlayFS	Not supported
Huawei Cloud EulerOS 2.0	OverlayFS	Supported only by Docker clusters of a version earlier than v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, or v1.28.4-r0. There are no limits by default. Supported by both Docker and containerd clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. There are no limits by default.

Table 3-71 Node OSs and container engines in CCE Turbo clusters

OS	Container Storage Rootfs	Custom Basesize
CentOS 7.x	OverlayFS	Not supported
Ubuntu 18.04	OverlayFS	Not supported
Ubuntu 22.04	OverlayFS	Not supported
EulerOS 2.9	OverlayFS	Supported when Rootfs is set to OverlayFS and the runtime is Docker. There are no limits by default. Supported by both Docker and containerd clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.
EulerOS 2.10	Device Mapper for ECSs (PM)	Supported when Rootfs is set to Device Mapper and the runtime is containerd. The default value is 10 GiB.
Huawei Cloud EulerOS 1.1	OverlayFS	Not supported

OS	Container Storage Rootfs	Custom Basesize
Huawei Cloud EulerOS 2.0	OverlayFS	Supported only by Docker clusters of a version earlier than v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, or v1.28.4-r0. There are no limits by default. Supported by both Docker and containerd clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. There are no limits by default.

Garbage Collection Policies for Container Images

When the container engine space is insufficient, image garbage collection is triggered.

The policy for garbage collecting images takes two factors into consideration: **HighThresholdPercent** and **LowThresholdPercent**. Disk usage exceeding the high threshold (default: 80%) will trigger garbage collection. The garbage collection will delete least recently used images until the low threshold (default: 70%) is met.

Recommended Configuration for the Container Engine Space

- The container engine space should be greater than the total disk space used by containers. Formula: **Container engine space** > **Number of containers** x **basesize**
- You are advised to create and delete files of containerized services in local storage volumes (such as emptyDir and hostPath volumes) or cloud storage directories mounted to the containers. In this way, the thin pool space is not occupied. emptyDir volumes occupy the kubelet space. Therefore, properly plan the size of the kubelet space.
- You can deploy services on nodes that use the OverlayFS (for details, see **Mapping Between OS and Container Storage Rootfs**) so that the disk space occupied by files created or deleted in containers can be released immediately.

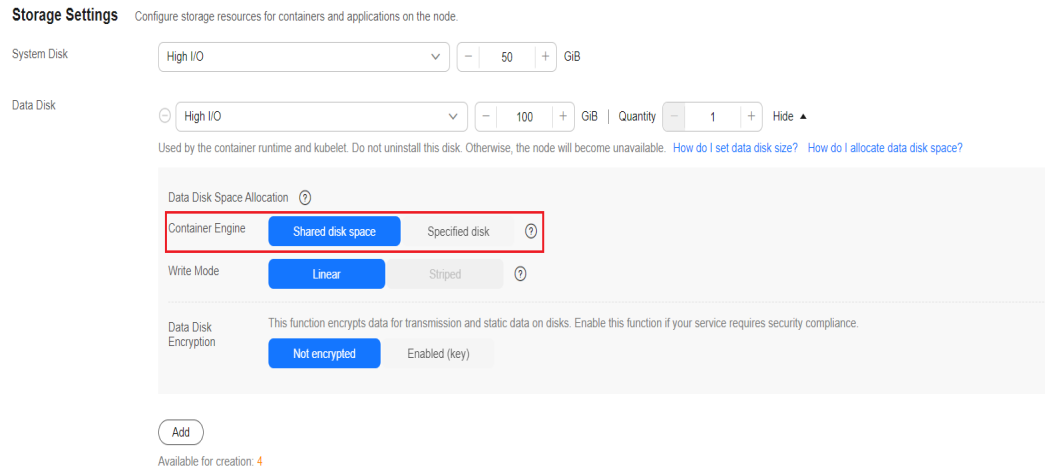
Data Disk Shared Between a Container Engine and kubelet Components

Docker/containerd and kubelet components share the space of a data disk.

NOTICE

- This function is available only to clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
- If Rootfs is set to OverlayFS, shared data disks are supported. If Rootfs is set to Device Mapper, shared data disks are not supported.
- If you have installed an NPD add-on in the cluster, upgrade the add-on to v1.18.10 or later. Otherwise, false alarms will be generated.
- If you have installed a log-agent add-on in the cluster, upgrade the add-on to v1.3.0 or later. Otherwise, log collection will be affected.
- If you have installed ICAgent in the cluster, upgrade it to v5.12.140 or later. Otherwise, log collection will be affected. For details about how to view or upgrade an ICAgent version, see **CCE Access**.

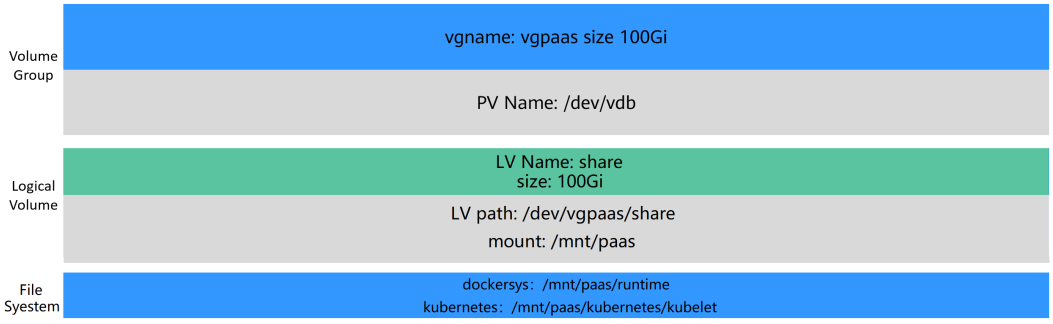
Figure 3-19 Configuration for sharing disk space



For nodes using a shared data disk, the container storage Rootfs is of the **OverlayFS** type. After such a node is created, the data disk space (for example, 100 GiB) will not be divided for the container engines, container images, and kubelet components. The data disk is mounted to **/mnt/paas**, and the storage space is divided using two file systems.

- dockersys: /mnt/paas/runtime
- Kubernetes: /mnt/paas/kubernetes/kubelet

Figure 3-20 Allocating the storage space of a shared data disk



Helpful Links

How Do I Expand the Storage Capacity of a Container?

Expanding the Disk Capacity of a Node in a CCE Cluster

3.9.3 Maximum Number of Pods That Can Be Created on a Node

Calculation of the Maximum Number of Pods on a Node

The maximum number of pods that can be created on a node is calculated based on the cluster type:

Network Model	Maximum Number of Pods on a Node	Recommended Configuration
Tunnel network	**Maximum number of pods on a node**	None
VPC network	The smaller value between the **maximum number of pods on a node** and the **number of container IP addresses that can be allocated on a node**	To ensure new pods run smoothly on a node, make sure that the number of pods on the node does not exceed the number of container IP addresses that can be assigned to it. If there are not enough container IP addresses available on the node, the new pods may not function properly.
Cloud Native Network 2.0 (for CCE Turbo clusters)	The smaller value between the **maximum number of pods on a node** and the **number of ENIs on a node in a CCE Turbo cluster**	To ensure new pods run smoothly on a node, make sure that the number of pods on the node does not exceed the number of ENIs on it. If there are not enough ENIs available on the node, the new pods may not function properly.

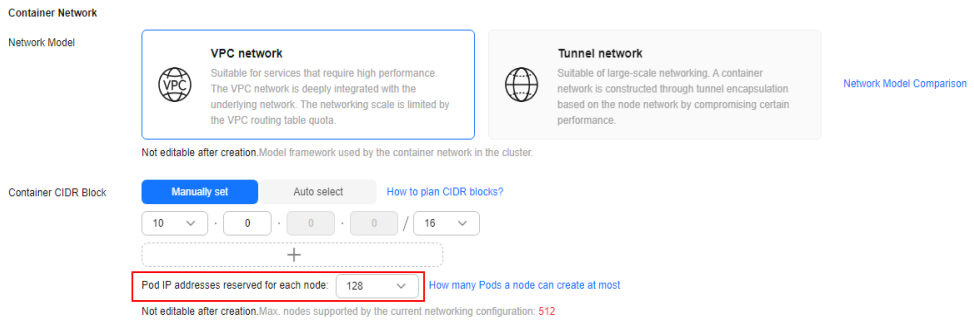
Number of Allocatable Container IP Addresses on a Node

When creating a cluster in the VPC network model, follow the **rules for managing container IP addresses** and specify the number of container IP addresses that can be allocated to each node using alpha.cce/fixPoolMask.

The maximum number of pods that can be created on a node is determined by the number of container IP addresses that can be allocated to it. In a **container network**, each pod needs its own IP address. If there are not enough pre-allocated

container IP addresses on the node, pods cannot be created. If **hostNetwork: true** is configured in the YAML file, pods will use the **host network** instead of the allocatable container IP addresses. For details, see **Pod IP Address Allocation Differences Between the Container Network and Host Network**.

Figure 3-21 Specifying the number of allocatable container IP addresses on a node in the VPC network model



By default, a node occupies three container IP addresses (network address, gateway address, and broadcast address). Therefore, the number of container IP addresses that can be allocated to a node equals the number of selected container IP addresses minus 3. For example, in the preceding figure, the number of container IP addresses that can be allocated to a node is 125 (128 – 3).

Maximum Number of Pods on a Node

When creating a node, you can configure the maximum number of pods (maxPods) that can be created on the node. This parameter is a configuration item of kubelet and determines the maximum number of pods that can be created by kubelet.

NOTICE

For nodes in the default node pool (**DefaultPool**), the maximum number of pods cannot be changed after the nodes are created.

After a node in a custom node pool is created, you can modify the **max-pods** parameter in the node pool configuration to change the maximum number of pods on the node. For details, see **Configuring a Node Pool**.

By default, the maximum number of pods on a node can be adjusted to 256. To increase the deployment density on a node, submit a **service ticket** to increase the maximum number of pods on a node, which can be 512.

Figure 3-22 Configuring the maximum number of pods during node creation

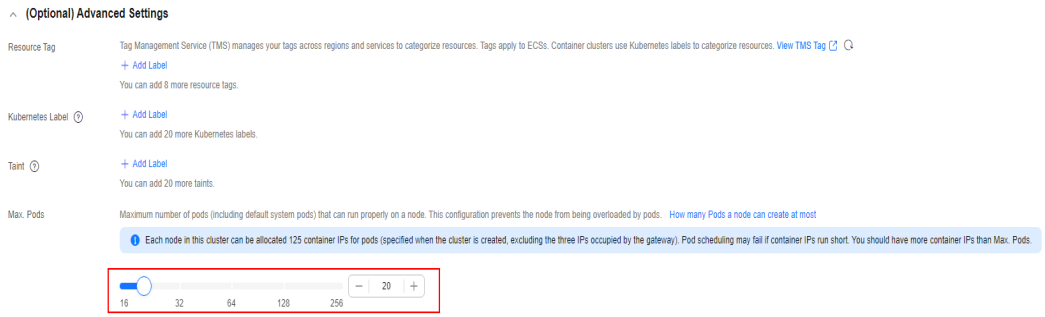


Table 3-72 lists the default maximum number of pods on a node based on node specifications.

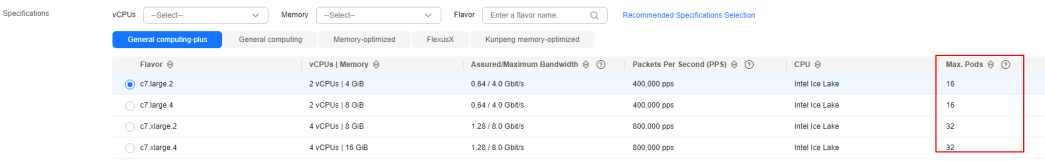
Table 3-72 Default maximum number of pods on a node

Memory	Max. Pods
4 GB	20
8 GB	40
16 GB	60
32 GB	80
64 GB or above	110

Number of Node ENIs (Available Only in CCE Turbo Clusters)

In a CCE Turbo cluster, ECS nodes use sub-ENIs and BMS nodes use ENIs. The maximum number of pods that can be created on a node depends on the number of ENIs that can be used by the node.

Figure 3-23 Node ENIs



Pod IP Address Allocation Differences Between the Container Network and Host Network

When creating a pod, you can select the container network or host network for the pod.

- Container network (default): **Each pod is assigned an IP address by the cluster networking add-ons, which occupies the IP addresses of the container network**.

- Host network: Pods with **hostNetwork: true** configured directly use the network of the host. For details, see **Configuring hostNetwork for Pods**. After the configuration, the pods use the ports on the host and their IP address is identical to that of the host, **without relying on the IP addresses of the container network**. When using a host network, avoid conflicts between the pod ports and service ports on the host. Use a host network only if a particular application requires access to a specific port on the host.

3.9.4 Differences in kubelet and Runtime Component Configurations Between CCE and the Native Community

To maintain the stability of nodes, CCE stores Kubernetes and container runtime components on separate data disks. Kubernetes uses the **/mnt/paas/kubernetes** directory, and the container runtime uses the **/mnt/paas/runtime** directory. The paths to these directories are the same as the default paths used in the community through soft links.

Differences in Default Paths to kubelet and Runtime

Path Name	Kubernetes Native Path	Path on CCE Nodes
kubelet **root-dir**	/var/lib/kubelet	/mnt/paas/kubernetes/kubelet
kubelet	/var/lib/kubelet	/mnt/paas/kubernetes/kubelet The soft link from **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet** is also available.
Container runtime (Docker)	/var/lib/docker	• **Data Disk Space Allocation** set to shared: /mnt/paas/runtime The soft link from **/var/lib/docker** to **/mnt/paas/runtime** is also available. • **Data Disk Space Allocation** set to specific: **/var/lib/docker**, which is identical to the Kubernetes native path
Container runtime (containerd)	/var/lib/containerd	• **Data Disk Space Allocation** set to shared: /mnt/paas/runtime The soft link from **/var/lib/containerd** to **/mnt/paas/runtime** is also available. • **Data Disk Space Allocation** set to specific: **/var/lib/containerd**, which is identical to the Kubernetes native path

Path Name	Kubernetes Native Path	Path on CCE Nodes
containerd logs	/var/log/pods	/var/lib/containerd/container_logs The soft link from **/var/log/pods** to **/var/lib/containerd/container_logs** is also available.

NOTE

The differences in default paths to kubelet and runtime between CCE and the native community may have the following impacts:

- If a soft link file is mounted into a container, the real path that the soft link file points to cannot be accessed.

  For example, if **/var/log** is mounted to **/mnt/log** in a container through a hostPath, **/mnt/log/pods** is an abnormal soft link file in the container, and the actual data in **/var/log/pods** cannot be accessed.

  Mount the actual file path to the container to prevent a file read failure caused by soft links.

- The kubelet **root-dir** path (**/mnt/paas/kubernetes/kubelet**) is inconsistent with the community path (**/var/lib/kubelet**). If the container mount path of a third-party CSI add-on is the community path, the file mounting does not take effect.

  For example, when Vault (an open-source third-party add-on) uses secrets-store-csi-driver to mount a secret, if the **root-dir** path to the add-on is inconsistent with the CCE configuration path (the default value of the add-on is the same as the community path **/var/lib/kubelet**), the Vault secret cannot be obtained in the container.

  This is because CSI uses mountPropagation to mount volumes to target containers. kubelet includes the mount path (**/mnt/paas/kubernetes/kubelet/pods/{podID}/volume/...**) related to kubelet **root-dir** into the CSI request for mounting a volume. When the CSI container mounts a volume to the mount path requested by kubelet CSI, if the mount path is irrelevant to the hostPath, the mount propagation will be invalid. For example, the CSI container mounts host **/var/lib/kubelet** to container **/var/lib/kubelet**, and **/mnt/paas/kubernetes/kubelet/pods/{podID}/volume/...** in the CSI container is irrelevant to the hostPath.

  Update the **root-dir** path in CSI to match the kubelet **root-dir** path on the current CCE node.

Changing the Soft Link Mode of CCE kubelet and Runtime Components to Mounting

To ensure compatibility with third-party open-source software, you can change the soft link mode of the kubelet and runtime components in CCE clusters of v1.23.18-r0, v1.25.13-r0, v1.27-10-r0, v1.28.8-r0, v1.29.4-r0, or later versions by configuring the cluster or node pool. The node pool configuration takes priority over the cluster configuration.

This operation can only be performed through APIs.

- Example of **creating a cluster**:

```
POST /api/v3/projects/{project_id}/clusters
{
    "kind": "Cluster",
    "apiVersion": "v3",
    "metadata": {
```



```
        "name": "xxxxxxxxxx",
        ...
    },
    "spec": {
        ...
        "configurationsOverride": [
            {
                "name": "node-config",
                "configurations": [
                    {
                        "name": "enable-mount-bind",
                        "value": true
                    }
                ]
            }
        ],
        ...
    }
}
```

- Example of **creating a node pool**:

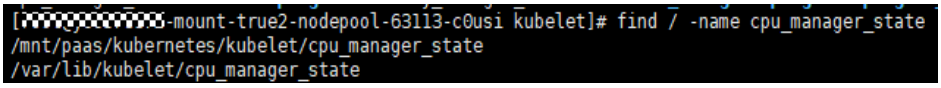
```
{
    "kind": "NodePool",
    "apiVersion": "v3",
    "metadata": {
        "name": "xxxxxxxxx",
        ...
    },
    "spec": {
        ...
        "nodeTemplate": {
            "configurationsOverride": [{
                "name": "node-config",
                "configurations": [{
                    "name": "enable-mount-bind",
                    "value": true
                }]
            }],
            ...
        }
    }
}
```


NOTE

Changing the soft link mode of CCE kubelet and runtime components to mounting may have the following impacts:

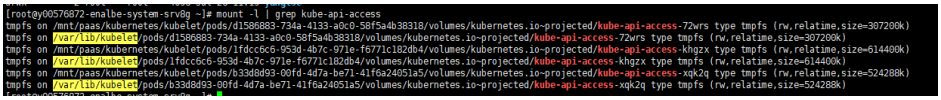
- After the change, the source directory remains unchanged, but the target soft link file is replaced with the mounted directory. This ensures that the container that was initially mounted to the soft link can still be accessed correctly without any further adjustments needed.
- After the soft link mode is changed to mounting, the source and target directories will have two identical path structures. If you search for a file in the public parent directory, you may find the same file in different paths.

  For example, a file may be found in both **/var/lib/kubelet** and **/mnt/paas/kubernetes/kubelet**.

- There will be two identical mounting records in the source and destination directories. If your service code relies on the mounting records, evaluate how your services will be affected.

  Example

3.9.5 Migrating Nodes from Docker to containerd

As of Kubernetes v1.24, **dockershim has been deprecated**. To maintain compatibility and ensure continued support for future Kubernetes releases, switch your node's container runtime from Docker to the officially endorsed containerd.

Prerequisites

- At least one cluster that supports containerd nodes has been created. For details, see **Mapping Between Node OSs and Container Engines**.
- There is a Docker node or Docker node pool in your cluster.

Precautions

- Theoretically, migration during container running will interrupt services for a short period of time. Therefore, it is strongly recommended that the services to be migrated have been deployed as multi-instance. In addition, you are advised to test the migration impact in the test environment to minimize potential risks.
- containerd cannot build images. Do not use the **docker build** command to build images on containerd nodes. For other differences between Docker and containerd, see **Container Engines**.

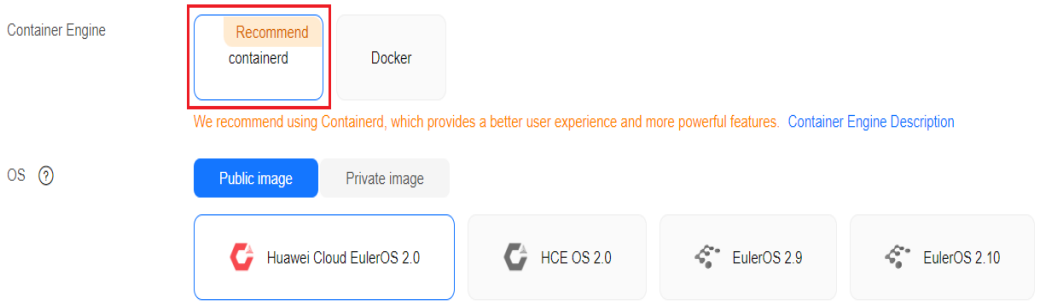
Procedure for Migrating Nodes in the Default Node Pool

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab.

Step 3 In the node list, select one or more nodes to be reset and choose **More** > **Reset Node** in the **Operation** column.

Step 4 Set **Container Engine** to **containerd**. You can adjust other parameters as required or retain them as set during creation.



Step 5 If the node status is **Installing**, the node is being reset.

When the node status is **Running**, you can see that the node version is switched to containerd. You can log in to the node and run containerd commands such as **crictl** to view information about the containers running on the node.

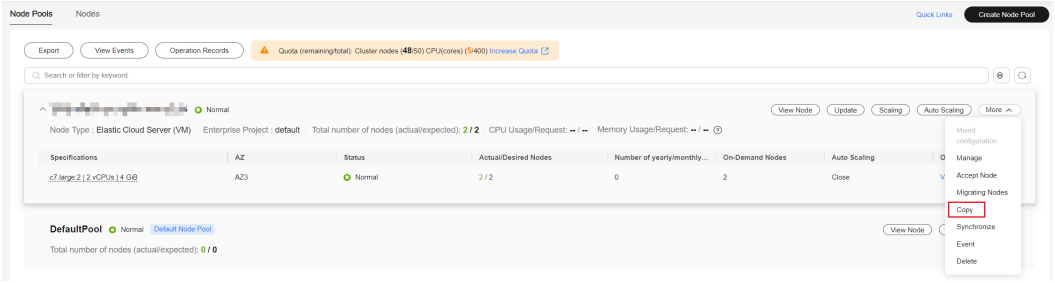
----End

Procedure for Migrating Nodes in a Custom Node Pool

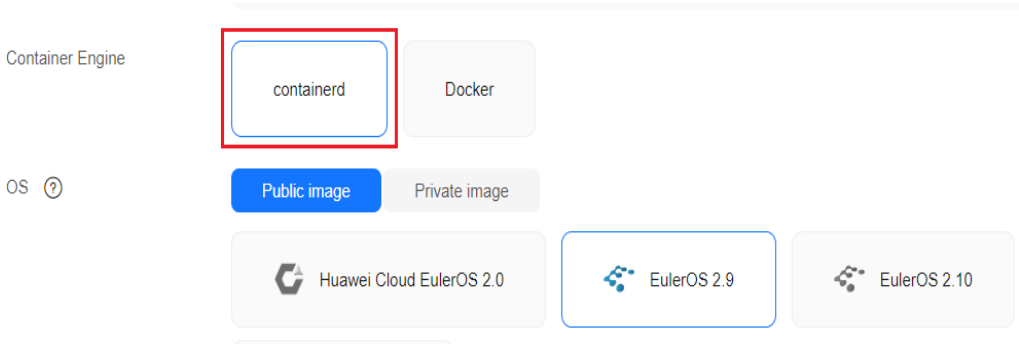
You can **copy a node pool**, set the container engine of the new node pool to containerd, and keep other configurations the same as those of the original Docker node pool.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the Docker node pool to be copied and choose **More** > **Copy** in the **Operation** column.



Step 3 In the **Configurations** area, set **Container Engine** to **containerd** and modify other parameter settings as needed to create the node pool.



Step 4 Scale the number of created containerd node pools to the number of original Docker node pools and delete nodes from the Docker node pools one by one.

Rolling migration is preferred. That is, add some containerd nodes and then delete some Docker nodes until the number of nodes in the new containerd node pool is the same as that in the original Docker node pool.

NOTE

If you have configured node affinity for the workloads deployed on the original Docker nodes or node pool, configure affinity policies for the workloads to run on the new containerd nodes or node pool.

Step 5 After the migration, delete the original Docker node pool.

----End

3.9.6 Optimizing Node System Parameters

3.9.6.1 Optimizable Node System Parameters

CCE provides default node system parameters, which may cause performance bottlenecks in some scenarios. Therefore, you can customize and optimize some node system parameters. **Optimizable Node System Parameters** describes the node system parameters.

NOTICE

- The modification has certain risks. Be familiar with Linux commands and Linux OS.
- The parameters listed in **Table 3-73** have been tested and verified. **Do not modify other parameters**. Otherwise, node faults may occur.
- The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.
- After the node is restarted, run the **sysctl -p** command to update the parameter value.

Table 3-73 System parameters that can be optimized

Parameter	Parameter Location	Description	Reference
kernel.pid_max	/etc/sysctl.conf	Maximum number of process IDs on a node Obtaining the parameter: `sysctl kernel.pid_max`	**Changing Process ID Limits (kernel.pid_max)**
RuntimeMaxUse	/etc/systemd/journald.conf	Upper limit of the memory occupied by the node log cache. If this parameter is not set, a large amount of memory will be occupied after the system runs for a long time. Obtaining the parameter: `cat /etc/systemd/journald.conf | grep RuntimeMaxUse`	**Changing the RuntimeMaxUse of the Memory Used by the Log Cache on a Node**
Openfiles	/etc/security/limits.conf	Maximum number of file handles for a single process on a node, which can be adjusted as required. Obtaining the parameter: `ulimit -n`	**Changing the Maximum Number of File Handles for a Single Process on a Node**

Parameter	Parameter Location	Description	Reference

(inside the Openfiles container)
LimitNOFILE
LimitNPROC
	• CentOS/ EulerOS:
	– Docker nodes: /usr/lib/systemd/system/docker.service
	– Docker nodes: /usr/lib/systemd/system/containerd.service
	• Ubuntu:
	– Docker nodes: /lib/systemd/system/docker.service
	– Docker nodes: /lib/systemd/system/containerd.service
		Maximum number of file handles for a single process in a container, which can be adjusted as required.
		Obtaining the parameter:
		Docker nodes:
		`cat /proc/`pidof dockerd`/limits | grep files`
		containerd nodes:
		`cat /proc/`pidof containerd`/limits | grep files`
			Changing the Maximum Number of File Handles for a Single Container Process

file-max	/etc/sysctl.conf	Maximum number of file handles in the system, which can be adjusted as required. Obtaining the parameter: `sysctl fs.file-max`	**Changing the Maximum Number of System-Level File Handles on a Node**

Parameter	Parameter Location	Description	Reference

nf_conntrack_buckets
nf_conntrack_max
	/etc/sysctl.conf	Capacity of the connection tracing table, which can be adjusted as required.
		Bucket usage = [nf_conntrack_count]/[nf_conntrack_buckets]
		Adjust the buckets value to ensure that the bucket usage is lower than 0.7.
		Obtaining the parameter:
		`sysctl net.netfilter.nf_conntrack_count`
		`sysctl net.netfilter.nf_conntrack_buckets`
		`sysctl net.netfilter.nf_conntrack_max`
			Modifying Node Kernel Parameters

net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Expiration time of the entry of the connection in the close state in the connection tracking table. Shortening the expiration time can speed up the recycling. Obtaining the parameter: `sysctl net.netfilter.nf_conntrack_tcp_timeout_close`	
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	The parameter value is **0** or **1**. • **0**: The function is disabled. All packets that are not in the TCP window are marked as invalid. • **1**: The function is enabled. Only packets that are not in the TCP window are marked as invalid. In containers, enabling this parameter can prevent the bandwidth of TCP connections that have been translated using NAT from being limited. Obtaining the parameter: `sysctl net.netfilter.nf_conntrack_tcp_be_liberal`	

Parameter	Parameter Location	Description	Reference
tcp_keepalive_time	/etc/sysctl.conf	Interval at which a TCP keepalive message is sent. If this parameter is set to a large value, TCP connections may be suspended with **Close_wait** state for a long time, which exhausts system resources. Obtaining the parameter: `sysctl net.ipv4.tcp_keepalive_time`	
tcp_max_syn_backlog	/etc/sysctl.conf	Maximum number of TCP half-connections, that is, the maximum number of connections in the **SYN_RECV** queue. Obtaining the parameter: `sysctl net.ipv4.tcp_max_syn_backlog`	
tcp_max_tw_buckets	/etc/sysctl.conf	Specifies the maximum number of sockets in the **time-wait** state that can exist at any time. If the parameter value is too large, node resources may be exhausted. Obtaining the parameter: `sysctl net.ipv4.tcp_max_tw_buckets`	
net.core.somaxconn	/etc/sysctl.conf	Maximum number of TCP connections. This parameter controls the number of TCP connections in a queue. If this parameter is set to a small value, the number of TCP connections is prone to insufficiency. If this parameter is set to a large value, system resources may be wasted because each client waiting for connection in the connection queue occupies certain memory resources. Obtaining the parameter: `sysctl net.core.somaxconn`	

Parameter	Parameter Location	Description	Reference
max_user_instances	/etc/sysctl.conf	Maximum number of inotify instances allowed for each user. If the parameter value is too small, the number of inotify instances may be insufficient in containers. Obtaining the parameter: `sysctl fs.inotify.max_user_instances`	
max_user_watches	/etc/sysctl.conf	Maximum number of directories of all monitoring instances. If the parameter value is too small, the number of directories may be insufficient in containers. Obtaining the parameter: `sysctl fs.inotify.max_user_watches`	
netdev_max_backlog	/etc/sysctl.conf	Size of the packet receiving queue of the network protocol stack. If the parameter value is too small, the queue size may be insufficient. Obtaining the parameter: `sysctl net.core.netdev_max_backlog`	

net.core.wmem_max
net.core.rmem_max
	/etc/sysctl.conf	Memory size (bytes) of the sending and receiving buffer. If this parameter is set to a small value, the memory size may be insufficient in large file scenarios.
		Obtaining the parameter:
		`sysctl net.core.wmem_max`
		`sysctl net.core.rmem_max`	

Parameter	Parameter Location	Description	Reference

net.ipv4.neigh.default.gc_thresh1
net.ipv4.neigh.default.gc_thresh2
net.ipv4.neigh.default.gc_thresh3
	/etc/sysctl.conf	Optimization of the garbage collection of ARP entries.
		• **gc_thresh1**: indicates the minimum number of entries that can be reserved. If the number of entries is less than the **gc_thresh1** value, the garbage collector (GC) will not reclaim these entries. **Do not modify the default parameter setting.**
		• **gc_thresh2**: When the number of entries exceeds the value of this parameter, the GC will clear the entries that have been stored for more than 5 seconds. **Do not modify the default parameter setting.**
		• **gc_thresh3**: indicates the maximum number of non-permanent entries. If the system provides a large number of APIs or is directly connected to a large number of devices, increase the value of this parameter.
		Obtaining the parameter:
		`sysctl net.ipv4.neigh.default.gc_thresh1`
		`sysctl net.ipv4.neigh.default.gc_thresh2`
		`sysctl net.ipv4.neigh.default.gc_thresh3`	

vm.max_map_count	/etc/sysctl.conf	If this parameter is set to a small value, a message is displayed indicating that the space is insufficient during ELK installation. Obtaining the parameter: `sysctl vm.max_map_count`	

3.9.6.2 Changing the RuntimeMaxUse of the Memory Used by the Log Cache on a Node

Journald is a log system in Linux. It writes log information into binary files and uses **/run/log/journal** as the log cache directory by default. The Journald configuration file is stored in the **/etc/systemd/journald.conf** directory on the node. The **RuntimeMaxUse** parameter indicates the maximum memory usage of the log cache. If **RuntimeMaxUse** is not set, a large amount of memory will be occupied after the system runs for a long time.

NOTICE

The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.

Changing RuntimeMaxUse

Step 1 Log in to the node and view the **/etc/systemd/journald.conf** file.

```
cat /etc/systemd/journald.conf
```

Step 2 Modify **RuntimeMaxUse**. The recommended value is **100M**.

- If **RuntimeMaxUse** has been set in the **journald.conf** file, run the following command to change the value:
  ```
  sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/journald.conf && systemctl restart systemd-journald
  ```
- If **RuntimeMaxUse** is not set in the **journald.conf** file, run the following command to add it:
  ```
  echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart systemd-journald
  ```

Step 3 If the returned value is the same as the modified value, the modification is correct.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----End

Automatically Configuring RuntimeMaxUse When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure the **RuntimeMaxUse** size.

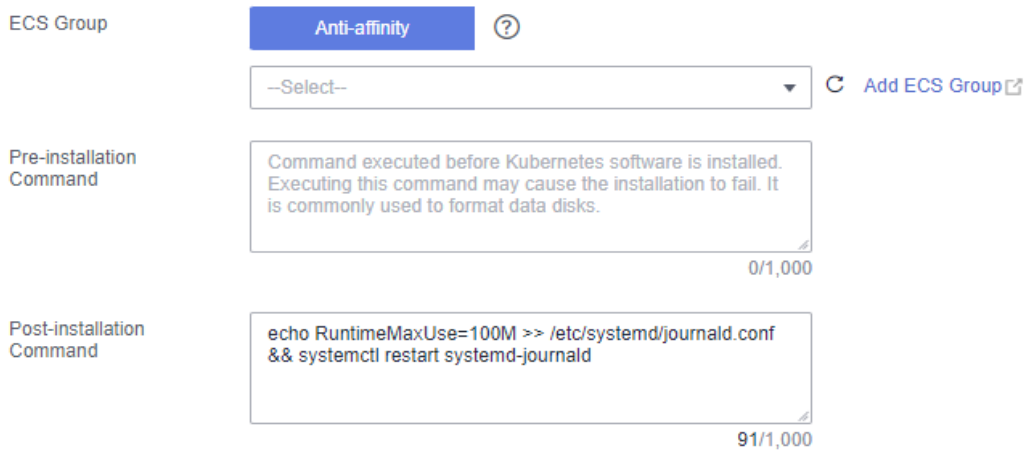
Step 1 Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

Step 2 Manually test the script commands on nodes **in the same cluster and running the same OS**. For details about how to manually run the script, see **Changing RuntimeMaxUse**.

Step 3 When creating a node or node pool, choose **Advanced Settings** > **Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)**

- Log in to the node and view the **/etc/systemd/journald.conf** file. If **RuntimeMaxUse** has been set, run the following command to change the value:
  ```
  sed -i "s/RuntimeMaxUse=[0-9]*M/RuntimeMaxUse=100M/g" /etc/systemd/journald.conf && systemctl restart systemd-journald
  ```
- Log in to the node and view the **/etc/systemd/journald.conf** file. If **RuntimeMaxUse** is not set, run the following command to add it:
  ```
  echo RuntimeMaxUse=100M >> /etc/systemd/journald.conf && systemctl restart systemd-journald
  ```

The command in the following figure is used only as an example. Change it as required.



Step 4 After the node is created, log in to the node to check whether the parameters are successfully modified.

```
cat /etc/systemd/journald.conf | grep RuntimeMaxUse
```

----End

3.9.6.3 Changing the Maximum Number of File Handles

The maximum number of file handles is the maximum number of files that can be opened. In Linux, there are two file handle restrictions. The one is system-level restriction, where the maximum number of files that can be opened by all user processes at the same time. The other is user-level restriction, where the maximum number of files that can be opened by a single user process. Containers have the third file handle restriction, that is, the maximum number of file handles of a single process in the container.

NOTICE

The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.

Changing the Maximum Number of System-Level File Handles on a Node

Step 1 Log in to the node and check the **/etc/sysctl.conf** file.

```
cat /etc/sysctl.conf
```


Step 2 Modify the **fs.file-max** parameter. **fs.file-max=1048576** indicates the kernel parameter name and recommended value.

- If the value of fs.file-max has been set in the **sysctl.conf** file, run the following command to change the value:
  ```
  sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
  ```
- If **fs.file-max** is not set in the **sysctl.conf** file, run the following command to add it:
  ```
  echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
  ```

Step 3 Run the following commands to check whether the change is successful (whether the returned value is the same as that you configure).

```
# sysctl fs.file-max
fs.file-max = 1048576
```

----End

Changing the Maximum Number of File Handles for a Single Process on a Node

Step 1 Log in to the node and view the **/etc/security/limits.conf** file.

```
cat /etc/security/limits.conf
```

The maximum number of file handles for a single process of a node is specified by the following parameters:

```
...
root soft nofile 65535
root hard nofile 65535
* soft nofile 65535
* hard nofile 65535
```

Step 2 Run the **sed** command to change the maximum number of file handles. In the command, **65535** is the recommended maximum number of file handles.
The **/etc/security/limits.conf** file on the EulerOS 2.3 node does not contain the default configuration related to nofile. Therefore, you cannot run the **sed** command to modify the configuration.

```
sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
```

Step 3 **Log in to the node again** and run the following command to check whether the modification is successful. If the returned value is the same as the modified value, the modification is successful.

```
# ulimit -n
65535
```

----End

Changing the Maximum Number of File Handles for a Single Container Process

Step 1 Log in to the node and view the **/usr/lib/systemd/system/docker.service** file.

- CentOS/EulerOS:
  - Docker nodes:
    ```
    cat /usr/lib/systemd/system/docker.service
    ```
  - containerd nodes:
    ```
    cat /usr/lib/systemd/system/containerd.service
    ```


- Ubuntu:
  - Docker nodes:
    ```
    cat /lib/systemd/system/docker.service
    ```
  - containerd nodes:
    ```
    cat /lib/systemd/system/containerd.service
    ```

NOTE

If **LimitNOFILE** or **LimitNPROC** is set to **infinity**, the maximum number of file handles supported by a single process of a container is **1,048,576**.

The maximum number of file handles for a single process of a container is specified by the following parameters:

```
...
LimitNOFILE=1048576
LimitNPROC=1048576
...
```

Step 2 Run the following commands to modify the two parameters. In the command, **1048576** is the recommended value of the maximum number of file handles.

NOTICE

Changing the maximum number of file handles of a container will restart the docker/containerd process.

- CentOS/EulerOS:
  - Docker nodes:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```
  - containerd nodes:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service && systemctl daemon-reload && systemctl restart containerd
    ```
- Ubuntu:
  - Docker nodes:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```
  - containerd nodes:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/containerd.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/containerd.service && systemctl daemon-reload && systemctl restart containerd
    ```

Step 3 Check the maximum number of file handles of a single process in the container. If the returned value is the same as the modified value, the modification is successful.

- Docker nodes:
  ```
  # cat /proc/`pidof dockerd`/limits | grep files
  Max open files            1048576              1048576              files
  ```
- containerd nodes:
  ```
  # cat /proc/`pidof containerd`/limits | grep files
  Max open files            1048576              1048576              files
  ```

----End

Automatically Configuring the Maximum Number of File Handles When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure the maximum number of file handles.

Step 1 Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

Step 2 Manually test the script commands on nodes **in the same cluster and running the same OS**.

- **Changing the Maximum Number of System-Level File Handles on a Node**
- **Changing the Maximum Number of File Handles for a Single Process on a Node**
- **Changing the Maximum Number of File Handles for a Single Container Process**

Step 3 When creating a node or node pool, choose **Advanced Settings** > **Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)**

- Change the maximum number of system-level file handles on a node.
  - Log in to the node and check the **/etc/sysctl.conf** file. If the value of **fs.file-max** has been set in the file, run the following command to change it:
    ```
    sed -i "s/fs.file-max=[0-9]*$/fs.file-max=1048576/g" /etc/sysctl.conf && sysctl -p
    ```
  - Log in to the node and check the **/etc/sysctl.conf** file. If the value of **fs.file-max** is not set in the file, run the following command to add it:
    ```
    echo fs.file-max=1048576 >> /etc/sysctl.conf && sysctl -p
    ```

  In the preceding command, **fs.file-max=1048576** indicates the kernel parameter name and recommended value.
- Run the following command to change the maximum number of file handles for a single process on a node:
  ```
  sed -i "s/nofile.[0-9]*$/nofile 65535/g" /etc/security/limits.conf
  ```

  In the preceding command, **65535** is the recommended maximum number of file handles.
- Change the maximum number of file handles for a single process of a container.
  - CentOS/EulerOS:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /usr/lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /usr/lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```
  - Ubuntu:
    ```
    sed -i "s/LimitNOFILE=[0-9a-Z]*$/LimitNOFILE=1048576/g" /lib/systemd/system/docker.service;sed -i "s/LimitNPROC=[0-9a-Z]*$/LimitNPROC=1048576/g" /lib/systemd/system/docker.service && systemctl daemon-reload && systemctl restart docker
    ```

  In the preceding command, **1048576** is the recommended maximum number of file handles.

The command in the following figure is used only as an example. Change it as required.

Step 4 After the node is created, log in to the node to check whether the parameters are successfully modified.

----End

3.9.6.4 Modifying Node Kernel Parameters

The default Linux kernel parameters may not satisfy all users. You can modify the **/etc/sysctl.conf** configuration file on the node to modify the kernel parameters.

NOTICE

- The commands for modifying node system parameters are valid only when public images are used. The commands provided in this document are for reference only when private images are used.
- After the node is restarted, run the **sysctl -p** command to update the parameter value.

Table 3-74 Kernel parameters of a node

Parameter	Parameter Location	Description	Recommended Value
file-max	/etc/sysctl.conf	Maximum number of file handles in the system, which can be adjusted as required. Obtaining the parameter: `sysctl fs.file-max`	fs.file-max=1048576

Parameter	Parameter Location	Description	Recommended Value

nf_conntrack_buckets
nf_conntrack_max
	/etc/sysctl.conf	Capacity of the connection tracing table, which can be adjusted as required.
		Bucket usage = [nf_conntrack_count]/ [nf_conntrack_buckets]
		If the CPU usage is greater than 0.7 for a long time, increase the value of buckets to lower the CPU usage to less than 0.7.
		Obtaining the parameter:
		`sysctl net.netfilter.nf_conntrack_count`
		`sysctl net.netfilter.nf_conntrack_buckets`
		`sysctl net.netfilter.nf_conntrack_max`
		NOTE
		Note: **net.netfilter.nf_conntrack_buckets** on EulerOS 2.3, EulerOS 2.5, and CentOS 7.6 cannot be modified by editing **/etc/sysctl.conf**, you can modify buckets by modifying **/sys/module/nf_conntrack/parameters/hashsize**.
			The default value is set based on the node memory size. To change the value, refer to the following formula:
			• net.netfilter.nf_conntrack_buckets = [nf_conntrack_count] / 0.7
			• net.netfilter.nf_conntrack_max = 4* [nf_conntrack_buckets]

net.netfilter.nf_conntrack_tcp_timeout_close	/etc/sysctl.conf	Expiration time of the entry of the connection in the close state in the connection tracking table. Shortening the expiration time can speed up the recycling. Obtaining the parameter: `sysctl net.netfilter.nf_conntrack_tcp_timeout_close`	net.netfilter.nf_conntrack_tcp_timeout_close=3

Parameter	Parameter Location	Description	Recommended Value
net.netfilter.nf_conntrack_tcp_be_liberal	/etc/sysctl.conf	The parameter value is **0** or **1**. • **0**: The function is disabled. All packets that are not in the TCP window are marked as invalid. • **1**: The function is enabled. Only packets that are not in the TCP window are marked as invalid. In containers, enabling this parameter can prevent the bandwidth of TCP connections that have been translated using NAT from being limited. Obtaining the parameter: `sysctl net.netfilter.nf_conntrack_tcp_be_liberal`	net.netfilter.nf_conntrack_tcp_be_liberal=1
tcp_keepalive_time	/etc/sysctl.conf	Interval for sending keepalive detection messages through TCP. If this parameter is set to a large value, TCP connections may be suspended in the **Close_wait** phase for a long time, which exhausts system resources. Obtaining the parameter: `sysctl net.ipv4.tcp_keepalive_time`	net.ipv4.tcp_keepalive_time=600
tcp_max_syn_backlog	/etc/sysctl.conf	Maximum number of TCP half-connections, that is, the maximum number of connections in the **SYN_RECV** queue. Obtaining the parameter: `sysctl net.ipv4.tcp_max_syn_backlog`	net.ipv4.tcp_max_syn_backlog=8096

Parameter	Parameter Location	Description	Recommended Value
tcp_max_tw_buckets	/etc/sysctl.conf	Specifies the maximum number of sockets in the time-wait state that can exist at any time. If the parameter value is too large, node resources may be exhausted. Obtaining the parameter: `sysctl net.ipv4.tcp_max_tw_buckets`	net.ipv4.tcp_max_tw_buckets=5000
net.core.somaxconn	/etc/sysctl.conf	Maximum number of TCP connections and maximum size of the ESTABLISHED queue. If the parameter value is too small, the value may be insufficient. Obtaining the parameter: `sysctl net.core.somaxconn`	net.core.somaxconn=32768
max_user_instances	/etc/sysctl.conf	Maximum number of inotify instances allowed for each user. If the parameter value is too small, the number of inotify instances may be insufficient in containers. Obtaining the parameter: `sysctl fs.inotify.max_user_instances`	fs.inotify.max_user_instances=8192
max_user_watches	/etc/sysctl.conf	Maximum number of directories of all monitoring instances. If the parameter value is too small, the number of directories may be insufficient in containers. Obtaining the parameter: `sysctl fs.inotify.max_user_watches`	fs.inotify.max_user_watches=524288
netdev_max_backlog	/etc/sysctl.conf	Size of the packet receiving queue of the network protocol stack. If the parameter value is too small, the queue size may be insufficient. Obtaining the parameter: `sysctl net.core.netdev_max_backlog`	net.core.netdev_max_backlog=16384

Parameter	Parameter Location	Description	Recommended Value

net.core.wmem_max
net.core.rmem_max
	/etc/sysctl.conf	Memory size (bytes) of the sending and receiving buffer. If this parameter is set to a small value, the memory size may be insufficient in large file scenarios.
		Obtaining the parameter:
		`sysctl net.core.wmem_max`
		`sysctl net.core.rmem_max`
			net.core.wmem_max=16777216
			net.core.rmem_max=16777216

net.ipv4.neigh.default.gc_thresh1
net.ipv4.neigh.default.gc_thresh2
net.ipv4.neigh.default.gc_thresh3
	/etc/sysctl.conf	Garbage collection optimization of ARP entries.
		• **gc_thresh1**: indicates the minimum number of entries that can be reserved. If the number of entries is less than the **gc_thresh1** value, the garbage collector (GC) will not reclaim these entries. **Do not modify the default parameter setting.**
		• **gc_thresh2**: When the number of entries exceeds the value of this parameter, the GC will clear the entries that have been stored for more than 5 seconds. **Do not modify the default parameter setting.**
		• **gc_thresh3**: indicates the maximum number of non-permanent entries. If the system provides a large number of APIs or is directly connected to a large number of devices, increase the value of this parameter.
		Obtaining the parameter:
		`sysctl net.ipv4.neigh.default.gc_thresh1`
		`sysctl net.ipv4.neigh.default.gc_thresh2`
		`sysctl net.ipv4.neigh.default.gc_thresh3`
			net.ipv4.neigh.default.gc_thresh1=0
			net.ipv4.neigh.default.gc_thresh2=4096
			net.ipv4.neigh.default.gc_thresh3=163790

Parameter	Parameter Location	Description	Recommended Value
vm.max_map_count	/etc/sysctl.conf	If this parameter is set to a small value, a message is displayed indicating that the space is insufficient during ELK installation. Obtaining the parameter: `sysctl vm.max_map_count`	vm.max_map_count=262144

Modifying Node Kernel Parameters

Table 3-74 lists the kernel parameters of nodes. The following describes how to change the value of **tcp_keepalive_time**, which indicates the interval for sending keepalive detection messages over TCP.

Step 1 Log in to the node and check the **/etc/sysctl.conf** file.

```
cat /etc/sysctl.conf
```

Step 2 Modify the **net.ipv4.tcp_keepalive_time** parameter. **net.ipv4.tcp_keepalive_time=600** indicates the kernel parameter name and recommended value. For details about the recommended value, see Table 3-74.

To modify other kernel parameters, replace the parameter names and values in the commands by referring to Table 3-74.

- If **net.ipv4.tcp_keepalive_time** has been set in the **sysctl.conf** file, run the following command to change the value:

  ```
  sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
  ```

- If **net.ipv4.tcp_keepalive_time** is not set in the **sysctl.conf** file, run the following command to add it:

  ```
  echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
  ```

Step 3 Run the command in Table 3-74 to check whether the modification is successful. If the returned value is the same as the modified value, the modification is successful.

```
# sysctl net.ipv4.tcp_keepalive_time
net.ipv4.tcp_keepalive_time = 600
```

----End

Automatically Configuring Kernel Parameters When Creating a Node or Node Pool

You can set the script to be executed after a node or node pool is created. When creating a node or node pool, you can use the script to configure kernel parameters.

The **tcp_keepalive_time** parameter is used as an example to describe how to change the interval for sending keepalive detection messages over TCP. The value is the recommended value in Table 3-74.

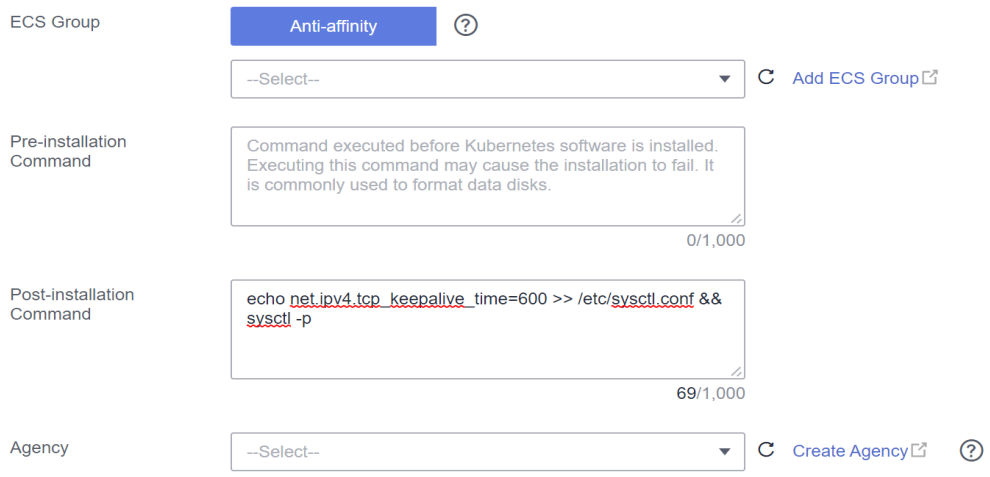
Step 1 Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

Step 2 Manually test the script commands on nodes **in the same cluster and running the same OS**. For details about how to manually run the script, see **Modifying Node Kernel Parameters**.

Step 3 When creating a node or node pool, choose **Advanced Settings** > **Post-installation Command** to add commands. **(The following commands must be configured after the verification is successful.)** To modify other kernel parameters, replace the parameter names and values in the commands by referring to **Table 3-74**.

- Log in to the node and check the **/etc/sysctl.conf** file. If **net.ipv4.tcp_keepalive_time** has been set in the file, run the following command to change it:
  ```
  sed -i "s/net.ipv4.tcp_keepalive_time=[0-9]*$/net.ipv4.tcp_keepalive_time=600/g" /etc/sysctl.conf && sysctl -p
  ```
- Log in to the node and check the **/etc/sysctl.conf** file. If **net.ipv4.tcp_keepalive_time** is not set in the file, run the following command to add it:
  ```
  echo net.ipv4.tcp_keepalive_time=600 >> /etc/sysctl.conf && sysctl -p
  ```

The command in the following figure is used only as an example. Change it as required.



Step 4 After the node is created, log in to the node and run the command in **Table 3-74** to check whether the modification is successful.

----End

3.9.6.5 Changing Process ID Limits (kernel.pid_max)

Context

Process IDs (PIDs) are a fundamental resource on nodes. It is trivial to hit the task limit without hitting any other resource limits, which can then cause instability to a host machine.

You can adjust the PID limit (kernel.pid_max) according to service requirements.

kernel.pid_max Defaults

Starting from January 2022, CCE changes the default value of kernel.pid_max to **4194304** for EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04 nodes in clusters of v1.17 or later. Specific conditions:

- Cluster version: v1.17.17 or later
- Node creation: after January 30, 2022

If the preceding two conditions are not met, **kernel.pid_max** on EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04 nodes defaults to **32768**.

Table 3-75 kernel.pid_max defaults

OS	Clusters of 1.17.9 and Earlier	Clusters of 1.17.17 and Later	
		Nodes Created on or Before January 30, 2022	Nodes Created After January 30, 2022
EulerOS 2.5	32768	32768	4194304
CentOS 7.6	32768	32768	4194304
Ubuntu 18.04	N/A	32768	4194304
EulerOS 2.3	57344	57344	57344
EulerOS 2.9	N/A	4194304	4194304

Change Suggestion

- EulerOS 2.3: Change the default to **4194304** for all nodes. For details, see **Changing kernel.pid_max of a Node**. Use a pre-installation script to do so for new nodes and node pools. For details, see **Configuring kernel.pid_max When Creating a Node Pool** or **Configuring kernel.pid_max When Creating a Node**.
- EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04:
  - Change the value of **kernel.pid_max** to **4194304** for nodes created on January 30, 2022 or earlier in clusters of v1.17.17 or later. For details, see **Changing kernel.pid_max of a Node**.
  - For clusters of 1.17.9 and earlier:
    - Change the value of **kernel.pid_max** to **4194304** for existing nodes. For details, see **Changing kernel.pid_max of a Node**.
    - Use a pre-installation script to do so for new nodes and node pools. For details, see **Configuring kernel.pid_max When Creating a Node Pool** or **Configuring kernel.pid_max When Creating a Node**.

Viewing kernel.pid_max

Log in to the node and run the following command to obtain the value of **kernel.pid_max**:

sysctl kernel.pid_max

```
# sysctl kernel.pid_max
kernel.pid_max = 32768
```

Change **kernel.pid_max**, if necessary, as instructed in **Changing kernel.pid_max of a Node**.

Checking Node PIDs

Log in to the node and run the following command to check how many PIDs are in use:

ps -eflL | wc -l

```
# ps -eflL | wc -l
691
```

Changing kernel.pid_max of a Node

Log in to the node and run the following command. **4194304** indicates the value of **kernel.pid_max** and is used as an example here.

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p

echo *4194304* > /sys/fs/cgroup/pids/kubepods/pids.max

Run the following commands to check whether the returned value is the same as that you configured:

```
# sysctl kernel.pid_max
kernel.pid_max = 4194304
# cat /sys/fs/cgroup/pids/kubepods/pids.max
4194304
```

Configuring kernel.pid_max When Creating a Node Pool

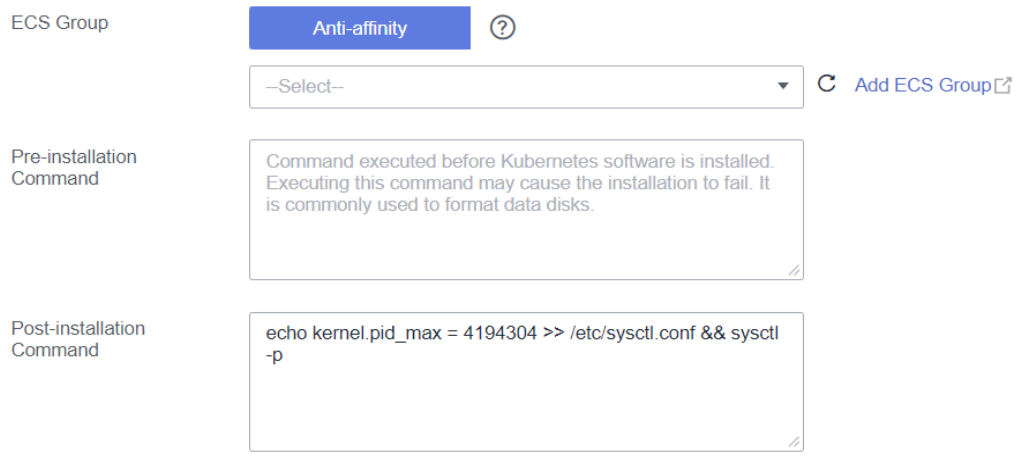
EulerOS 2.3: Configuration required.

EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04: **Configuration required** for clusters of v1.17.9 and earlier. **Configuration NOT required** for clusters of v1.17.17 and later because the value has been changed.

You can configure **kernel.pid_max** in the pre-installation script to create a node from a node pool.

When creating a node pool, choose **Advanced Settings** > **Post-installation Command** and add the following command:

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p



Configuring kernel.pid_max When Creating a Node

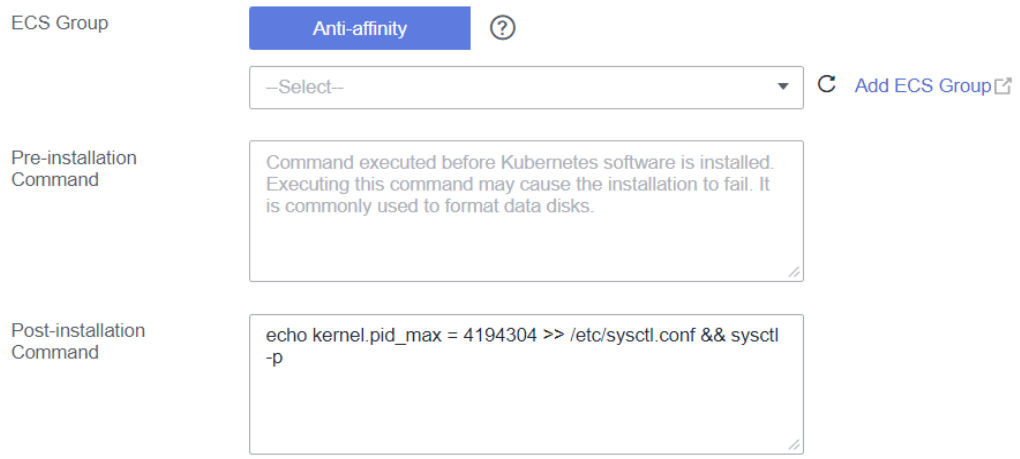
EulerOS 2.3: Configuration required.

EulerOS 2.5, CentOS 7.6, and Ubuntu 18.04: **Configuration required** for clusters of v1.17.9 and earlier. **Configuration NOT required** for clusters of v1.17.17 and later because the value has been changed.

You can configure **kernel.pid_max** by using the pre-installation script when creating a node.

Choose **Advanced Settings** > **Post-installation Command** and add the following command:

echo kernel.pid_max = *4194304* >> /etc/sysctl.conf && sysctl -p



3.9.7 Configuring Node Fault Detection Policies

The node fault detection function depends on the **NPD** add-on. The add-on instances run on nodes and monitor nodes. This section describes how to enable node fault detection.

Prerequisites

The **CCE Node Problem Detector** add-on has been installed in the cluster.

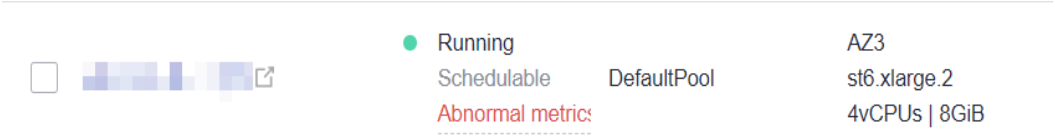
Enabling Node Fault Detection

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

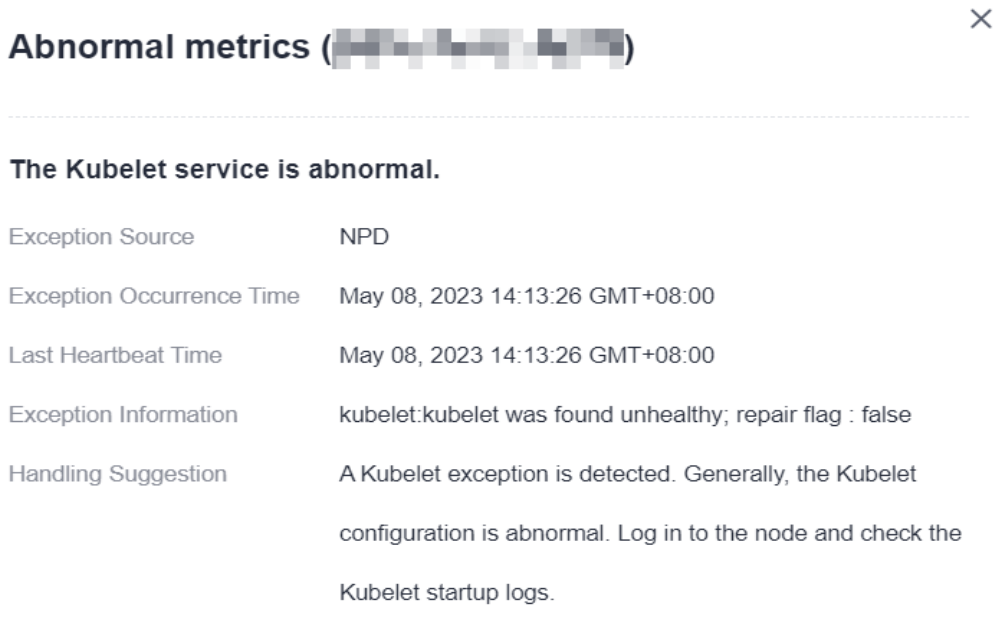
Step 2 In the navigation pane, choose **Nodes** and then click the **Nodes** tab. Check whether the NPD add-on has been installed in the cluster or whether the add-on has been upgraded to the latest version. After the NPD add-on has been installed, you can use the fault detection function.

Step 3 If the NPD add-on is running properly, click **Node Fault Detection Policy** to view the current fault detection items. For details about the NPD check item list, see **NPD Check Items**.

Step 4 If the check result of the current node is abnormal, a message is displayed in the node list, indicating that the metric is abnormal.



Step 5 You can click **Abnormal metrics** and rectify the fault as prompted.



----End

Customized Check Items

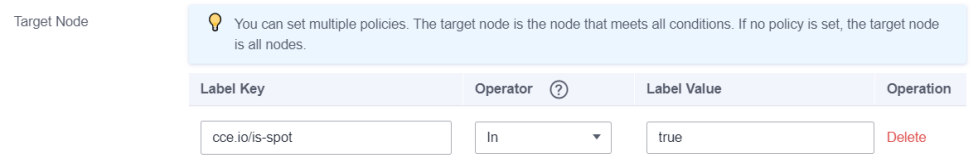
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes** and then click the **Nodes** tab. Then, click **Fault Detection Policy**.

Step 3 On the displayed page, view the current check items. Click **Edit** in the **Operation** column and edit checks.

Currently, the following configurations are supported:

- **Enable/Disable**: Enable or disable a check item.
- **Target Node**: By default, check items run on all nodes. You can change the fault threshold based on special scenarios. For example, the spot price ECS interruption reclamation check runs only on the spot price ECS node.



- **Trigger Threshold**: The default thresholds match common fault scenarios. You can customize and modify the fault thresholds as required. For example, change the threshold for triggering connection tracking table exhaustion from 90% to 80%.

- **Check Period**: The default check period is 30 seconds. You can modify this parameter as required.

- **Troubleshooting Strategy**: After a fault occurs, you can select the strategies listed in the following table.

Table 3-76 Troubleshooting strategies

Troubleshooting Strategy	Effect
Prompting Exception	Kubernetes events are reported.
Disabling scheduling	Kubernetes events are reported and the **NoSchedule** taint is added to the node.
Evict Node Load	Kubernetes events are reported and the **NoExecute** taint is added to the node. This operation will evict workloads on the node and interrupt services. Exercise caution when performing this operation.

----End

NPD Check Items

NOTE

Check items are supported only in 1.16.0 and later versions.

Check items cover events and statuses.

- Event-related

  For event-related check items, when a problem occurs, NPD reports an event to the API server. The event type can be **Normal** (normal event) or **Warning** (abnormal event).

  Table 3-77 Event-related check items

Check Item	Function	Description
OOMKilling	Listen to the kernel logs and check whether OOM events occur and are reported. Typical scenario: When the memory usage of a process in a container exceeds the limit, OOM is triggered and the process is terminated.	Warning event Listening object: **/dev/kmsg** Matching rule: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d+kB.*"
TaskHung	Listen to the kernel logs and check whether taskHung events occur and are reported. Typical scenario: Disk I/O suspension causes process suspension.	Warning event Listening object: **/dev/kmsg** Matching rule: "task \ \S+:\\w+ blocked for more than \\w+ seconds\\."
Readonly Filesystem	Check whether the **Remount root filesystem read-only** error occurs in the system kernel by listening to the kernel logs. Typical scenario: A user detaches a data disk from a node by mistake on the ECS, and applications continuously write data to the mount point of the data disk. As a result, an I/O error occurs in the kernel and the disk is remounted as a read-only disk. **NOTE** If the rootfs of node pods is of the device mapper type, an error will occur in the thin pool if a data disk is detached. This will affect NPD and NPD will not be able to detect node faults.	Warning event Listening object: **/dev/kmsg** Matching rule: **Remounting filesystem read-only**

- Status-related

For status-related check items, when a problem occurs, NPD reports an event to the API server and changes the node status synchronously. This function can be used together with **Node-problem-controller fault isolation** to isolate nodes.

If the check period is not specified in the following check items, the default period is 30 seconds.

Table 3-78 Checking system components

Check Item	Function	Description
Container network component error CNIProblem	Check the status of the CNI components (container network components).	None
Container runtime component error CRIProblem	Check the status of Docker and containerd of the CRI components (container runtime components).	Check object: Docker or containerd
Frequent restarts of Kubelet FrequentKubeletRestart	Periodically backtrack system logs to check whether the key component Kubelet restarts frequently.	• Default threshold: 10 restarts within 10 minutes If Kubelet restarts for 10 times within 10 minutes, it indicates that the system restarts frequently and a fault alarm is generated. • Listening object: logs in the **/run/log/journal** directory **NOTE** The Ubuntu and HCE 2.0 OSs do not support the preceding check items due to incompatible log formats.
Frequent restarts of Docker FrequentDockerRestart	Periodically backtrack system logs to check whether the container runtime Docker restarts frequently.	
Frequent restarts of containerd FrequentContainerdRestart	Periodically backtrack system logs to check whether the container runtime containerd restarts frequently.	
kubelet error KubeletProblem	Check the status of the key component Kubelet.	None
kube-proxy error KubeProxyProblem	Check the status of the key component kube-proxy.	None

Table 3-79 Checking system metrics

Check Item	Function	Description
Conntrack table full ConntrackFullProblem	Check whether the conntrack table is full.	• Default threshold: 90% • Usage: **nf_conntrack_count** • Maximum value: **nf_conntrack_max**
Insufficient disk resources DiskProblem	Check the usage of the system disk and CCE data disks (including the CRI logical disk and kubelet logical disk) on the node.	• Default threshold: 90% • Source: `df -h` Currently, additional data disks are not supported.
Insufficient file handles FDProblem	Check if the FD file handles are used up.	• Default threshold: 90% • Usage: the first value in **/proc/sys/fs/file-nr** • Maximum value: the third value in **/proc/sys/fs/file-nr**
Insufficient node memory MemoryProblem	Check whether memory is used up.	• Default threshold: 80% • Usage: **MemTotal-MemAvailable** in **/proc/meminfo** • Maximum value: **MemTotal** in **/proc/meminfo**
Insufficient process resources PIDProblem	Check whether PID process resources are exhausted.	• Default threshold: 90% • Usage: **nr_threads in /proc/loadavg** • Maximum value: smaller value between **/proc/sys/kernel/pid_max** and **/proc/sys/kernel/threads-max**.

Table 3-80 Checking the storage

Check Item	Function	Description
Disk read-only DiskReadonly	Periodically perform write tests on the system disk and CCE data disks (including the CRI logical disk and Kubelet logical disk) of the node to check the availability of key disks.	Detection paths: • /mnt/paas/kubernetes/kubelet/ • /var/lib/docker/ • /var/lib/containerd/ • /var/paas/sys/log/cceaddon-npd/ The temporary file **npd-disk-write-ping** is generated in the detection path. Currently, additional data disks are not supported.

Check Item	Function	Description
emptyDir storage pool error EmptyDirVolumeGroupStatusError	Check whether the ephemeral volume group on the node is normal. Impact: Pods that depend on the storage pool cannot write data to the temporary volume. The temporary volume is remounted as a read-only file system by the kernel due to an I/O error. Typical scenario: When creating a node, a user configures two data disks as an ephemeral volume storage pool. Some data disks are deleted by mistake. As a result, the storage pool becomes abnormal.	• Detection period: 30s • Source: `vgs -o vg_name, vg_attr` • Principle: Check whether the VG (storage pool) is in the P state. If yes, some PVs (data disks) are lost. • Joint scheduling: The scheduler can automatically identify a PV storage pool error and prevent pods that depend on the storage pool from being scheduled to the node. • Exceptional scenario: The NPD add-on cannot detect the loss of all PVs (data disks), resulting in the loss of VGs (storage pools). In this case, kubelet automatically isolates the node, detects the loss of VGs (storage pools), and updates the corresponding resources in **nodestatus.allocatable** to **0**. This prevents pods that depend on the storage pool from being scheduled to the node. The damage of a single PV cannot be detected by this check item, but by the ReadonlyFilesystem check item.
PV storage pool error LocalPvVolumeGroupStatusError	Check the PV group on the node. Impact: Pods that depend on the storage pool cannot write data to the persistent volume. The persistent volume is remounted as a read-only file system by the kernel due to an I/O error. Typical scenario: When creating a node, a user configures two data disks as a persistent volume storage pool. Some data disks are deleted by mistake.	

Check Item	Function	Description
Mount point error MountPointProblem	Check the mount point on the node. Exceptional definition: You cannot access the mount point by running the **cd** command. Typical scenario: Network File System (NFS), for example, obsfs and s3fs is mounted to a node. When the connection is abnormal due to network or peer NFS server exceptions, all processes that access the mount point are suspended. For example, during a cluster upgrade, a kubelet is restarted, and all mount points are scanned. If the abnormal mount point is detected, the upgrade fails.	Alternatively, you can run the following command: `for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir; done && echo "ok"`

Check Item	Function	Description
Suspended disk I/O DiskHung	Check whether I/O suspension occurs on all disks on the node, that is, whether I/O read and write operations are not responded. Definition of I/O suspension: The system does not respond to disk I/O requests, and some processes are in the D state. Typical scenario: Disks cannot respond due to abnormal OS hard disk drivers or severe faults on the underlying network.	• Check object: all data disks • Source: /proc/diskstat Alternatively, you can run the following command: `iostat -xmt 1` • Thresholds: (All following conditions must be met.) – Average usage (**ioutil**) ≥ 0.99 – Average I/O queue length (**avgqu-sz**) ≥ 1 – Average I/O transfer volume ≤ 1 Average I/O transfer volume = Number of writes completed per second (**iops**, unit: w/s) + Amount of data written per second (**ioth**, unit: wMB/s) **NOTE** In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of **iowait** should be greater than 0.8.

Check Item	Function	Description
Slow disk I/O DiskSlow	Check whether all disks on the node have slow I/Os, that is, whether I/Os respond slowly. Typical scenario: EVS disks have slow I/Os due to network fluctuation.	• Check object: all data disks • Source: /proc/diskstat Alternatively, you can run the following command: `iostat -xmt 1` • Default threshold: Average I/O latency (**await**) ≥ 5000 ms **NOTE** If I/O requests are not responded and the **await** data is not updated, this check item is invalid.

Table 3-81 Other check items

Check Item	Function	Description
Abnormal NTP NTPProblem	Check whether the node clock synchronization service ntpd or chronyd is running properly and whether a system time drift is caused.	Default clock offset threshold: 8000 ms
Process D error ProcessD	Check whether there is a process D on the node.	Default threshold: 10 abnormal processes detected for three consecutive times Source: • /proc/{PID}/stat • Alternately, you can run the **ps aux** command. Exceptional scenario: The ProcessD check item ignores the resident D processes (heartbeat and update) on which the SDI driver on the BMS node depends.
Process Z error ProcessZ	Check whether the node has processes in Z state.	

Check Item	Function	Description
ResolvConf error ResolvConfFileProblem	Check whether the ResolvConf file is lost. Check whether the ResolvConf file is normal. Exceptional definition: No upstream domain name resolution server (nameserver) is included.	Object: **/etc/resolv.conf**
Existing scheduled event ScheduledEvent	Check whether scheduled live migration events exist on the node. A live migration plan event is usually triggered by a hardware fault and is an automatic fault rectification method at the IaaS layer. Typical scenario: The host is faulty. For example, the fan is damaged or the disk has bad sectors. As a result, live migration is triggered for VMs.	Source: • http://169.254.169.254/meta-data/latest/events/scheduled This check item is an Alpha feature and is disabled by default.
The spot price node is being reclaimed. SpotPriceNodeReclaimNotification	Check whether the reclaiming of a spot price node is interrupted due to preemption.	Default check interval: 120 seconds Default fault handling policy: evicts node loads.

The kubelet component has the following default check items, which have bugs or defects. You can fix them by upgrading the cluster or using NPD.

Table 3-82 Default kubelet check items

Check Item	Function	Description
Insufficient PID resources PIDPressure	Check whether PIDs are sufficient.	• Interval: 10 seconds • Threshold: 90% • Defect: In community version 1.23.1 and earlier versions, this check item becomes invalid when over 65535 PIDs are used. For details, see **issue 107107**. In community version 1.24 and earlier versions, thread-max is not considered in this check item.
Insufficient memory MemoryPressure	Check whether the allocable memory for the containers is sufficient.	• Interval: 10 seconds • Threshold: max. 100 MiB • Allocable = Total memory of a node – Reserved memory of a node • Defect: This check item checks only the memory consumed by containers, and does not consider that consumed by other elements on the node.
Insufficient disk resources DiskPressure	Check the disk usage and inodes usage of the kubelet and Docker disks.	• Interval: 10 seconds • Threshold: 90%

3.9.8 Executing the Pre- or Post-installation Commands During Node Creation

Background

When creating a node, use the pre- or -installation commands to install tools or perform security hardening on the node. This section provides guidance for you to correctly use the pre- or post-installation scripts. To use advanced installation scripts, store the scripts in OBS buckets to prevent problems such as excessive characters in the scripts. For details, see **Using OBS Buckets to Implement Custom Script Injection During Node Creation**.

Precautions

- Do not use pre- or post-installation scripts that take a long time to execute.

  The pre-installation script has a 15-minute time limit, while the post-installation script has a 30-minute time limit. If the node is not available within the designated time, the node reclaim process will be initiated. Therefore, do not use pre- or post-installation scripts that take a long time to execute.

- Do not directly use **reboot** in the script.

  CCE executes the post-installation command after installing mandatory components on a node. The node will be available only after the post-installation command is executed. If you run **reboot** directly, the node may be restarted before its status is reported. As a result, it cannot reach the running state within 30 minutes, and a rollback due to timeout will be triggered. Therefore, do not use **reboot**.

  If you need to restart a node, perform the following operations:

  – Run **shutdown -r** *<time >* in the script to delay the restart. For example, you can run **shutdown -r 1** to delay the restart for 1 minute.

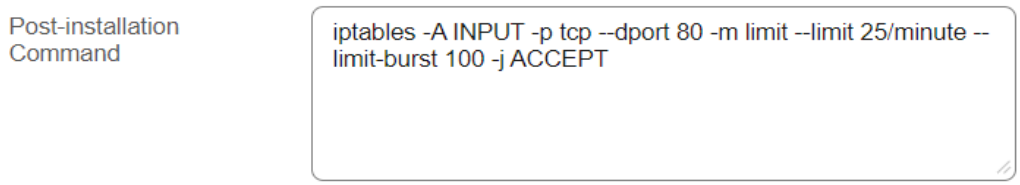
  – After the node is available, manually restart it.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Clusters**. Click the target cluster name to access the cluster console.

Step 2 Choose **Nodes** in the navigation pane, click the **Nodes** tab, click **Create Node** in the right corner, and configure the parameters.

Step 3 In the **Advanced Settings** area, enter pre- or post-installation commands.



For example, you can create iptables rules by running a post-installation command to allow a maximum of 25 TCP data packets to be addressed to port 80 per minute and allow a maximum of 100 data packets to be addressed to the port when the limit is exceeded to prevent DDoS attacks.

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT
```

NOTE

The command example here is for reference only.

Step 4 After the configuration, enter the number of nodes to be purchased and click **Next: Confirm**.

Step 5 Click **Submit**.

----End

3.9.9 ECS Event Handling Suggestions

ECS Event Overview

On the CCE console, you can use ECSs as nodes to create highly available Kubernetes clusters. During routine O&M, hardware and software faults on hosts accommodating ECSs are proactively forecasted and prevented by Huawei Cloud.

If the faults cannot be avoided, the system generates and reports ECS events (like ECS redeployment and system maintenance) for the affected ECSs to minimize the impacts of resource unavailability or performance deterioration.

NOTE

You can manage ECS events on the ECS console. For details, see **ECS Event Overview**.

Scenarios

During ECS event handling, the ECS may be unavailable, which makes the node and pods cannot run properly. To prevent any disruption to your services, when the system generates ECS events for your node, you are advised to migrate services running on the node to other available nodes in advance before handling the ECS events.

Procedure

Step 1 Perform pre-processing.

Before handling ECS events, you need to migrate services to other available nodes and then isolate the faulty node.

You can refer to **Draining a Node** to migrate pods running on the node and then isolate the node. Alternatively, you can manually migrate pods running on the node and then refer to **Managing Node Taints** to add a **NoSchedule** taint to the node for its isolation.

You can also migrate services to other available nodes, create a node, and delete the faulty node to skip subsequent steps.

Step 2 Handle events.

Go to the ECS console to **query events** and handle ECS events by event type.

Step 3 Perform post-processing.

After processing the ECS events, cancel the node isolation.

To restore the node to the schedulable state, you can click **More** > **Enable Scheduling** on the **Nodes** page or remove the **NoSchedule** taint of the node by referring to **Managing Node Taints**.

----End

4 Node Pools

4.1 Node Pool Overview

Introduction

CCE introduces node pools to help you better manage nodes in Kubernetes clusters. A node pool contains one node or a group of nodes with identical configuration in a cluster.

You can create custom node pools on the CCE console. With node pools, you can quickly create, manage, and destroy nodes without affecting the cluster. All nodes in a custom node pool have identical parameters and node type. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool.

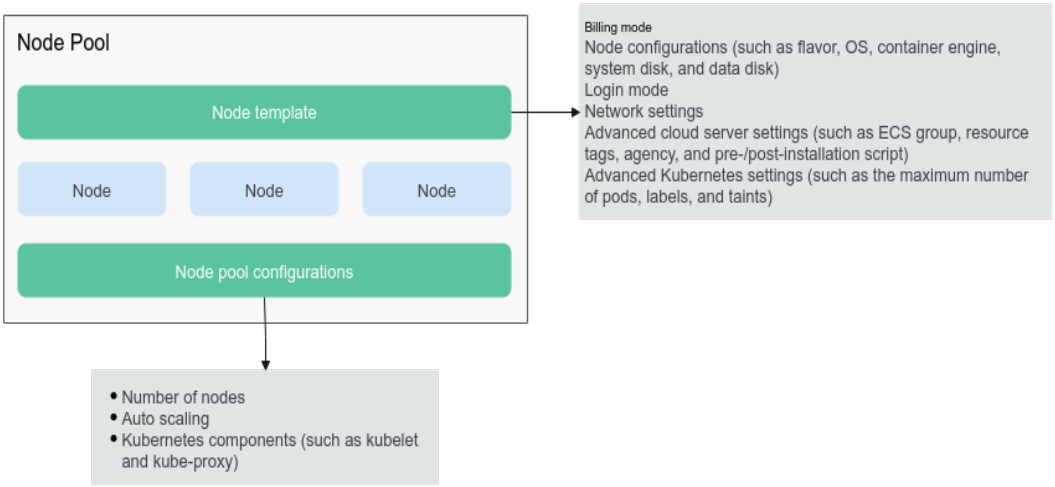
You can also use node pools for auto scaling (supported only by pay-per-use node pools).

- When a pod in a cluster cannot be scheduled due to insufficient resources, scale-out can be automatically triggered.
- When there is an idle node or a monitoring metric threshold is met, scale-in can be automatically triggered.

This section describes how node pools work in CCE and how to create and manage node pools.

Node Pool Architecture

Figure 4-1 Overall architecture of a node pool



Generally, all nodes in a node pool have the following same attributes:

- Node OS
- Node login mode
- Node container runtime
- Enterprise project
- Startup parameters of Kubernetes components on a node
- Custom startup script of a node
- Kubernetes labels and taints

CCE provides the following extended attributes for node pools:

- Node pool OS
- Maximum number of pods on each node in a node pool

Notes on a Node Pool Upgrade

For clusters of v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later versions, new node pools support the creation of both pay-per-use and yearly/monthly nodes by default, providing you with better resource management experience.

The highlights of new node pools are as follows:

- Both pay-per-use and yearly/monthly nodes can be created in one node pool, and the required duration of yearly/monthly nodes can be different.
- Pay-per-use and yearly/monthly nodes can be accepted for management.
- Auto scaling can be enabled and various scaling policies can be configured for more efficient, flexible resource management.

The following table lists the changes after an existing node pool is upgraded.

Billing Mode of the Original Node Pool	Changes After an Upgrade
Pay-per-use	All capabilities of the original node pool will be automatically inherited. Yearly/Monthly nodes created in the new node pool cannot be manually scaled in. They can only be deleted or unsubscribed from.
Yearly/Monthly	The original node pool can be hitlessly upgraded, without affecting the existing nodes in the node pool. After the upgrade, the following changes will occur: • Nodes created in the new node pools will be billed on a pay-per-use basis by default. • Yearly/Monthly nodes created in the new node pool cannot be manually scaled in. They can only be deleted or unsubscribed from. • Auto scaling can be enabled. By default, only pay-per-use nodes will be scaled by default. • The API for updating a node pool cannot be used to create yearly/monthly nodes. To create yearly/monthly nodes, use the API for resizing a node pool.

Description of DefaultPool

DefaultPool is not a real node pool. It only **classifies** nodes that are not in the custom node pools. These nodes are directly created on the console or by calling APIs. DefaultPool does not support any user-created node pool functions, including scaling and parameter configuration. DefaultPool cannot be edited, deleted, expanded, or auto scaled, and nodes in it cannot be migrated.

Applicable Scenarios

When a large-scale cluster is required, you are advised to use node pools to manage nodes.

The following table describes multiple scenarios of large-scale cluster management and the functions of node pools in each scenario.

Table 4-1 Using node pools for different management scenarios

Scenario	Function
Multiple heterogeneous nodes (with different models and configurations) in the cluster	Nodes can be grouped into different pools for management.
Frequent node scaling required in a cluster	Node pools support auto scaling to dynamically add or reduce nodes.

Scenario	Function
Complex application scheduling rules in a cluster	Node pool tags can be used to quickly specify service scheduling rules.

Functions and Precautions

Function	Description	Precaution
Creating a node pool	Add a node pool.	It is recommended that a cluster contains no more than 100 node pools.
Deleting a node pool	Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools.	If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.
Enabling auto scaling for a node pool	After auto scaling is enabled, nodes will be automatically created or deleted in the node pool based on the cluster loads.	Do not store important data on nodes in a node pool because the nodes may be deleted after scale-in. Data on the deleted nodes cannot be restored.
Disabling auto scaling for a node pool	After auto scaling is disabled, the number of nodes in a node pool will not automatically change with the cluster loads.	None
Adjusting the size of a node pool	The number of nodes in a node pool can be directly adjusted. If the number of nodes is reduced, nodes are randomly removed from the current node pool.	After auto scaling is enabled, you are not advised to manually adjust the node pool size.
Modifying node pool configurations	You can change the node pool name and number of nodes, add or delete Kubernetes labels, resource tags, and taints, and adjust node pool configurations such as the disk, OS, and container engine of the node pool.	The deleted or added Kubernetes labels and taints (as well as their quantity) will apply to all nodes in the node pool, which may cause pod re-scheduling. Therefore, exercise caution when performing this operation.

Function	Description	Precaution
Removing a node from a node pool	Nodes in a node pool can be migrated to the default node pool of the same cluster.	Nodes in the default node pool cannot be migrated to other node pools, and nodes in a user-created node pool cannot be migrated to other user-created node pools.
Copying a node pool	You can copy the configuration of an existing node pool to create a new node pool.	None
Setting Kubernetes parameters	You can configure core components with fine granularity.	• This function is supported only in clusters of v1.15 or later. It is not displayed for versions earlier than v1.15. • The default node pool does not support this type of configuration.

Deploying a Workload in a Specified Node Pool

When configuring a workload, you can set the workload affinity and node affinity on the **Scheduling** tab to forcibly deploy the workload to a specific node pool. This way, the workload runs only on nodes in that node pool. To better control where the workload is to be scheduled, you can use affinity or anti-affinity policies between workloads and nodes described in **Configuring Node Affinity Scheduling (nodeAffinity)**.

For example, you can use container's resource request as a nodeSelector so that workloads will run only on the nodes that meet the resource request.

If the workload definition file defines a container that requires four CPUs, the scheduler will not choose the nodes with two CPUs to run workloads.

Related Operations

You can log in to the CCE console and refer to the following sections to perform operations on node pools:

- **Creating a Node Pool**
- **Managing a Node Pool**
- **Creating a Deployment**
- **Configuring Node Affinity Scheduling (nodeAffinity)**

4.2 Upgrading to New Node Pools

Upgraded node pools provide additional benefits beyond flexible payment options (pay-per-use and yearly/monthly). They also enhance configuration management and improve resource management efficiency and flexibility.

Why to Upgrade to the New Version?

- **Flexible resource configuration**: New node pools allow you to create pay-per-use nodes and select more cost-effective yearly/monthly nodes, depending on your needs.
- **Multiple flavors available**: You can easily filter different flavors for nodes, such as GPU nodes, based on CPU and memory parameters to meet the specific requirements of various service scenarios.
- **Advanced auto scaling**: New node pools can automatically scale and you can configure multiple scaling policies to optimize resource utilization for different service scenarios.
- **Enhanced configuration management**: New node pools offer improved customization options for Kubernetes parameters and provide better guidance to meet the complex requirements of managing containerized applications.

What Are Changes in the New Version?

- Yearly/Monthly nodes created in the new node pools cannot be manually scaled in. They can only be deleted or unsubscribed from.
- During auto scaling, only pay-per-use nodes are scaled in or out by default.
- You cannot create yearly/monthly nodes using the API for updating a node pool. Instead, you must increase or decrease them.
- By default, nodes created in new yearly/monthly node pools will be billed on a pay-per-use basis.

Table 4-2 Comparison before and after a node pool upgrade

Type	Before the Upgrade		After the Upgrade
	Pay-per-Use Node Pool	Yearly/Monthly Node Pool	
Billing mode of nodes	Pay-per-use only	Yearly/Monthly only	Both pay-per-use and yearly/monthly
Node scale-out	Change the number of nodes in the node pool configuration.	Change the number of nodes in the node pool configuration.	Scale in or out the node pool.

Type	Before the Upgrade		After the Upgrade
	Pay-per-Use Node Pool	Yearly/Monthly Node Pool	
Node scale-in	• Change the number of nodes in the node pool configuration. • Delete or remove nodes.	• Change the number of nodes in the node pool configuration. • Unsubscribe from or remove nodes.	Pay-per-use nodes: Decrease them. Yearly/Monthly nodes: Unsubscribe from or remove them.
Upgrade	Upgrade the cluster to v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later. • After a pay-per-use node pool is upgraded to the new version, all features of the new version will be readily accessible without any additional steps. • Increasing or decreasing the number of nodes in a yearly/monthly node pool will trigger an automatic upgrade to the new version. Upon confirmation, the node pool will be upgraded without affecting the existing nodes. **NOTE** After a node pool is upgraded, any nodes added using the API for updating a node pool will be pay-per-use by default.		

How to Upgrade to the New Version?

You need to upgrade your cluster to v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, or later, and then perform the following operations to trigger the upgrade process of new node pool.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Click **Scaling** in the **Operation** column of the target node pool. In the displayed dialog box, click **Upgrade Now**.

----End

4.3 Creating a Node Pool

Scenario

This section describes how to create a node pool and perform operations on the node pool. For details about how a node pool works, see **Node Pool Overview**.

Procedure

Step 1 Log in to the **CCE console**.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 In the upper right corner of the page, click **Create Node Pool**.

Basic Settings

Table 4-3 Basic settings

Parameter	Description
Node Pool Name	Name of a node pool. By default, the name is in the format of *Cluster name*-nodepool-*Random number*. If you do not want to use the default name format, you can customize the name.
Enterprise Project	This parameter is available only for enterprise users who have enabled an enterprise project, and the cluster version must be v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later. After an enterprise project is selected, nodes will be created in the node pool within that project. To manage clusters and other resources like nodes, load balancers, and node security groups, you can use the Enterprise Project Management Service (EPS). For more details, see **Enterprise Management**.

Node Configuration

You can configure the flavor and OS of a cloud server, on which your containerized applications run.

Table 4-4 Node configuration parameters

Parameter	Description
Node Type	Select a node type based on service requirements. Then, you can select a proper flavor from the node flavor list. CCE standard clusters support the following node types: • ECS (VM): A virtualized ECS is used as a cluster node. • ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node. • BMS: A traditional bare metal server is used as a cluster node. Local disks of bare metal servers can be used as data disks. CCE Turbo clusters support the following node types: • ECS (VM): A virtualized ECS is used as a cluster node. A CCE Turbo cluster supports only the cloud servers that allow multiple ENIs. Select a server type displayed on the CCE console. • ECS (physical machine): A QingTian-backed bare metal server is used as a cluster node.
Specifications	Select a node flavor based on service requirements. The available node flavors vary depending on regions. For details, see the CCE console. **NOTE** • If a node pool is configured with multiple node flavors, only the flavors (which can be located in different AZs) of the same node type are supported. For example, a node pool consisting of general computing-plus nodes supports only general computing-plus node flavors, but not the flavors of general computing nodes. • A maximum of 10 node flavors can be added to a node pool (the flavors in different AZs are counted separately). When adding a node flavor, you can choose multiple AZs, but you need to specify them. • Nodes in a newly created node pool are created using the default flavor. If the resources for the default flavor are insufficient, node creation will fail. • After a node pool is created, the flavors of existing nodes cannot be deleted.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see **Mapping Between Node OSs and Container Engines**.

Parameter	Description
OS	Select an OS type. Different types of nodes support different OSs. • **Public image**: Select a public image for the node. • **Private image**: Select a private image for the node. For details about how to create a private image, see **Creating a Custom CCE Node Image**. **NOTE** Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS.
Login Mode	• **Password** The default username is **root**. Enter the password for logging in to the node and confirm the password. Be sure to remember the password as you will need it when you log in to the node. • **Key Pair** Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**. • **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images) Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image.

Storage Settings

Configure storage resources on a node for the containers running on it. Select a disk type and configure its size based on service requirements. For details about EVS disks, see **Disk Types and Performance**.

Table 4-5 Configuration parameters

Parameter	Description
System Disk	System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB. **NOTE** General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**. General-purpose SSD V2 EVS disks are available only in clusters of v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **System Disk Encryption**: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. Only the nodes of the **Elastic Cloud Server (VM)** type in certain regions support system disk encryption. For details, see the console. • **Not encrypted** is selected by default. • If you select **Enabled (key)** for **System Disk Encryption**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box. • If you select **Enabled (KMS key ID)** for **System Disk Encryption**, enter a KMS key (which can be shared by others) in the current region.
System Component Storage	Select a disk for storing system components. • **Data Disk**: added for storing container runtime and kubelet components by default. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. This data disk cannot be deleted or detached. Otherwise, the node will be unavailable. • **System Disk**: stores CCE resources such as downloaded images, ephemeral storage for containers, and container stdout logs. If the system disk is fully occupied, it will negatively affect the stability of the node. **NOTE** In clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later, you can select a disk for storing system components. If **CCE Node Problem Detector** is used, ensure that its version is 1.19.2 or later.

Parameter	Description
Data Disk	• **At least one default data disk must be added** for storing container runtime and kubelet components if **System Component Storage** is set to **Data Disk**. **This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.** This function is available for clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, or v1.29.4-r0. – Default data disk: used for container runtime and kubelet components. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. – Other common data disks: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. • If **System Component Storage** is set to **System Disk**, you do not need to add a default data disk. In this case, all data disks are common ones: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. This function is available for clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later versions. **NOTE** • If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk. • Local disks may break down and do not ensure data reliability. Store your service data in EVS disks, which are more reliable than local disks. • General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**. General-purpose SSD V2 EVS disks are available only in clusters of v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. **Advanced Settings** Expand the area and configure the following parameters: • **Data Disk Space Allocation**: allocates space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. • **Data Disk Encryption**: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **BMS nodes do not support data disk encryption, and this option is available only in certain regions. For details, see the console.** – **Not encrypted** is selected by default. – If you select **Enabled (key)** for **Data Disk Encryption**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box.

Parameter	Description
	– If you select **Enabled (KMS key ID)** for **Data Disk Encryption**, enter a KMS key (which can be shared by others) in the current region. **Adding data disks** A maximum of 16 data disks can be attached to an ECS and 10 to a BMS. By default, a raw disk is created without any processing. You can also click **Expand** and select any of the following options: • **Default**: By default, a raw disk is created without any processing. • **Mount Disk**: The data disk is attached to a specified directory. • **Use as PV**: applicable when there is a high performance requirement on PVs. The **node.kubernetes.io/local-storage-persistent** label is added to the node with PV configured. The value is **linear** or **striped**. • **Use as ephemeral volume**: applicable when there is a high performance requirement on emptyDir. **PVs** and **EVs** support the following write modes: • **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up. • **Striped**: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out. This option can be selected only when there are multiple volumes. **NOTE** • Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended. • Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 4-6 Configuration parameters

Parameter	Description
VPC	The VPC to which the cluster belongs by default, which cannot be changed.

Parameter	Description
Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead. • Multiple subnets: You can select multiple subnets in the same VPC for nodes. Newly added nodes will preferentially use the IP addresses from the top-ranking subnet. • Single subnet: Only one subnet is configured for your node pool. If the IP addresses of a single subnet are insufficient, configure multiple subnets. Otherwise, a node pool scale-out may fail.
Node IP Address	Random allocation is supported.
Associate Security Group	Security group used by the nodes created in the node pool. A maximum of five security groups can be selected. When a cluster is created, a node security group named **{*Cluster name*}-cce-node-{*Random ID*}** is created and used by default. Traffic needs to pass through certain ports in the node security group to ensure node communications. Ensure that you have enabled these ports if you select another security group. For details, see **How Can I Configure a Security Group Rule in a Cluster?** **NOTE** After a node pool is created, its associated security group cannot be modified.

Advanced Settings

Configure advanced node capabilities such as labels, taints, and startup command.

Table 4-7 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag.

Parameter	Description
Kubernetes Label	A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click **Add Label** for more. A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see **Labels and Selectors**.
Taint	This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters: • **Key**: A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. • **Value**: A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. For details, see **Managing Node Taints**. **NOTE** For a cluster of v1.19 or earlier, the workload may have been scheduled to a node before the taint is added. To avoid such a situation, select a cluster of v1.19 or later.
Synchronization for Existing Nodes	After the options are selected, changes to resource tags and Kubernetes labels/taints in a node pool will be synchronized to existing nodes in the node pool.

Parameter	Description
New Node Scheduling	Default scheduling policy for the nodes newly added to a node pool. If you select **Unschedulable**, newly created nodes in the node pool will be labeled as unschedulable. In this way, you can perform some operations on the nodes before pods are scheduled to these nodes. **Scheduled Scheduling**: After scheduled scheduling is enabled, new nodes will be automatically scheduled after the custom time expires. • **Disabled**: By default, scheduled scheduling is not enabled for new nodes. To manually enable this function, go to the node list. For details, see **Configuring a Node Scheduling Policy in One-Click Mode**. • **Custom**: the default timeout for unschedulable nodes. The value ranges from 0 to 99 in the unit of minutes. **NOTE** • If auto scaling of node pools is also required, ensure the scheduled scheduling is less than 15 minutes. If a node added through Autoscaler cannot be scheduled for more than 15 minutes, Autoscaler determines that the scale-out failed and triggers another scale-out. Additionally, if the node cannot be scheduled for more than 20 minutes, the node will be scaled in by Autoscaler. • After this function is enabled, nodes will be tainted with **node.cloudprovider.kubernetes.io/uninitialized** during a node pool creation or update.
Max. Pods	Maximum number of pods that can run on the node, including the default system pods. Value range: 16 to 256 This limit prevents the node from being overloaded with pods. This number is also decided by other factors. For details, see **Maximum Number of Pods That Can Be Created on a Node**.
ECS Group	An ECS group logically groups ECSs. The ECSs in the same ECS group comply with the same policy associated with the ECS group. Anti-affinity: ECSs in an ECS group are deployed on different physical hosts to improve service reliability. Select an existing ECS group, or click **Add ECS Group** to create one. After the ECS group is created, click the refresh icon.
Pre-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.

Parameter	Description
Post-installation Command	Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded. The script will be executed after Kubernetes software is installed, which does not affect the installation. **NOTE** Do not run the **reboot** command in the post-installation script to restart the system immediately. To restart the system, run the **shutdown -r 1** command to restart with a delay of one minute.
Agency	An agency is created by the tenant administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. If no agency is available, click **Create Agency** on the right to create one.
Custom Prefix and Suffix	Custom name prefix and suffix of a node in a node pool. After the configuration, the nodes in the node pool will be named with the configured prefix and suffix. For example, if the prefix is **prefix-** and the suffix is **-suffix**, the nodes in the node pool will be named in the format of "prefix-Node pool name with five-digit random characters-suffix". **NOTICE** ● A prefix and suffix can be customized only when a node pool is created, and they cannot be modified after the node pool is created. ● A prefix can end with a special character, and a suffix can start with a special character. ● A node name consists of a maximum of 56 characters in the format of "Prefix-Node pool name with five-digit random characters-Suffix". ● A node name does not support the combination of a period (.) and special characters (such as .., .-, or -.). ● This function is available only in clusters of v1.28.1, v1.27.3, v1.25.6, v1.23.11, v1.21.12, or later.

Parameter	Description
Kubernetes Node Name	The Kubernetes node name is the value of **metadata.labels.kubernetes.io/hostname** in the YAML file of the node. The following two values are supported: • **Node private IP** (default) • **Cloud server name**: Use the custom cloud server name configured in node settings. Cloud server names may be duplicate. To prevent name conflicts, CCE randomly adds a five-digit random suffix to the end of each cloud server name. **NOTICE** – This function is available only when the cluster version is v1.23.4-r0 or later. – The name of a cloud server can be specified as the name of a Kubernetes node only when the cloud server is created or managed. After the cloud server is created or managed, the Kubernetes node name cannot be changed. For details, see **ECS Names, Node Names, and Kubernetes Node Names**. – Existing nodes in the cluster still use the private IP address as the Kubernetes node name. Newly created or accepted nodes can use cloud server names. In this scenario, some Kubernetes node names may be inconsistent with node private IP addresses, and adaptation is required. For example, when configuring node affinity, you cannot use the node private IP address as the node name to configure a scheduling policy. To change the Kubernetes node name of the existing nodes to the cloud server name, remove these nodes from the cluster and accept them again. Before doing so, learn about the possible impacts on services when **removing** or **accepting a node**.

Step 4 Click **Next: Confirm**. Ensure that you have read and understood the **Image Management Service Statement** .

Step 5 Click **Submit**.

----End

Related Operations

By default, the total number of nodes in a node pool is set to 0 upon creation. You will need to manually increase the number of nodes. For details, see **Scaling a Node Pool**.

4.4 Scaling a Node Pool

You can specify a specification in a node pool for scaling.

NOTICE

The default node pool does not support scaling. Use **Creating a Node** to add a node.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **Scaling** next to the target node pool.

Step 4 In the displayed **Node Pool Scaling** window, configure scaling parameters.

- Add or reduce nodes for scaling.
- Use the selected flavor to increase or decrease the number of nodes.
- Select a billing mode only when adding nodes.
  - Yearly/Monthly

    You must specify the required duration if **Yearly/Monthly** is selected. You can choose whether to select **Auto-renew** based on site requirements. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year.
  - Pay-per-use

    Resources will be billed based on usage duration. You can provision or delete resources at any time.
  - Spot price

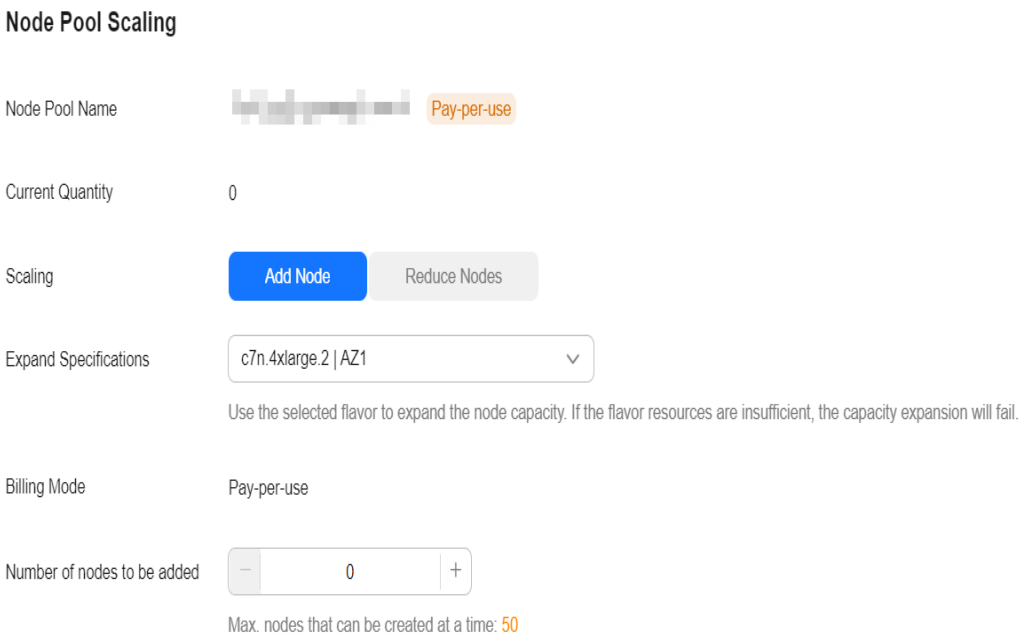
    A postpaid billing mode, in which a spot cloud server will be billed based on the service duration at a lower price than that of a pay-per-use ECS cloud server with the same specifications. For details, see **Spot Pricing ECSs**.

    NOTE

    - If you purchase both a data disk and an EIP when creating a spot pricing ECS, they will be automatically released when the ECS is released. If you attach data disks or bind EIPs to an existing spot pricing ECS, these resources must be manually released after the ECS is deleted.
    - Spot pricing ECSs cannot be reset, accepted for management, removed, migrated, changed to yearly/monthly, or upgraded by resetting the cluster. Before resetting a cluster for an upgrade, delete spot pricing nodes and set the number of pods in the spot pricing node pool to 0.
    - Spot pricing ECSs may be released if the quoted price is lower than the market price or if there are not enough resources available. Install the latest NPD add-on in the cluster so that you can receive a notification 5 minutes before a spot pricing ECS is released. This add-on generates a ReceivedReclaimNodeNotification event, adds taint **node-problem-controller.cce.io/SpotPriceNodeReclaimNotification: NoExecute** to the node (spot pricing ECS), and evicts pods on the node. This allows pods to be migrated to other nodes before the spot pricing ECS is deleted.
- Configure the number of nodes to be added or deleted.

- When scaling out a node pool, make sure that the total number of nodes, both existing and new, does not exceed the management scale of the current cluster.
- When scaling in a node pool, make sure that the number of nodes to be removed does not exceed the number of nodes currently in the pool.

  Scaling in can result in the unavailability of resources associated with a node, such as local storage and workloads that were scheduled to that node. Exercise caution when performing this operation to avoid impact on running services.



Step 5 Click **OK**.

----End

4.5 Managing a Node Pool

4.5.1 Updating a Node Pool

Precautions

- Only clusters of v1.19 or later support the modification of the container engine, OS, system/data disk size, data disk space allocation, and pre-installation/post-installation script configuration.
- Changes to the container engine, OS, or pre-/post-installation script in a node pool take effect only on new nodes. To synchronize the modification onto existing nodes, manually reset these nodes.
- The modification of data disk space allocation and the system/data disk size of a node pool takes effect only for new nodes. The configuration cannot be synchronized even if the existing nodes are reset.

- Changes to resource tags and Kubernetes labels/taints in a node pool will be automatically synchronized to existing nodes after the options of **Synchronization for Existing Nodes** are selected. You do not need to reset these nodes.

Updating a Node Pool

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Click **Update** next to the name of the node pool you will edit. Configure the parameters in the displayed **Update Node Pool** page.

Basic Settings

Table 4-8 Basic settings

Parameter	Description
Node Pool Name	Name of the node pool.
Enterprise Project	This parameter is available only for enterprise users who have enabled an enterprise project. After an enterprise project is selected, nodes will be created in the node pool within that project. To manage clusters and other resources like nodes, load balancers, and node security groups, you can use the Enterprise Project Management Service (EPS). For more details, see **Enterprise Management**.

Node Configuration

Table 4-9 Node configuration parameters

Parameter	Description
Specifications	Select node specifications that best fit your service needs. **NOTE** • If a node pool is configured with multiple node flavors, only the flavors (which can be located in different AZs) of the same node type are supported. For example, a node pool consisting of general computing-plus nodes supports only general computing-plus node flavors, but not the flavors of general computing nodes. • A maximum of 10 node flavors can be added to a node pool (the flavors in different AZs are counted separately). When adding a node flavor, you can choose multiple AZs, but you need to specify them. • Nodes in a newly created node pool are created using the default flavor. If the resources for the default flavor are insufficient, node creation will fail. • After a node pool is created, the flavors of existing nodes cannot be deleted.
Container Engine	The container engines supported by CCE include Docker and containerd, which may vary depending on cluster types, cluster versions, and OSs. Select a container engine based on the information displayed on the CCE console. For details, see **Mapping Between Node OSs and Container Engines**. **NOTE** After the container engine is modified, the modification automatically takes effect on newly added nodes. For existing nodes, manually reset the nodes for the modification to take effect.
OS	Select an OS type. Different types of nodes support different OSs. • **Public image**: Select a public image for the node. • **Private image**: Select a private image for the node. For details about how to create a private image, see **Creating a Custom CCE Node Image**. **NOTE** • Service container runtimes share the kernel and underlying calls of nodes. To ensure compatibility, select a Linux distribution version that is the same as or close to that of the final service container image for the node OS. • After the OS is modified, the modification automatically takes effect on newly added nodes. Manually reset existing nodes for the modification to take effect.

Parameter	Description
Modify Login Settings	After this function is enabled, you can modify the node login mode. • **Password** The default username is **root**. Enter the password for logging in to the node and confirm the password. Be sure to remember the password as you will need it when you log in to the node. • **Key Pair** Select the key pair used to log in to the node. You can select a shared key. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**. • **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images) Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image. **NOTE** The changes made to the key pair will be applied automatically to any new nodes added. However, for existing nodes, you will need to manually reset them for the modifications to take effect.

Storage Settings

Table 4-10 Configuration parameters

Parameter	Description
System Disk	System disk used by the node OS. The value ranges from 40 GiB to 1024 GiB. The default value is 50 GiB. **NOTE** After the system disk configuration is modified, the modification takes effect only on newly added nodes. The configuration cannot be synchronized to existing nodes even if they are reset. **System Disk Encryption**: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. Only the nodes of the **Elastic Cloud Server (VM)** type in certain regions support system disk encryption. For details, see the console. • **Not encrypted** is selected by default. • After setting **System Disk Encryption** to **Enabled (key)**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box. **NOTE** The modified system disk encryption automatically takes effect on new nodes. For existing nodes, manually reset the nodes for the modification to take effect.

Parameter	Description
Data Disk	• **At least one default data disk must be added** for storing container runtime and kubelet components if **System Component Storage** is set to **Data Disk**. **This data disk cannot be deleted or detached. Otherwise, the node will be unavailable.** This function is available for clusters of a version earlier than v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, or v1.29.4-r0. – Default data disk: used for container runtime and kubelet components. The disk size ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. – Other common data disks: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. • If **System Component Storage** is set to **System Disk**, you do not need to add a default data disk. In this case, all data disks are common ones: You can set the data disk size to a value ranging from 10 GiB to 32768 GiB. The default value is 100 GiB. This function is available for clusters of v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, or later versions. **NOTE** After the data disk configuration is modified, the modification takes effect only on newly added nodes. The configuration cannot be synchronized to existing nodes even if they are reset. **Advanced Settings** Expand the area and configure the following parameters: • **Data Disk Space Allocation**: allocates space for container engines, images, and ephemeral storage for them to run properly. For details about how to allocate data disk space, see **Space Allocation of a Data Disk**. **NOTE** After the data disk space allocation configuration is modified, the modification takes effect only for new nodes. The configuration cannot take effect for the existing nodes even if they are reset. • **Enabled**: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **BMS nodes do not support data disk encryption, and this option is available only in certain regions. For details, see the console.** – **Not encrypted** is selected by default. – After setting **Data Disk Encryption** to **Enabled**, choose an existing key. If no key is available, click **View Key List** and create a key. After the key is created, click the refresh icon next to the text box.

Parameter	Description
	NOTE After the **Data Disk Encryption** is modified, the modification takes effect only on newly added nodes. The configuration cannot be synchronized to existing nodes even if they are reset. **Adding data disks** A maximum of 16 data disks can be attached to an ECS and 10 to a BMS. By default, a raw disk is created without any processing. You can also click **Expand** and select any of the following options: • **Default**: By default, a raw disk is created without any processing. • **Mount Disk**: The data disk is attached to a specified directory. • **Use as PV**: applicable when there is a high performance requirement on PVs. The **node.kubernetes.io/local-storage-persistent** label is added to the node with PV configured. The value is **linear** or **striped**. • **Use as ephemeral volume**: applicable when there is a high performance requirement on emptyDir. **PVs** and **EVs** support the following write modes: • **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up. • **Striped**: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. A storage pool consisting of striped volumes cannot be scaled-out. This option can be selected only when there are multiple volumes. **NOTE** • Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended. • Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later. **Local Disk Description** If the node flavor is disk-intensive or ultra-high I/O, one data disk can be a local disk. Local disks may break down and do not ensure data reliability. Store your service data in EVS disks, which are more reliable than local disks.

Network Settings

Table 4-11 Configuration parameters

Parameter	Description
VPC	The VPC to which the cluster belongs by default, which cannot be changed.
Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead. • Multiple subnets: You can select multiple subnets in the same VPC for nodes. Newly added nodes will preferentially use the IP addresses from the top-ranking subnet. • Single subnet: Only one subnet is configured for your node pool. If the IP addresses of a single subnet are insufficient, configure multiple subnets. Otherwise, a node pool scale-out may fail.

Advanced Settings

Table 4-12 Advanced settings

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag. **NOTE** Modified resource tags automatically take effect on new nodes as well as existing nodes if **Resource tags synchronized** is selected in **Synchronization for Existing Nodes**.
Kubernetes Label	A key-value pair added to a Kubernetes object (such as a pod). After specifying a label, click **Add Label** for more. A maximum of 20 labels can be added. Labels can be used to distinguish nodes. With workload affinity settings, container pods can be scheduled to a specified node. For more information, see **Labels and Selectors**. **NOTE** Modified Kubernetes labels automatically take effect on new nodes as well as existing nodes if **Kubernetes labels** is selected in **Synchronization for Existing Nodes**.

Parameter	Description
Taint	This parameter is left blank by default. You can add taints to configure anti-affinity for the node. A maximum of 20 taints are allowed for each node. Each taint contains the following parameters: • **Key**: A key must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. • **Value**: A value must contain 1 to 63 characters, starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. For details, see **Managing Node Taints**. **NOTE** Modified taints automatically take effect on new nodes as well as existing nodes if **Taints** is selected in **Synchronization for Existing Nodes**.

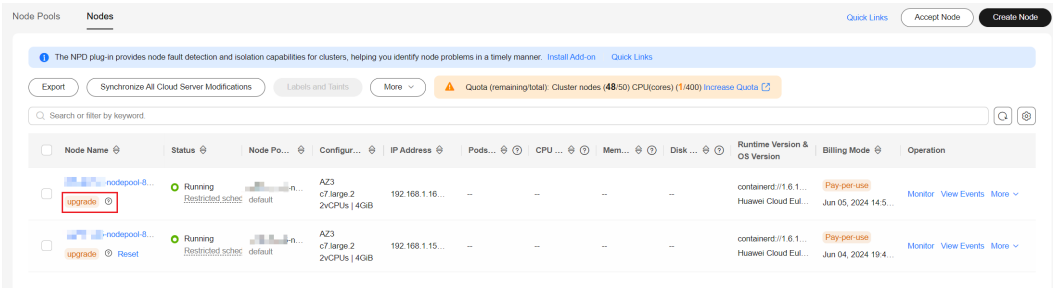
Parameter	Description
Synchronization for Existing Nodes	After the options are selected, changes to resource tags and Kubernetes labels/taints in a node pool will be synchronized to existing nodes in the node pool. **NOTE** When you update a node pool, pay attention to the following if you change the state of **Resource tags synchronized**: • After the option is selected: – CCE will synchronize the resource tags configured in the node pool to existing nodes. If a resource tag with the same key of a resource tag in the node pool already exists on an ECS, the value of the tag on the ECS will be changed to that of the resource tag in the node pool. – Typically, it takes less than 10 minutes to synchronize resource tags onto existing nodes, depending on the number of nodes in the node pool. – Issue a resource tag synchronization request only after the previous synchronization is complete. Otherwise, the resource tags may be inconsistent between existing nodes. When you update a node pool, pay attention to the following if you change the state of **Kubernetes labels** or **Taints**: • When these options are deselected, the Kubernetes labels/taints of the existing and new nodes in the node pool may be inconsistent. If service scheduling relies on node labels or taints, the scheduling may fail or the node pool may fail to scale. • When these options are selected: – If you have **modified** or **added** labels or taints in the node pool, the modifications will be automatically synchronized to existing nodes typically in 10 minutes after **Kubernetes labels** or **Taints** is selected. – If you have **deleted** a label or taint in the node pool, you must manually delete the label or taint on the node list page after **Kubernetes labels** or **Taints** is selected. – If you have **manually changed the key or effect of a taint on an existing node**, a new taint will be added to the existing node after **Kubernetes labels** or **Taints** is selected. In the new taint, its key is different from the manually changed key but its value and effect are the same as those manually changed ones, or its effect is different from the manually changed effect but its key and value are the same as those manually changed ones. This is because a Kubernetes taint natively uses a key and effect as a key-value pair. The taints with different keys or effects are considered as two taints.

Parameter	Description
New Node Scheduling	Default scheduling policy for the nodes newly added to a node pool. If you select **Unschedulable**, newly created nodes in the node pool will be labeled as unschedulable. In this way, you can perform some operations on the nodes before pods are scheduled to these nodes. **Scheduled Scheduling**: After scheduled scheduling is enabled, new nodes will be automatically scheduled after the custom time expires. • **Disabled**: By default, scheduled scheduling is not enabled for new nodes. To manually enable this function, go to the node list. For details, see **Configuring a Node Scheduling Policy in One-Click Mode**. • **Custom**: the default timeout for unschedulable nodes. The value ranges from 0 to 99 in the unit of minutes. **NOTE** • If auto scaling of node pools is also required, ensure the scheduled scheduling is less than 15 minutes. If a node added through Autoscaler cannot be scheduled for more than 15 minutes, Autoscaler determines that the scale-out failed and triggers another scale-out. Additionally, if the node cannot be scheduled for more than 20 minutes, the node will be scaled in by Autoscaler. • After this function is enabled, nodes will be tainted with **node.cloudprovider.kubernetes.io/uninitialized** during a node pool creation or update.
Agency	An agency is created by the account administrator on the IAM console. Using an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. If no agency is available, click **Create Agency** on the right to create one. **NOTE** After an agency is modified, the modification will only apply to new nodes and not to existing ones, even if they are reset.

Parameter	Description
Custom Prefix and Suffix	Custom name prefix and suffix of a node in a node pool. After the configuration, the nodes in the node pool will be named with the configured prefix and suffix. For example, if the prefix is **prefix-** and the suffix is **-suffix**, the nodes in the node pool will be named in the format of "prefix-Node pool name with five-digit random characters-suffix". • A prefix and suffix can be customized only when a node pool is created, and they cannot be modified after the node pool is created. • A prefix can end with a special character, and a suffix can start with a special character. • A node name consists of a maximum of 56 characters in the format of "Prefix-Node pool name with five-digit random characters-Suffix". • A node name does not support the combination of a period (.) and special characters (such as .., .-, or -.). • This function is available only in clusters of v1.28.1, v1.27.3, v1.25.6, v1.23.11, v1.21.12, or later. **NOTE** After the custom name prefix and suffix are modified, the modification will only apply to new nodes and not to existing ones, even if they are reset.

Step 4 After the configuration, click **OK**.

After node pool parameters are modified, you can find the update on the **Nodes** page. Reset the nodes in the target node pool to synchronize the configuration update.



----End

4.5.2 Updating an AS Configuration

Auto Scaling (AS) enables elastic scaling of nodes in a node pool based on scaling policies. Without this function, you have to manually adjust the number of nodes in a node pool.

Notes and Constraints

To enable AS, the **CCE Cluster Autoscaler** add-on must be installed in the target cluster.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab, locate the row containing the target node pool and click **Auto Scaling**.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see **CCE Cluster Autoscaler**.
- If Autoscaler has been installed, directly configure scaling policies.

Step 3 Configure auto scaling policies.

AS Configuration

- **Customized Rule**: Click **Add Rule**. In the dialog box displayed, configure parameters. You can add multiple node scaling policies, a maximum of one CPU usage-based rule, and one memory usage-based rule. The total number of rules cannot exceed 10.

  The following table lists custom rules.

Table 4-13 Custom rules

Rule Type	Configuration
Metric-based	– **Trigger**: Select **CPU allocation rate** or **Memory allocation rate** and enter a value. The percentage must be greater than the value specified in the node resource requirements for a node scale-in when you configure a scaling policy (**Configuring an Auto Scaling Policy for a Cluster**). **NOTE** ▪ Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool ▪ **If multiple rules meet the conditions, the rules are executed in either of the following modes:** If rules based on the **CPU allocation rate** and **memory allocation rate** are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed. If a rule is configured based on the **CPU allocation rate** and **a periodic rule** and both the rules meet the scale-out conditions, the periodic rule executed early changes the node pool to the scaling state. As a result, the metric-based rule cannot be executed. After the periodic rule is executed and the node pool status becomes normal, the metric-based rule will not be executed. If the metric-based rule is executed early, the periodic rule will be executed after the metric-based rule is executed. ▪ If a rule is configured based on the **CPU allocation rate** and **memory allocation rate**, the policy detection period varies with the processing logic of each loop of the Autoscaler add-on. A scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cooldown period and node pool status. ▪ If the number of nodes reaches the upper limit of the cluster scale, **the upper limit of the nodes supported in a node pool**, or **the upper limit of the nodes of a specific flavor**, a metric-based scale-out will not be triggered. ▪ If the number of nodes, CPUs, or memory resources reaches **the upper limit for a node scale-out**, a metric-based scale-out will not be triggered. – **Action**: Configure an action to be performed when the triggering condition is met. ▪ **Custom**: Add a specified number of nodes to a node pool. ▪ **Auto calculation**: When the trigger condition is met, nodes are automatically added and the allocation rate is restored to a value lower than the threshold. The formula is as follows: Number of nodes to be added = [Resource request of pods in the node pool/(Available resources of a single node x Target allocation rate)] – Number of current nodes + 1

Rule Type	Configuration
Periodic	– **Trigger Time**: You can select a specific time every day, every week, every month, or every year. – **Action**: specifies an action to be carried out when the trigger time is reached. A specified number of nodes will be added to the node pool.

- **Nodes**: The number of nodes in a node pool will always be within the range during auto scaling.
- **Cooldown Period**: a period during which the nodes added in the current node pool cannot be scaled in.

AS Object

- **Specifications**: Configure whether to enable auto scaling for node flavors in a node pool.

  NOTE

  If multiple flavors are configured for a node pool, you can specify the upper limit for the number of nodes and the priority for each flavor separately.

Step 4 Click **OK**.

----End

4.5.3 Modifying Node Pool Configurations

Notes and Constraints

The default node pool does not support the management operations described in this section.

Configuration Management

CCE allows you to highly customize Kubernetes parameter settings on core components in a cluster. For more information, see **kubelet**.

This function is supported only in clusters of **v1.15 or later**. It is not displayed for versions earlier than v1.15.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Click **Manage** in the **Operation** column of the target node pool

Step 4 On the **Manage Configurations** page on the right, modify node pool parameter setting.

The following parameters can be configured for a node pool:

- **kubelet**

- **kube-proxy**
- **Docker (Available Only for Docker Node Pools)**
- **containerd (Available Only for containerd Node Pools)**
- **Networking Components (Available Only for CCE Turbo Clusters)**
- **Pod Security Groups (Available Only for CCE Turbo Clusters)**

Step 5 Click **OK**.

----End

kubelet

Item	Parameter	Description	Value	Modification
CPU management policy	cpu-manager-policy	CPU management policy configuration. For details, see **CPU Scheduling**. • **none**: disables pods from exclusively occupying CPUs. Select this value if you want a large pool of shareable CPU cores. • **static**: enables pods to exclusively occupy CPUs. Select this value if your workload is sensitive to latency in CPU cache and scheduling. • **enhanced-static**: allows burstable pods to preferentially use CPU cores. Select this value if your workload has huge peak-trough difference and is in the trough state most of the time.	Default: none	The CPU management policy does not apply to ECS (PM) node pools in CCE Turbo clusters.

Item	Parameter	Description	Value	Modification
QPS for requests to kube-apiserver	kube-api-qps	Number of queries per second for communication with the API server.	Default: 100	None
Burst for requests to kube-apiserver	kube-api-burst	Maximum number of burst requests sent to the API server per second.	Default: 100	None
Limit on the pods managed by kubelet	max-pods	Maximum number of pods that can run on a node.	• For a CCE standard cluster, the maximum number of pods is determined based on **the maximum number of pods on a node**. • For a CCE Turbo cluster, the maximum number of pods is determined based on **the number of NICs on a CCE Turbo cluster node**.	None
Limited number of processes in a pod	pod-pids-limit	Maximum number of PIDs that can be used in each pod.	Default: -1, which indicates that the number of PIDs is not limited	None

Item	Parameter	Description	Value	Modification
Whether to use a local IP address as a node's ClusterDNS	with-local-dns	The default ENI IP address of the node will be automatically added to the node's kubelet configuration as the preferred DNS address.	Default: false	None
QPS limit on creating events	event-qps	Number of events that can be generated per second.	Default: 5	None
Upper Limit for Burst Events	event-burst	Upper limit for burst event creation. The number of burst events can be temporarily increased to the specified value.	Default: 10	None
Allowed unsafe sysctls	allowed-unsafe-sysctls	Insecure system configuration allowed. Starting from **v1.17.17**, CCE enables pod security policies for kube-apiserver. Add corresponding configurations to **allowedUnsafeSysctls** of a pod security policy to make the policy take effect. (This configuration is not required for clusters earlier than v1.17.17.) For details, see **Example of Enabling Unsafe sysctls in a PSP**.	Default: []	None

Item	Parameter	Description	Value	Modification
Topology management policy	topology-manager-policy	Set the topology management policy. Valid values are as follows: • **restricted**: kubelet accepts only pods that achieve optimal NUMA alignment on the requested resources. • **best-effort**: kubelet preferentially selects pods that implement NUMA alignment on CPU and device resources. • **none** (default): The topology management policy is disabled. • **single-numa-node**: kubelet allows only pods that are aligned to the same NUMA node in terms of CPU and device resources.	Default: none	**NOTICE** Modifying **topology-manager-policy** and **topology-manager-scope** will restart kubelet, and the resource allocation of pods will be recalculated based on the modified policy. In this case, running pods may restart or even fail to receive any resources.
Topology management scope	topology-manager-scope	Configure the resource alignment granularity of the topology management policy. Valid values are as follows: • **container** (default) • **pod**	Default: container	

Item	Parameter	Description	Value	Modification
Specified DNS configuration file	resolv-conf	DNS resolution configuration file specified by the container	Default: null	None
Timeout for all runtime requests except long-running requests	runtime-request-timeout	Timeout interval of all runtime requests except long-running requests (pull, logs, exec, and attach).	Default: 2m0s	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Whether to allow kubelet to pull only one image at a time	serialize-image-pulls	Pull an image in serial mode. • **false**: recommended configuration so that an image can be pulled in parallel mode to improve pod startup. • **true**: allows images to be pulled in serial mode.	• Enabled by default if the cluster version is earlier than v1.21.12-r0, v1.23.11-r0, v1.27.3-r0, v1.28.1-r0 or v1.25.6-r0 • Disabled by default if the cluster version is v1.21.12-r0, v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, v1.28.1-r0, or later	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Image repository pull limit per second	registry-pull-qps	QPS upper limit of an image repository.	Default: 5 Value range: 1 to 50	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.

Item	Parameter	Description	Value	Modification
Upper limit of burst image pull	registry-burst	Maximum number of burst image pulls.	Default: 10 The value ranges from 1 to 100 and must be greater than or equal to the value of **registry-pull-qps**.	This parameter is available only in clusters of v1.21.10-r0, v1.23.8-r0, v1.25.3-r0, or later versions.
Maximum Number of Container Log Files	container-log-max-files	Maximum number of container log files. When the number of existing log files exceeds this value, the earliest log file will be deleted to release space for new log files.	Default: 10 Value range: 2 to 100	This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Maximum Container Log File Size	container-log-max-size	Maximum size of a single container log file. When the size of a log file reaches this value, the current log file will be closed and a new log file will be created to continue logging.	Default: 50 Value range: 1 to 4096	This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Upper Limit for Image Garbage Collection	image-gc-high-threshold	When the kubelet disk usage reaches this value, kubelet starts to collect image garbage.	Default: 80 Value range: 1 to 100	To disable image garbage collection, set this parameter to **100**. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Lower Limit for Image Garbage Collection	image-gc-low-threshold	Image garbage collection stops when the disk usage drops below the specified threshold.	Default: 70 Value range: 1 to 100	The value of this parameter cannot be greater than the upper limit for image garbage collection. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Node memory reservation	system-reserved-mem	System memory reservation reserves memory resources for OS system daemons such as sshd and udev.	Default value: automatically calculated, which varies depending on node flavors. For details, see **Node Resource Reservation Policy**.	The sum of **kube-reserved-mem** and **system-reserved-mem** must be less than 50% of the minimum memory of nodes in the node pool.
	kube-reserved-mem	Kubernetes memory reservation reserves memory resources for Kubernetes daemons such kubelet and container runtime.		

Item	Parameter	Description	Value	Modification
Hard eviction	memory.available	Available memory on a node.	The value is fixed at 100 MiB.	For details, see **Node-pressure Eviction**. **NOTICE** **Exercise caution** when modifying an eviction configuration item. Improper configuration may cause pods to be frequently evicted or fail to be evicted when the node is overloaded. kubelet can identify the following specific file system identifiers: • **nodefs**: main file system of a node. It is used for local disk volumes, emptyDir volumes that are not supported by memory, and log storage. For example, **nodefs** contains **/var/lib/kubelet/**. • **imagefs**: file system partition used by a container engine. For a **shared data disk**, kubelet and the container engine share the disk space. In this case, configure only the **nodefs** threshold or set the **imagefs** threshold to the same value as the **nodefs** threshold.
	nodefs.available	Percentage of the available capacity in the filesystem used by kubelet.	Default: 10% Value range: 1% to 99%	
	nodefs.inodesFree	Percentage of available inodes in the filesystem used by kubelet.	Default: 5% Value range: 1% to 99%	
	imagefs.available	Percentage of the available capacity in the filesystem used by container runtimes to store resources such as images.	Default: 10% Value range: 1% to 99%	
	imagefs.inodesFree	Percentage of available inodes in the filesystem used by container runtimes to store resources such as images.	This parameter is left blank by default. Value range: 1% to 99%	
	pid.available	Percentage of allocatable PIDs reserved for pods.	Default: 10% Value range: 1% to 99%	
Soft eviction	memory.available	Available memory on a node. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	This parameter is left blank by default. Value range: 100 to 1000000	
	nodefs.available	Percentage of the available capacity in	This parameter is	

Item	Parameter	Description	Value	Modification
		the filesystem used by kubelet. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	left blank by default. Value range: 1% to 99%	
	nodefs.inodesFree	Percentage of available inodes in the filesystem used by kubelet. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	This parameter is left blank by default. Value range: 1% to 99%	
	imagefs.available	Percentage of the available capacity in the filesystem used by container runtimes to store resources such as images. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	This parameter is left blank by default. Value range: 1% to 99%	

Item	Parameter	Description	Value	Modification
	imagefs.inodesFree	Percentage of available inodes in the filesystem used by container runtimes to store resources such as images. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	This parameter is left blank by default. Value range: 1% to 99%	
	pid.available	Percentage of allocatable PIDs reserved for pods. The value must be greater than the hard eviction value of the same parameter, and the eviction grace period (**evictionSoftGracePeriod**) must be configured accordingly.	This parameter is left blank by default. Value range: 1% to 99%	

kube-proxy

Item	Parameter	Description	Value	Modification
Maximum number of connection tracking entries	conntrack-min	Maximum number of connection tracking entries To obtain the value, run the following command: `sysctl net.nf_conntrack_max`	Default: 131072	None

Item	Parameter	Description	Value	Modification
Wait time of a closed TCP connection	conntrack-tcp-timeout-close-wait	Wait time of a closed TCP connection To obtain the value, run the following command: `sysctl net.netfilter.nf_conntrack_tcp_timeout_close_wait`	Default: 1h0m0s	None

Docker (Available Only for Docker Node Pools)

Item	Parameter	Description	Value	Modification
Container umask	native-umask	The default value **normal** indicates that the umask value of the started container is **0022**.	Default: normal	The parameter value cannot be changed.
Available data space for a single container	docker-base-size	Maximum data space that can be used by each container.	Default: 0	The parameter value cannot be changed.
Insecure image source address	insecure-registry	Whether an insecure image source address can be used.	false	The parameter value cannot be changed.
Maximum size of a container **core** file	limitcore	Maximum size of a core file in a container. The unit is byte. If not specified, the value is **infinity**.	Default: 5368709120	None

Item	Parameter	Description	Value	Modification
Limit on the number of handles in a container	default-ulimit-nofile	Maximum number of handles that can be used in a container.	Default: {soft}:{hard}	The value cannot exceed the value of the kernel parameter **nr_open** and cannot be a negative number. You can run the following command to obtain the kernel parameter **nr_open**: `sysctl -a | grep nr_open`
Image pull timeout	image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	Default: 1m0s	This parameter is supported in v1.25.3-r0 and later.
Maximum Number of Concurrent Requests for Downloading an Image at a Time	max-concurrent-downloads	This parameter specifies the maximum number of concurrent requests for downloading an image at a time.	Default: 3 Value range: 1 to 20	If this parameter is set to a large value, the network performance of other services on the node may be affected or the disk I/O and CPU usage may increase. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Maximum Container Log File Size	max-size	Maximum size of a container log file to be dumped. When the size of a log file reaches this value, the current log file will be closed and a new log file will be created to continue logging.	Default: 50 Value range: 1 to 4096	If this parameter is set to a small value, important logs may be lost. If this parameter is set to a large value, too much disk space may be occupied. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Maximum Number of Container Log Files	max-file	Maximum number of log files that can be retained in a container. When the number of existing log files exceeds this value, the earliest log file will be deleted to release space for new log files.	Default: 20 Value range: 2 to 100	If this parameter is set to a small value, important logs may be lost. If this parameter is set to a large value, too much disk space may be occupied. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Modify Image Repository Configuration	registry-mirrors	You can configure one or multiple substitute image repositories to be selected when obtaining images from the container runtime.	Default: [] Enter an IP address or domain name starting with http:// or https:// for a substitute image repository. For example, if you use the on-premises image repositories with IP addresses **http://example.com** and **https://example.com**, respectively, as the substitute repositories of the default repository, the parameter value is **["http://example.com,https://example.com"]**.	• To speed up image pulling, you can use an on-premises image repository as a substitute. • To improve fault tolerance and availability, you can configure multiple substitute image repositories. **NOTICE** If a substitute image repository is configured incorrectly, containers may not be able to pull the necessary image. This parameter is available only in clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later versions.

containerd (Available Only for containerd Node Pools)

Item	Parameter	Description	Value	Modification
Maximum size of a container **core** file	limitcore	Maximum size of a core file in a container. The unit is byte. If not specified, the value is **infinity**.	Default: 5368709120	None
Limit on the number of handles in a container	default-ulimit-nofile	Maximum number of handles that can be used in a container.	Default: 1048576	The value cannot exceed the value of the kernel parameter **nr_open** and cannot be a negative number. You can run the following command to obtain the kernel parameter **nr_open**: `sysctl -a | grep nr_open`
Image pull timeout	image-pull-progress-timeout	If the image fails to be pulled before time outs, the image pull will be canceled.	Default: 1m0s	This parameter is supported in v1.25.3-r0 and later.
Verification on insure skips	insecure_skip_verify	Whether to skip repository certificate verification.	Default: false	The parameter value cannot be changed.
Maximum Number of Concurrent Requests for Downloading an Image at a Time	max-concurrent-downloads	This parameter specifies the maximum number of concurrent requests for downloading an image at a time.	Default: 3 Value range: 1 to 20	If this parameter is set to a large value, the network performance of other services on the node may be affected or the disk I/O and CPU usage may increase. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Item	Parameter	Description	Value	Modification
Maximum Container Log Line Size	max-container-log-line-size	Maximum log line size of a container, in the unit of bytes. The log lines exceeding the limit will be split into multiple lines.	Default: 16384 Value range: 1 to 2097152	A larger value will lead to more containerd memory consumption. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
Available data space for a single container	container-base-size	Maximum data space that can be used by each container.	Default: 0	The parameter value cannot be changed.

Item	Parameter	Description	Value	Modification
Modify Image Repository Configuration	registry-mirrors	You can configure one or multiple substitute image repositories to be selected when obtaining images from the container runtime.	If you do not specify this parameter, the **docker.io** image repository will be used by default, and the SWR image repository will be used as a substitute. An image repository must be an IP address or domain name. A substitute image repository must be an IP address or domain name starting with **http://** or **https://**.	• Add local image repositories for faster image pulling. • Configure multiple image repositories for higher fault tolerance and availability. This parameter is available only in clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions. **NOTICE** If the image repository or its substitute is configured incorrectly, containers may not be able to pull the necessary image.

Item	Parameter	Description	Value	Modification
Certificate Authentication Skipped Image Repository	insecure-registries	When you specify image repositories, you can bypass security certificate-based authentication. This is typically done to connect to an insecure or self-signed image repository.	This parameter is left blank by default. Enter an IP address or domain name.	• Use this function only in development or test environments, not in production environments. • Enable this option only when using a self-signed certificate or when attempting to access a private image repository that cannot obtain an authorized certificate. This parameter is available only in clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

Networking Components (Available Only for CCE Turbo Clusters)

Item	Parameter	Description	Value	Modification
Node pool ENI pre-binding	enable-node-nic-configuration	Whether to enable ENI pre-binding in a node pool.	Default: false	After network component configuration is disabled in a node pool, the dynamic container ENI pre-binding parameter settings of the node pool are the same as those of cluster-level parameter settings.

Item	Parameter	Description	Value	Modification
ENI threshold	nic-threshold	Low threshold of the number of bound ENIs: High threshold of the number of bound ENIs	Default: 0:0	**NOTE** This parameter is being discarded. Use the dynamic pre-binding parameters of the other four ENIs.
Minimum number of ENIs bound to a node in a node pool	nic-minimum-target	Minimum number of container ENIs bound to a node. The parameter value must be a positive integer. The value **10** indicates that at least 10 container ENIs must be bound to a node. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.	Default: 10	Configure these parameters based on the number of pods typically running on most nodes.

Item	Parameter	Description	Value	Modification
Maximum number of ENIs pre-bound to a node in a node pool	nic-maximum-target	After the number of ENIs bound to a node exceeds the **nic-maximum-target** value, CCE will not proactively pre-bind ENIs. Checking the upper limit of pre-bound container ENIs is enabled only when the value of this parameter is greater than or equal to the minimum number of container ENIs (**nic-minimum-target**) bound to a node. The parameter value must be a positive integer. The value **0** indicates that checking the upper limit of pre-bound container ENIs is disabled. If the number you specified exceeds the container ENI quota of the node, the ENI quota will be used.	Default: 0	Configure these parameters based on the maximum number of pods running on most nodes.

Item	Parameter	Description	Value	Modification
Number of ENIs dynamically pre-bound to a node in a node pool	nic-warm-target	Extra ENIs will be pre-bound after the **nic-minimum-target** is used up in a pod. The value can only be a number. When the sum of the **nic-warm-target** value and the number of ENIs bound to the node is greater than the **nic-maximum-target** value, CCE will pre-bind the number of ENIs specified by the difference between the **nic-maximum-target** value and the current number of ENIs bound to the node.	Default: 2	Set the parameter value to the number of pods that can be scaled out instantaneously within 10 seconds on most nodes.

Item	Parameter	Description	Value	Modification
Threshold for reclaiming the ENIs pre-bound to a node in a node pool	nic-max-above-warm-target	Only when the difference between the number of idle ENIs on a node and the **nic-warm-target** value is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number. • A large value will accelerate pod startup but slow down the unbinding of idle container ENIs and decrease the IP address usage. **Exercise caution when performing this operation.** • A small value will speed up the unbinding of idle container ENIs and increase the IP address usage but will slow down pod startup, especially when a large number of pods increase instantaneously.	Default: 2	Set the parameter value to the difference between the number of pods that are frequently scaled on most nodes within minutes and the number of pods that are instantly scaled out on most nodes within 10 seconds.

Pod Security Groups (Available Only for CCE Turbo Clusters)

Item	Parameter	Description	Value	Modification
Default security group used by pods in a node pool	security_groups_for_nodepool	You can enter the security group ID. If this parameter is not configured, the default security group of the cluster container network will be used, and a maximum of five security group IDs that are separated by semicolons (;) can be specified at a time. The priority of the security group is lower than that of the security group configured for **SecurityGroups**.	None	None

4.5.4 Accepting Nodes in a Node Pool

If you want to add a newly purchased ECS to a node pool in a cluster, or remove a node from a node pool and add it to the node pool again, accept the node.

NOTICE

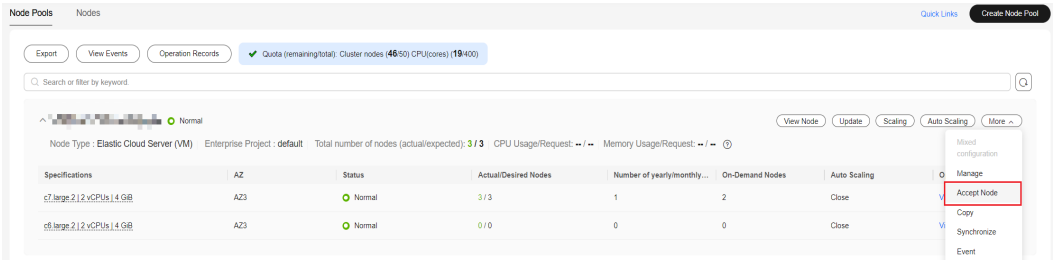
- When an ECS is accepted for management, the ECS OS will be reset to the standard image provided by CCE to ensure node stability.
- LVM information, including volume groups (VGs), logical volumes (LVs), and physical volumes (PVs), will be deleted from the system disks and data disks attached to the selected ECSs during acceptance. Ensure that the information has been backed up.
- During the acceptance of an ECS, do not perform any operation on the ECS through the ECS console.
- After a node is managed by a node pool, if a scaling-in task is triggered in the node pool, the node will be deleted.

Procedure

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **More** > **Accept Node** in the operation bar of the target node pool.



Step 4 Select one or more nodes that meet the requirements.

- The nodes and the current node pool are deployed in the same VPC and subnet.
- The nodes and the current node pool belong to the same enterprise project.
- The billing mode of the nodes must be the same as that of the current node pool. For example, a pay-per-use node pool can accept only pay-per-use nodes.
- The cloud server group of the nodes must be the same as that of the current node pool.
- The nodes must be running and cannot be labeled with **CCE-Dynamic-Provisioning-Node**.
- Data disks must be attached to the nodes to be managed if the system components of these nodes are separately stored. A local disk (disk-intensive disk) or a data disk of at least 20 GiB can be attached to the node, and any data disks already attached cannot be smaller than 10 GiB. For details about how to attach a data disk, see **Adding a Disk to an ECS**.
- The flavor of the nodes must be at least 2 vCPUs and 4 GiB memory, and only one NIC can be bound to each of the nodes.
- Only the cloud servers that have the same flavor, AZ, resource reservation, system disk, and data disk configurations as the node pool can be added for batch acceptance.
- Partitioned disks on cloud servers will not be accepted as data disks. Back up data and clear the disks before node acceptance.

Step 5 Click **OK**.

----End

4.5.5 Copying a Node Pool

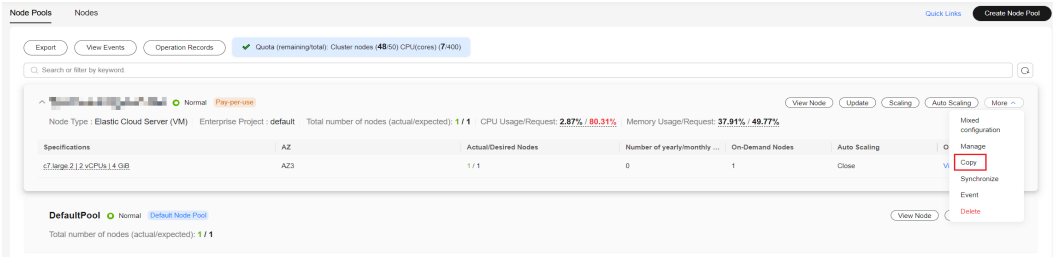
You can copy the configuration of an existing node pool on the CCE console to create new node pools.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **More** > **Copy** in the **Operation** column of the target node pool.

Figure 4-2 Copying a node pool



Step 4 In the **Copy Resource Pool** window, the configurations of the node pool to be copied are displayed. Modify the configurations as needed. For details, see **Creating a Node Pool**. After confirming the configuration, click **Next: Confirm**.

Step 5 On the **Confirm** page, confirm the node pool configurations and click **Submit**. Then, a new node pool is created based on the modified configurations.

----End

4.5.6 Synchronizing Node Pools

After the configuration of a node pool is updated, some configurations cannot be automatically synchronized for existing nodes. You can manually synchronize configurations for these nodes.

NOTICE

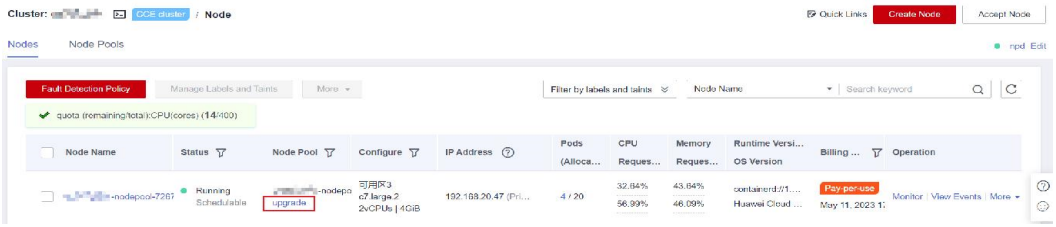
- Do not delete or reset nodes during batch synchronization. Otherwise, the synchronization of node pool configuration may fail.
- This operation involves resetting nodes. **Workloads running on a node may be interrupted due to standalone deployment or insufficient schedulable resources.** Evaluate the upgrade risks and perform the upgrade during off-peak hours. Alternatively, **specify a disruption budget for your key applications** to ensure the availability of these applications during the upgrade.
- During configuration synchronization for existing nodes, the nodes will be reset, and the system disks and data disks will be cleared. Back up important data before the synchronization.
- Only some node pool parameters can be synchronized by resetting nodes. The constraints are as follows:
  - Only clusters of v1.19 or later support the modification of the container engine, OS, system/data disk size, data disk space allocation, and pre-installation/post-installation script configuration.
  - Changes to the container engine, OS, or pre-/post-installation script in a node pool take effect only on new nodes. To synchronize the modification onto existing nodes, manually reset these nodes.
  - The modification of data disk space allocation and the system/data disk size of a node pool takes effect only for new nodes. The configuration cannot be synchronized even if the existing nodes are reset.
  - Changes to resource tags and Kubernetes labels/taints in a node pool will be automatically synchronized to existing nodes after the options of **Synchronization for Existing Nodes** are selected. You do not need to reset these nodes.

Synchronizing a Single Node

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Nodes** tab.

Step 3 Find **upgrade** in the **Node Pool** column of the existing nodes in the node pool.



Step 4 Click **update**. In the dialog box that is displayed, confirm whether to reset the node immediately.

----End

Batch Synchronization

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **More** > **Synchronize** in the **Operation** column of the target node pool.

Step 4 In the **Batch synchronization** window, configure parameters.

- **OS**: shows the image of the target version. You do not need to configure this parameter.
- **Synchronization Policy**: **Node Reset** is supported.
- **Max. Nodes for Batch Synchronize**: maximum number of nodes that will be unavailable during node synchronization. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **Node List**: Select the nodes requiring the synchronization of node pool configurations.

Step 5 Click **OK**.

----End

4.5.7 Upgrading an OS

After CCE releases a new OS image, if existing nodes cannot be automatically upgraded, you can manually upgrade them in batches.

Precautions

- This section describes how to upgrade an OS by resetting the target node. **Workloads running on a node may be interrupted due to standalone**

deployment or insufficient schedulable resources. Evaluate the upgrade risks and perform the upgrade during off-peak hours. Alternatively, **specify a disruption budget for your key applications** to ensure the availability of these applications during the upgrade.

- The system disk and data disks of the node will be cleared. **Back up important data** before resetting the node.
- Resetting a node will clear your configured Kubernetes labels and taints. As a result, resources (such as local storage and load of specified scheduling nodes) associated with the node may be unavailable. Exercise caution when performing this operation to avoid impact on running services.
- After the OS is upgraded, the node automatically starts.
- To ensure node stability, CCE will reserve some CPU and memory resources to run necessary system components.

Notes and Constraints

- Nodes running private images cannot be upgraded.
- Compatibility issues may occur when the node OS of an early version is upgraded. In this case, manually reset the node for the OS upgrade.

Procedure for Default Node Pools

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Click **Upgrade** next to the default node pool.

Step 4 In the displayed **Operating System Upgrade** window, configure parameters.

- **Target Operating System**: shows the image of the target version. You do not need to configure this parameter.
- **Upgrade Policy**: **Node Reset** is supported.
- **Max. Nodes for Batch Upgrade**: maximum number of nodes that will be unavailable during node upgrade. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **View Node**: Select the nodes to be upgraded.
- Login Mode:
  - **Password**

    The default username is **root**. Enter the password for logging in to the node and confirm the password.

    Be sure to remember the password as you will need it when you log in to the node.
  - **Key Pair**

    Select the key pair used to log in to the node. You can select a shared key.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create Key Pair**. For details about how to create a key pair, see **Creating a Key Pair**.

- **Inherit Password From Image** (supported for ECSs or PMs whose OSs are private images)

  Retain the password of the selected image. To use this option, ensure that you have set a password for the selected image.

- Pre-installation script:

  Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.

  The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may fail to be installed.

- Post-installation script:

  Installation script command, in which Chinese characters are not allowed. The script command will be Base64-transcoded.

  The script will be executed after Kubernetes software is installed, which does not affect the installation.

Step 5 Click **OK**.

----End

Procedure for Non-default Node Pools

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **More** > **Synchronize** in the **Operation** column of the target node pool.

Step 4 In the **Batch synchronization** window, configure parameters.

- **OS**: shows the image of the target version. You do not need to configure this parameter.
- **Synchronization Policy**: **Node Reset** is supported.
- **Max. Nodes for Batch Synchronize**: maximum number of nodes that will be unavailable during node synchronization. Nodes will be unavailable during synchronization by resetting the nodes. Properly configure this parameter to prevent pod scheduling failures caused by too many unavailable nodes in the cluster.
- **Node List**: Select the nodes requiring the synchronization of node pool configurations.

Step 5 Click **OK**.

----End

4.5.8 Migrating a Node

You can migrate nodes between node pools within a cluster. **Table 4-14** lists migration scenarios.

Table 4-14 Migration scenarios

Migration Scenario		Migration	Operation
Source Node Pool	**Target Node Pool**		
Custom node pool	Default node pool (DefaultPool)	Supported	**Migrating Nodes from a Custom Node Pool to the Default Node Pool**
Default node pool (DefaultPool)	Custom node pool	Supported by clusters of v1.23 or later	**Migrating Nodes from the Default Node Pool to a Custom Node Pool**
Custom node pool	Custom node pool	Not supported	None

Migrating Nodes from a Custom Node Pool to the Default Node Pool

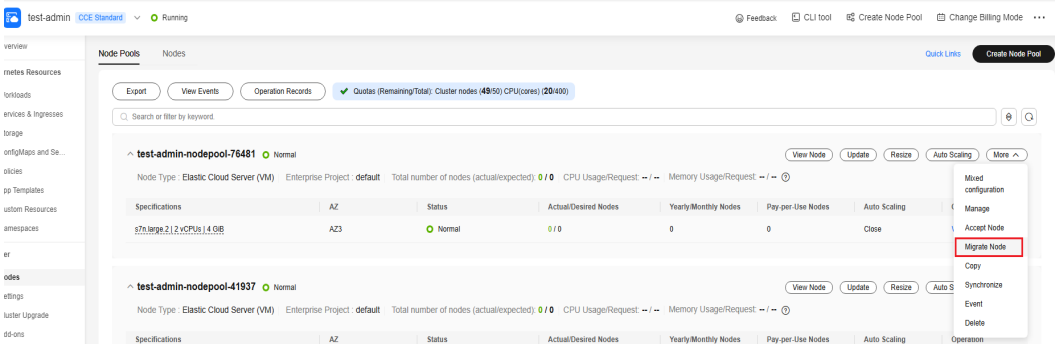
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes** and click the **Node Pools** tab.

Step 3 Click **View Node** in the **Operation** column of the node pool to be migrated.

Step 4 Choose **More** > **Migrate Node** in the **Operation** column of the target node to migrate the node.

Figure 4-3 Migrating nodes to the default node pool



Step 5 In the displayed **Migrate Node** dialog box, confirm the information.

NOTE

- The migration does not affect custom resource tags, and Kubernetes labels and taints of the node.
- After the migration, system labels **cce.cloud.com** and **cce-nodepool** on the node will be deleted. If an existing workload uses these labels for affinity or anti-affinity scheduling, the existing pods on the node will be stopped and rescheduled when kubelet is restarted.

----End

Migrating Nodes from the Default Node Pool to a Custom Node Pool

NOTICE

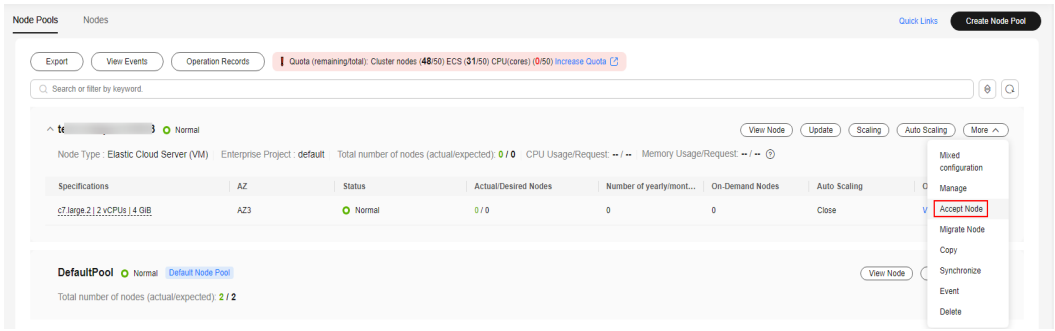
Nodes cannot be migrated to spot pricing node pools.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes** and click the **Node Pools** tab.

Step 3 Locate the target node pool and choose **More** > **Accept Node**.

Figure 4-4 Migrating nodes to a custom node pool



Step 4 In the **Accept Node** dialog box, select the nodes that meet the following conditions:

- The nodes and the current node pool are deployed in the same VPC and subnet.
- The nodes and the current node pool belong to the same enterprise project.
- The nodes and the current node pool are in the same cloud server group.
- The billing mode of the nodes is supported by the current node pool.
- The nodes are running and they are from the default node pool.
- The flavor, AZ, resource reservation, container engine, and OS configurations of the nodes are the same as those of the node pool.

Step 5 Click **OK**.

NOTE

- After nodes are migrated into a node pool, the pool's resource tags, Kubernetes labels, and Kubernetes taints will be synchronized to the nodes. If there is a conflict, the node pool's configurations will take precedence.
- After the migration, the pool's security group will take over the nodes' original security group.
- After the migration, the pool's agency will take over the nodes' original agency.
- After the migration, the nodes' original login mode will remain unchanged.
- After a node is migrated to a node pool after being accepted by the cluster, its acceptance tag will be removed. Scaling in the node pool may result in the removal of the node.

----End

4.5.9 Deleting a Node Pool

Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools.

Notes and Constraints

- Before deleting a yearly/monthly-billed node pool, remove all nodes in it.
- Deleting a node will cause PVC/PV data loss for the **local PVs** associated with the node. These PVCs and PVs cannot be restored or used again. In this scenario, the pod that uses the local PV is evicted from the node. A new pod is created and stays in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled.

Precautions

- Deleting a node pool will delete all nodes in the node pool. Back up data in a timely manner to prevent data loss.
- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours. If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.
- When deleting a node pool, the system sets all nodes in the current node pool to the unschedulable state.

Procedure

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Choose **More > Delete** in the **Operation** column of the target node pool.

Step 4 Read the precautions in the **Delete Node Pool** dialog box.

Step 5 In the text box, enter **DELETE** and click **Yes** to confirm that you want to continue the deletion.

----End

5 Workloads

5.1 Overview

A workload is an application running on Kubernetes. No matter how many components are there in your workload, you can run it in a group of Kubernetes pods. A workload is an abstract model of a group of pods in Kubernetes. Workloads in Kubernetes are classified as Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

CCE provides Kubernetes-native container deployment and management and supports lifecycle management of container workloads, including creation, configuration, monitoring, auto scaling, upgrade, uninstall, service discovery, and load balancing.

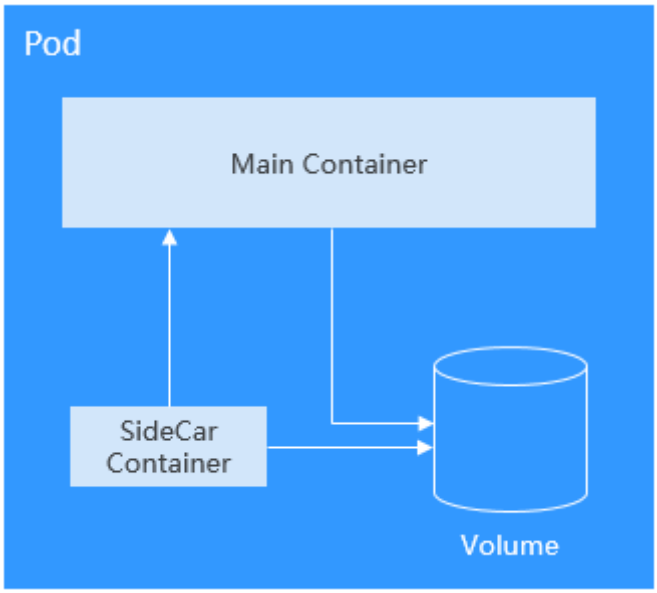
Overview of Pod

A pod is the smallest, simplest unit in the Kubernetes object model that you create or deploy. A pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. Each pod has a separate IP address.

Pods can be used in either of the following ways:

- A pod runs only one container. This is the most common usage of pods in Kubernetes. You can consider a pod as a container, but Kubernetes directly manages pods instead of containers.
- A pod runs multiple containers that need to be tightly coupled. In this scenario, a pod contains a main container and several sidecar containers, as shown in **Figure 5-1**. For example, the main container is a web server that provides file services from a fixed directory, and sidecar containers periodically download files to this fixed directory.

Figure 5-1 Pod running multiple containers

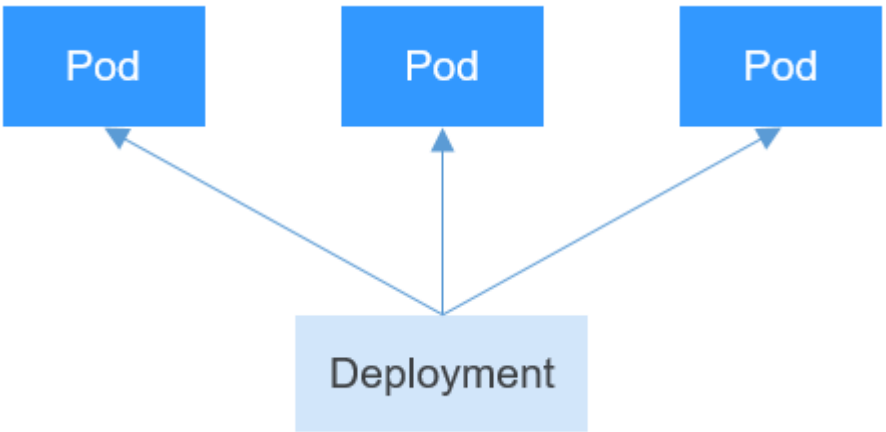


In Kubernetes, pods are rarely created directly. Instead, Kubernetes controller manages pods through pod instances such as Deployments and jobs. A controller typically uses a pod template to create pods. The controller can also manage multiple pods and provide functions such as replica management, rolling upgrade, and self-healing.

Overview of Deployment

A pod is the smallest and simplest unit that you create or deploy in Kubernetes. It is designed to be an ephemeral, one-off entity. A pod can be evicted when node resources are insufficient and disappears along with a cluster node failure. Kubernetes provides controllers to manage pods. Controllers can create and manage pods, and provide replica management, rolling upgrade, and self-healing capabilities. The most commonly used controller is Deployment.

Figure 5-2 Relationship between a Deployment and pods



A Deployment can contain one or more pods. These pods have the same role. Therefore, the system automatically distributes requests to multiple pods of a Deployment.

A Deployment integrates a lot of functions, including online deployment, rolling upgrade, replica creation, and restoration of online jobs. To some extent, Deployments can be used to realize unattended rollout, which greatly reduces difficulties and operation risks in the rollout process.

Overview of StatefulSet

All pods under a Deployment have the same characteristics except for the name and IP address. If required, a Deployment can use a pod template to create new pods. If not required, the Deployment can delete any one of the pods.

However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

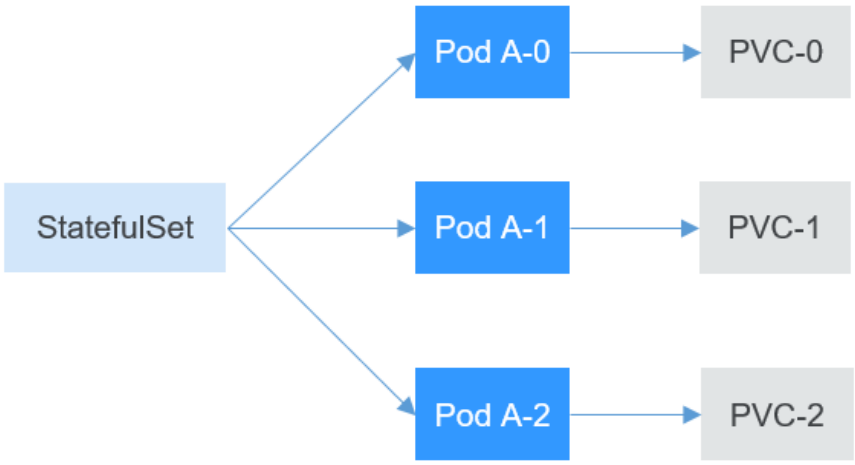
Distributed stateful applications involve different roles for different responsibilities. For example, databases work in active/standby mode, and pods depend on each other. To deploy stateful applications in Kubernetes, ensure pods meet the following requirements:

- Each pod must have a fixed identifier so that it can be recognized by other pods.
- Separate storage resources must be configured for each pod. In this way, the original data can be retrieved after a pod is deleted and restored. Otherwise, the pod status will be changed after the pod is rebuilt.

To address the preceding requirements, Kubernetes provides StatefulSets.

1. StatefulSets provide a fixed name for each pod following a fixed number ranging from 0 to N. After a pod is rescheduled, the pod name and the hostname remain unchanged.
2. StatefulSets use a headless Service to allocate a fixed domain name for each pod.
3. StatefulSets create PVCs with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.

Figure 5-3 StatefulSet

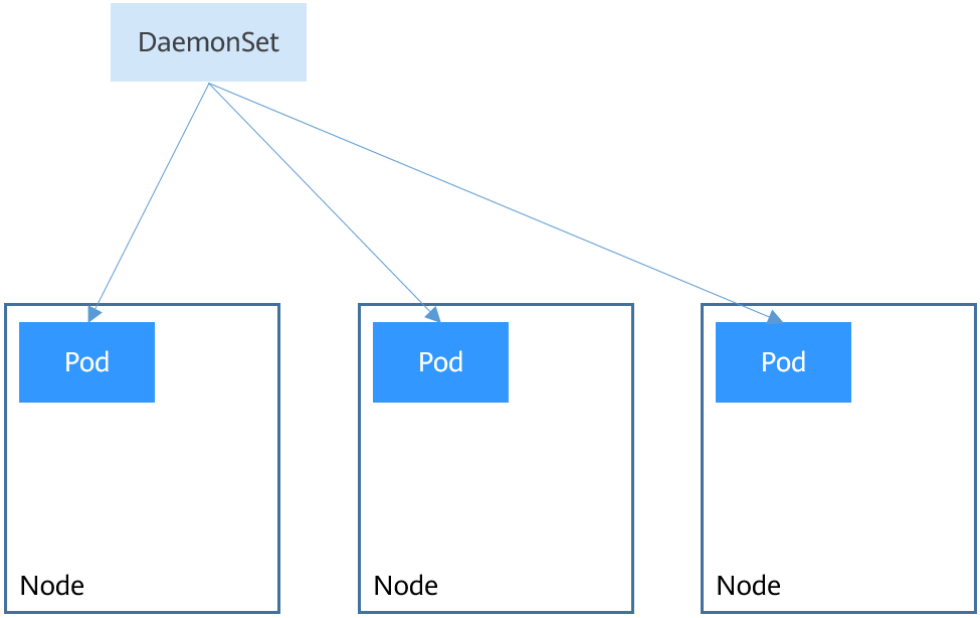


Overview of DaemonSet

A DaemonSet runs a pod on each node in a cluster and ensures that there is only one pod. This works well for certain system-level applications such as log collection and resource monitoring since they must run on each node and need only a few pods. A good example is kube-proxy.

DaemonSets are closely related to nodes. If a node becomes faulty, the DaemonSet will not create the same pods on other nodes.

Figure 5-4 DaemonSet



Overview of Job and CronJob

Jobs and CronJobs allow you to run short lived, one-off tasks in batch. They ensure the task pods run to completion.

- A job is a resource object used by Kubernetes to control batch tasks. Jobs are different from long-term servo tasks (such as Deployments and StatefulSets). The former is started and terminated at specific times, while the latter runs unceasingly unless being terminated. The pods managed by a job will be automatically removed after successfully completing tasks based on user configurations.
- A CronJob runs a job periodically on a specified schedule. A CronJob object is similar to a line of a crontab file in Linux.

This run-to-completion feature of jobs is especially suitable for one-off tasks, such as continuous integration (CI).

Workload Lifecycle

Table 5-1 Status description

Status	Description
Running	All pods are running or the number of pods is 0.
Unready	The container malfunctions and the pod under the workload is not working.
Processing	The workload is not running but no error is reported.
Available	For a multi-pod Deployment, some pods are abnormal but at least one pod is available.
Completed	The task is successfully executed. This status is available only for common tasks.
Stopped	The workload is stopped and the number of pods changes to 0. This status is available for workloads earlier than v1.13.
Deleting	The workload is being deleted.

5.2 Creating a Workload

5.2.1 Creating a Deployment

Scenario

Deployments are workloads (for example, Nginx) that do not store any data or status. You can create Deployments on the CCE console or by running kubectl commands.

Prerequisites

- Before creating a workload, you must have an available cluster. For details about how to create a cluster, see **Buying a CCE Standard/Turbo Cluster**.
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

  NOTE

  If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the Deployment will fail.

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Configure the workload.

Basic Info

- **Workload Type**: Select **Deployment**. For details about workload types, see **Overview**.
- **Workload Name**: Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses a common runtime by default, whereas a CCE Turbo cluster supports both common and secure runtimes. For details about the differences, see **Secure Runtime and Common Runtime**.
- **Time Zone Synchronization**: Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see **Configuring Time Zone Synchronization**.

Container Settings

- Container Information

  Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

  – **Basic Info**: Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.

Parameter	Description
Image Name	Click **Select Image** and select the image used by the container. ■ **SWR Shared Edition**: easy-to-use, secure, and reliable image management ○ **My Images**: You can select a private image that you have uploaded. For details about how to upload images, see **Uploading an Image Through a Container Engine Client**. ○ **Open Source Images**: Select a public image provided by SWR. For details, see **Image Center**. ○ **Shared Images**: You can select an image shared by another account. For details, see **Sharing a Private Image**. To use a third-party image, directly enter image path. Ensure that the **image access credential** can be used to access the image repository. For details, see **Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.

Parameter	Description
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **CCE AI Suite (NVIDIA GPU)** add-on has been installed. ■ **Do not use**: No GPU will be used. ■ **GPU card**: The GPU is dedicated for the container. ■ **GPU Virtualization**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container will use 10% of GPU resources. For details about how to use GPUs in the cluster, see **Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required NPU chips. The value must be an integer and the **CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **NPU Scheduling**.
(Optional) Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see **Init Containers**.

- (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see **Configuring Container Lifecycle Parameters**.
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **Configuring Container Health Check**.
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see **Configuring Environment Variables**.

  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see **Storage**.

    NOTE

    If the workload contains more than one pod, EVS volumes cannot be mounted.

  - (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
  - (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.

    To disable the standard output of the current workload, add the annotation **kubernetes.AOM.log.stdout: []** in **Labels and Annotations**. For details about how to use this annotation, see **Table 5-14**.

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to access images in SWR Shared Edition. For details about **default-secret**, see **default-secret**.
- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see **Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **Configuring Workload Upgrade Policies**.
- **Scheduling**: Configure affinity and anti-affinity policies for flexible workload scheduling. Load affinity and node affinity are provided.
  - **Load Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No load affinity policy is configured.
    - **Multi-AZ deployment preferred**: Workload pods are **preferentially** scheduled to nodes in different AZs through pod anti-affinity.
    - **Forcible multi-AZ deployment**: Workload pods are forcibly scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If there are fewer AZs than pods, the extra pods will fail to run.

        - **Customize affinity**: Affinity and anti-affinity policies can be customized. For details, see **Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**.
    - **Node Affinity**: Common node affinity policies are offered for quick load affinity deployment.
        - **Not configured**: No node affinity policy is configured.
        - **Specify node**: Workload pods can be deployed on specified nodes through node affinity (**nodeAffinity**). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
        - **Specify node pool**: Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
        - **Customize affinity**: Affinity and anti-affinity policies can be customized. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.
- **Toleration**: Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Configuring Tolerance Policies**.
- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see **Configuring Labels and Annotations**.
- **DNS**: Configure a separate DNS policy for the workload. For details, see **DNS Configuration**.
- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **Configuring APM**.
- **Network Configuration**
    - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **Configuring QoS for a Pod**.
    - Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see **Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration**.
    - Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

The following procedure uses Nginx as an example to describe how to create a workload using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name, and you can rename it as required.

vi nginx-deployment.yaml

The following is an example YAML file. For more information about Deployments, see **Kubernetes documentation**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx    # If you use an image from an open-source image registry, enter the image name. If you use an image in My Images, obtain the image path from SWR.
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

For details about the parameters, see **Table 5-2**.

Table 5-2 Deployment YAML parameters

Parameter	Description	Mandatory/ Optional
apiVersion	API version. **NOTE** Set this parameter based on the cluster version. • For clusters of v1.17 or later, the apiVersion format of Deployments is **apps/v1**. • For clusters of v1.15 or earlier, the apiVersion format of Deployments is **extensions/v1beta1**.	Mandatory
kind	Type of a created object.	Mandatory
metadata	Metadata of a resource object.	Mandatory

Parameter	Description	Mandatory/ Optional
name	Name of the Deployment.	Mandatory
spec	Detailed description of the Deployment.	Mandatory
replicas	Number of pods.	Mandatory
selector	Determines container pods that can be managed by the Deployment.	Mandatory
strategy	Upgrade mode. Possible values: • RollingUpdate • ReplaceUpdate By default, rolling update is used.	Optional
template	Detailed description of a created container pod.	Mandatory
metadata	Metadata.	Mandatory
labels	**metadata.labels**: Container labels.	Optional
spec: containers	• **image** (mandatory): Name of a container image. • **imagePullPolicy** (optional): Policy for obtaining an image. The options include **Always** (attempting to download images each time), **Never** (only using local images), and **IfNotPresent** (using local images if they are available; downloading images if local images are unavailable). The default value is **Always**. • **name** (mandatory): Container name.	Mandatory
imagePull Secrets	Name of the secret used during image pulling. If a private image is used, this parameter is mandatory. • To pull an image from the Software Repository for Container (SWR), set this parameter to **default-secret**. • To pull an image from a third-party image repository, set this parameter to the name of the created secret.	Optional

Step 3 Create a Deployment.

kubectl create -f nginx-deployment.yaml

If the following information is displayed, the Deployment is being created.


```
deployment.apps/nginx created
```

Step 4 Obtain the Deployment status.

kubectl get deployment

If the following information is displayed, the Deployment is running.

```
NAME         READY    UP-TO-DATE   AVAILABLE   AGE
nginx        1/1      1            1           4m5s
```

Parameter description

- **NAME**: Name of the application running in the pod.
- **READY**: indicates the number of available workloads. The value is displayed as "the number of available pods/the number of expected pods".
- **UP-TO-DATE**: indicates the number of replicas that have been updated.
- **AVAILABLE**: indicates the number of available pods.
- **AGE**: period the Deployment keeps running

Step 5 If the Deployment will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see **Network**.

----End

Documentation

- **Upgrading Pods Without Interrupting Services**
- **Configuring Domain Name Resolution for CCE Containers**

5.2.2 Creating a StatefulSet

Scenario

StatefulSets are a type of workloads whose data or status is stored while they are running. For example, MySQL is a StatefulSet because it needs to store new data.

A container can be migrated between different hosts, but data is not stored on the hosts. To store StatefulSet data persistently, attach HA storage volumes provided by CCE to the container.

Notes and Constraints

- When you delete or scale a StatefulSet, the system does not delete the storage volumes associated with the StatefulSet to ensure data security.
- When you delete a StatefulSet, reduce the number of replicas to **0** before deleting the StatefulSet so that pods in the StatefulSet can be stopped in order.
- When you create a StatefulSet, a headless Service is required for pod access. For details, see **Headless Services**.
- When a node is unavailable, pods become **Unready**. In this case, manually delete the pods of the StatefulSet so that the pods can be migrated to a normal node.

Prerequisites

- Before creating a workload, you must have an available cluster. For details on how to create a cluster, see **Buying a CCE Standard/Turbo Cluster**.
- To enable public access to a workload, ensure that an EIP or load balancer has been bound to at least one node in the cluster.

  NOTE

  If a pod has multiple containers, ensure that the ports used by the containers do not conflict with each other. Otherwise, creating the StatefulSet will fail.

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **StatefulSet**. For details about workload types, see **Overview**.
- **Workload Name**: Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses a common runtime by default, whereas a CCE Turbo cluster supports both common and secure runtimes. For details about the differences, see **Secure Runtime and Common Runtime**.
- **Time Zone Synchronization**: Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see **Configuring Time Zone Synchronization**.

Container Settings

- Container Information

  Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

  – **Basic Info**: Configure basic information about the container.

Parameter	Description
Container Name	Name the container.

Parameter	Description
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click **Select Image** and select the image used by the container. ■ **SWR Shared Edition**: easy-to-use, secure, and reliable image management ○ **My Images**: You can select a private image that you have uploaded. For details about how to upload images, see **Uploading an Image Through a Container Engine Client**. ○ **Open Source Images**: Select a public image provided by SWR. For details, see **Image Center**. ○ **Shared Images**: You can select an image shared by another account. For details, see **Sharing a Private Image**. To use a third-party image, directly enter image path. Ensure that the **image access credential** can be used to access the image repository. For details, see **Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.

Parameter	Description
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **CCE AI Suite (NVIDIA GPU)** add-on has been installed. ■ **Do not use**: No GPU will be used. ■ **GPU card**: The GPU is dedicated for the container. ■ **GPU Virtualization**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container will use 10% of GPU resources. For details about how to use GPUs in the cluster, see **Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required NPU chips. The value must be an integer and the **CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **NPU Scheduling**.
(Optional) Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see **Init Containers**.

- (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see **Configuring Container Lifecycle Parameters**.
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **Configuring Container Health Check**.
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see **Configuring Environment Variables**.

  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see **Storage**.

    NOTE

    - StatefulSets support dynamic attachment of EVS disks. For details, see **Dynamically Mounting an EVS Disk to a StatefulSet** or **Dynamically Mounting a Local PV to a StatefulSet**.

      Dynamic mounting is achieved by using the **volumeClaimTemplates** field and depends on the dynamic creation capability of StorageClass. A StatefulSet associates each pod with a PVC using the **volumeClaimTemplates** field, and the PVC is bound to the corresponding PV. Therefore, after the pod is rescheduled, the original data can still be mounted based on the PVC name.

    - After a workload is created, the storage that is dynamically mounted cannot be updated.

  - (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
  - (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.

    To disable the standard output of the current workload, add the annotation **kubernetes.AOM.log.stdout: []** in **Labels and Annotations**. For details about how to use this annotation, see **Table 5-14**.

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to access images in SWR Shared Edition. For details about **default-secret**, see **default-secret**.
- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

Headless Service Parameters

A headless Service is used to solve the problem of mutual access between pods in a StatefulSet. The headless Service provides a fixed access domain name for each pod. For details, see **Headless Services**.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see **Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **Configuring Workload Upgrade Policies**.
- **Pod Management Policies**

For some distributed systems, the StatefulSet sequence is unnecessary and/or should not occur. These systems require only uniqueness and identifiers.

- **OrderedReady**: The StatefulSet will deploy, delete, or scale pods in order and one by one. (The StatefulSet continues only after the previous pod is ready or deleted.) This is the default policy.
- **Parallel**: The StatefulSet will create pods in parallel to match the desired scale without waiting, and will delete all pods at once.

- **Scheduling**: Configure affinity and anti-affinity policies for flexible workload scheduling. Load affinity and node affinity are provided.
  - **Load Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No load affinity policy is configured.
    - **Multi-AZ deployment preferred**: Workload pods are **preferentially** scheduled to nodes in different AZs through pod anti-affinity.
    - **Forcible multi-AZ deployment**: Workload pods are forcibly scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If there are fewer AZs than pods, the extra pods will fail to run.
    - **Customize affinity**: Affinity and anti-affinity policies can be customized. For details, see **Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**.
  - **Node Affinity**: Common node affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No node affinity policy is configured.
    - **Specify node**: Workload pods can be deployed on specified nodes through node affinity (**nodeAffinity**). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Specify node pool**: Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Customize affinity**: Affinity and anti-affinity policies can be customized. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.
- **Toleration**: Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Configuring Tolerance Policies**.
- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see **Configuring Labels and Annotations**.
- **DNS**: Configure a separate DNS policy for the workload. For details, see **DNS Configuration**.

- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **Configuring APM**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **Configuring QoS for a Pod**.
  - Whether to enable the static IP address: available only for clusters that support this function. After this function is enabled, you can set the interval for reclaiming expired pod IP addresses. For details, see **Configuring a Static IP Address for a Pod**.
  - Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see **Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration**.
  - Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

In this example, a Nginx workload is used and the EVS volume is dynamically mounted to it using the **volumeClaimTemplates** field.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-statefulset.yaml** file.

nginx-statefulset.yaml is an example file name, and you can change it as required.

vi nginx-statefulset.yaml

The following provides an example of the file contents. For more information on StatefulSet, see the **Kubernetes documentation**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
```



```
        image: nginx:latest
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
          limits:
            cpu: 250m
            memory: 512Mi
        volumeMounts:
          - name: test
            readOnly: false
            mountPath: /usr/share/nginx/html
            subPath: ''
    imagePullSecrets:
      - name: default-secret
    dnsPolicy: ClusterFirst
    volumes: []
  serviceName: nginx-svc
  replicas: 2
  volumeClaimTemplates:  # Dynamically mounts the EVS volume to the workload.
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: test
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS  # SAS EVS volume type.
        labels:
          failure-domain.beta.kubernetes.io/region: ap-southeast-1  # region where the EVS volume is created.
          failure-domain.beta.kubernetes.io/zone:    # AZ where the EVS volume is created. It must be the
same as the AZ of the node.
      spec:
        accessModes:
          - ReadWriteOnce  # The value must be ReadWriteOnce for the EVS volume.
        resources:
          requests:
            storage: 10Gi
        storageClassName: csi-disk # StorageClass name. The value is csi-disk for the EVS volume.
  updateStrategy:
    type: RollingUpdate
```

vi nginx-headless.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
    version: v1
  clusterIP: None
  ports:
    - name: nginx
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Step 3 Create a workload and the corresponding headless service.

kubectl create -f nginx-statefulset.yaml

If the following information is displayed, the StatefulSet has been successfully created.


```
statefulset.apps/nginx created
```

kubectl create -f nginx-headless.yaml

If the following information is displayed, the headless service has been successfully created.

```
service/nginx-svc created
```

Step 4 If the workload will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see **Network**.

----End

5.2.3 Creating a DaemonSet

Scenario

CCE provides deployment and management capabilities for multiple types of containers and supports features of container workloads, including creation, configuration, monitoring, scaling, upgrade, uninstall, service discovery, and load balancing.

DaemonSet ensures that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. If a DaemonSet is deleted, all pods created by it will be deleted.

The typical application scenarios of a DaemonSet are as follows:

- Run the cluster storage daemon, such as glusterd or Ceph, on each node.
- Run the log collection daemon, such as Fluentd or Logstash, on each node.
- Run the monitoring daemon, such as Prometheus Node Exporter, collectd, Datadog agent, New Relic agent, or Ganglia (gmond), on each node.

You can deploy a DaemonSet for each type of daemons on all nodes, or deploy multiple DaemonSets for the same type of daemons. In the second case, DaemonSets have different flags and different requirements on memory and CPU for different hardware types.

Prerequisites

Before creating a DaemonSet, you must have an available cluster. For details on how to create a cluster, see **Buying a CCE Standard/Turbo Cluster**.

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **DaemonSet**. For details about workload types, see **Overview**.

- **Workload Name**: Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Container Runtime**: A CCE standard cluster uses a common runtime by default, whereas a CCE Turbo cluster supports both common and secure runtimes. For details about the differences, see **Secure Runtime and Common Runtime**.
- **Time Zone Synchronization**: Specify whether to enable time zone synchronization. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone. For details, see **Configuring Time Zone Synchronization**.

Container Settings

- Container Information

  Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

  – **Basic Info**: Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.
Image Name	Click **Select Image** and select the image used by the container. ▪ **SWR Shared Edition**: easy-to-use, secure, and reliable image management ○ **My Images**: You can select a private image that you have uploaded. For details about how to upload images, see **Uploading an Image Through a Container Engine Client**. ○ **Open Source Images**: Select a public image provided by SWR. For details, see **Image Center**. ○ **Shared Images**: You can select an image shared by another account. For details, see **Sharing a Private Image**. To use a third-party image, directly enter image path. Ensure that the **image access credential** can be used to access the image repository. For details, see **Using Third-Party Images**.

Parameter	Description
Image Tag	Select the image tag to be deployed.
CPU Quota	▪ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ **Limit**: maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
Memory Quota	▪ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ▪ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **CCE AI Suite (NVIDIA GPU)** add-on has been installed. ▪ **Do not use**: No GPU will be used. ▪ **GPU card**: The GPU is dedicated for the container. ▪ **GPU Virtualization**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container will use 10% of GPU resources. For details about how to use GPUs in the cluster, see **Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required NPU chips. The value must be an integer and the **CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **NPU Scheduling**.

Parameter	Description
(Optional) Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see **Init Containers**.

  - (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see **Configuring Container Lifecycle Parameters**.
  - (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **Configuring Container Health Check**.
  - (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see **Configuring Environment Variables**.
  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see **Storage**.
  - (Optional) **Security Context**: Assign container permissions to protect the system and other containers from being affected. Enter the user ID to assign container permissions and prevent systems and other containers from being affected.
  - (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.

    To disable the standard output of the current workload, add the annotation **kubernetes.AOM.log.stdout: []** in **Labels and Annotations**. For details about how to use this annotation, see **Table 5-14**.

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to access images in SWR Shared Edition. For details about **default-secret**, see **default-secret**.

- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards access traffic to pods and automatically balances load for these pods.

You can also create a Service after creating a workload. For details about Services of different types, see **Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **Configuring Workload Upgrade Policies**.
- **Scheduling**: Configure affinity and anti-affinity policies for flexible workload scheduling. Node affinity is provided.
  - **Node Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Not configured**: No node affinity policy is configured.
    - **Specified node scheduling**: Workload pods can be deployed on specified nodes through node affinity (**nodeAffinity**). If no node is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Specified node pool scheduling**: Workload pods can be deployed in a specified node pool through node affinity (**nodeAffinity**). If no node pool is specified, the pods will be randomly scheduled based on the default scheduling policy of the cluster.
    - **Custom policies**: Affinity and anti-affinity policies can be customized. For details, see **Configuring Node Affinity Scheduling (nodeAffinity)**.
- **Toleration**: Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Configuring Tolerance Policies**.
- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see **Configuring Labels and Annotations**.
- **DNS**: Configure a separate DNS policy for the workload. For details, see **DNS Configuration**.
- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **Configuring APM**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **Configuring QoS for a Pod**.

- Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see **Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration**.
- Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

The following procedure uses Nginx as an example to describe how to create a workload using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-daemonset.yaml** file. **nginx-daemonset.yaml** is an example file name, and you can change it as required.

vi nginx-daemonset.yaml

The content of the description file is as follows: The following provides an example. For more information on DaemonSets, see **Kubernetes documentation**.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx-daemonset
  labels:
    app: nginx-daemonset
spec:
  selector:
    matchLabels:
      app: nginx-daemonset
  template:
    metadata:
      labels:
        app: nginx-daemonset
    spec:
      nodeSelector:              # Node selection. A pod is created on a node only when the node meets daemon=need.
        daemon: need
      containers:
      - name: nginx-daemonset
        image: nginx:alpine
        resources:
          limits:
            cpu: 250m
            memory: 512Mi
          requests:
            cpu: 250m
            memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```


The **replicas** parameter used in defining a Deployment or StatefulSet does not exist in the above configuration for a DaemonSet, because each node has only one replica. It is fixed.

The nodeSelector in the preceding pod template specifies that a pod is created only on the nodes that meet **daemon=need**. If you want to create a pod on each node, delete the label.

Step 3 Create a DaemonSet.

kubectl create -f nginx-daemonset.yaml

If the following information is displayed, the DaemonSet is being created.

```
daemonset.apps/nginx-daemonset created
```

Step 4 Obtain the DaemonSet status.

kubectl get ds

```
$ kubectl get ds
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
nginx-daemonset   1         1         0       1            0           daemon=need     116s
```

Step 5 If the workload will be accessed through a ClusterIP or NodePort Service, configure the access mode. For details, see **Network**.

----End

5.2.4 Creating a Job

Scenario

Jobs are short-lived and run for a certain time to completion. They can be executed immediately after being deployed. It is completed after it exits normally (exit 0).

A job is a resource object that is used to control batch tasks. It is different from a long-term servo workload (such as Deployment and StatefulSet).

A job is started and terminated at specific times, while a long-term servo workload runs unceasingly unless being terminated. The pods managed by a job automatically exit after successfully completing the job based on user configurations. The success flag varies according to the spec.completions policy.

- One-off jobs: A single pod runs once until successful termination.
- Jobs with a fixed success count: N pods run until successful termination.
- A queue job is considered completed based on the global success confirmed by the application.

Prerequisites

Resources have been created. For details, see **Creating a Node**. If clusters and nodes are available, you need not create them again.

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **Job**. For details about workload types, see **Overview**.
- **Workload Name**: Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses a common runtime by default, whereas a CCE Turbo cluster supports both common and secure runtimes. For details about the differences, see **Secure Runtime and Common Runtime**.

Container Settings

- Container Information

  Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

  - **Basic Info**: Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.

Parameter	Description
Image Name	Click **Select Image** and select the image used by the container. ■ **SWR Shared Edition**: easy-to-use, secure, and reliable image management ○ **My Images**: You can select a private image that you have uploaded. For details about how to upload images, see **Uploading an Image Through a Container Engine Client**. ○ **Open Source Images**: Select a public image provided by SWR. For details, see **Image Center**. ○ **Shared Images**: You can select an image shared by another account. For details, see **Sharing a Private Image**. To use a third-party image, directly enter image path. Ensure that the **image access credential** can be used to access the image repository. For details, see **Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.

Parameter	Description
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **CCE AI Suite (NVIDIA GPU)** add-on has been installed. ■ **Do not use**: No GPU will be used. ■ **GPU card**: The GPU is dedicated for the container. ■ **GPU Virtualization**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container will use 10% of GPU resources. For details about how to use GPUs in the cluster, see **Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required NPU chips. The value must be an integer and the **CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **NPU Scheduling**.
(Optional) Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see **Init Containers**.

- (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see **Configuring Container Lifecycle Parameters**.
- (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see **Configuring Environment Variables**.
- (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and mounting modes vary with the storage type. For details, see **Storage**.

NOTE

If the workload contains more than one pod, EVS volumes cannot be mounted.

  - (Optional) **Logging**: Report standard container output logs to AOM by default, without requiring manual settings. You can manually configure the log collection path. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.

    To disable the standard output of the current workload, add the annotation **kubernetes.AOM.log.stdout: []** in **Labels and Annotations**. For details about how to use this annotation, see **Table 5-14**.

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to access images in SWR Shared Edition. For details about **default-secret**, see **default-secret**.
- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

(Optional) Advanced Settings

- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see **Configuring Labels and Annotations**.
- **Job Settings**
  - **Parallel Pods**: Maximum number of pods that can run in parallel during job execution. The value cannot be greater than the total number of pods in the job.
  - **Timeout (s)**: Once a job reaches this time, the job status becomes failed and all pods in this job will be deleted. If you leave this parameter blank, the job will never time out.
  - Completion Mode
    - **Non-indexed**: A job is considered complete when all the pods are successfully executed. Each pod completion is homologous to each other.
    - **Indexed**: Each pod gets an associated completion index from 0 to the number of pods minus 1. The job is considered complete when every pod allocated with an index is successfully executed. For an indexed job, pods are named in the format of $(job-name)-$(index).
  - **Suspend Job**: By default, a job is executed immediately after being created. The job's execution will be suspended if you enable this option, and resumed after you disable it.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **Configuring QoS for a Pod**.
  - Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see

Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration.

- Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

A job has the following configuration parameters:

- **.spec.completions**: indicates the number of pods that need to run successfully to end a job. The default value is **1**.
- **.spec.parallelism**: indicates the number of pods that run concurrently. The default value is **1**.
- **.spec.backoffLimit**: indicates the maximum number of retries performed if a pod fails. When the limit is reached, the pod will not try again.
- **.spec.activeDeadlineSeconds**: indicates the running time of pods. Once the time is reached, the job and its pods are terminated. **.spec.activeDeadlineSeconds** has a higher priority than **.spec.backoffLimit**. If a job reaches **.spec.activeDeadlineSeconds**, **spec.backoffLimit** is ignored.

Based on the **.spec.completions** and **.spec.parallelism** settings, jobs are classified into the following types.

Table 5-3 Job types

Job Type	Description	.spec.completions	.spec.parallelism
One-off jobs	A job creates one pod until it successfully completes.	1	1
Jobs with a fixed completion count	A job creates one pod in sequence and is complete when the number of successful pods reaches the value of **.spec.completions**.	>1	1
Parallel jobs with a fixed completion count	A job creates multiple pods in sequence and is complete when the number of successful pods reaches the value of **.spec.completions**.	>1	>1

Job Type	Description	.spec.completions	.spec.parallelism
Parallel jobs with a work queue	A job creates one or more pods. Each pod takes one task from the message queue, processes it, and repeats until the end of the queue is reached. Then the pod deletes the task and exists. For details, see **Fine Parallel Processing Using a Work Queue**.	Leave this parameter blank.	>1 or =1

The following is an example job, which calculates π till the 2000th digit and prints the output.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 50        # A total of 50 pods need to run to finish a job. In this example, π is printed for 50 times.
  parallelism: 5        # A total of 5 pods run in parallel.
  backoffLimit: 5        # A maximum of 5 retries is allowed.
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never     # For a job, set this parameter to Never or OnFailure. For other controllers (such as Deployments), set this parameter to Always.
      imagePullSecrets:
        - name: default-secret
```

Run the job.

Step 1 Start the job.

```
[root@k8s-master k8s]# kubectl apply -f myjob.yaml
job.batch/myjob created
```

Step 2 View the job details.

kubectl get job

```
[root@k8s-master k8s]# kubectl get job
NAME    COMPLETIONS   DURATION   AGE
myjob   50/50         23s        3m45s
```

If the value of **COMPLETIONS** is **50/50**, the job is successfully executed.

Step 3 View the pod status.

kubectl get pod

```
[root@k8s-master k8s]# kubectl get pod
NAME          READY   STATUS      RESTARTS   AGE
myjob-29qlw   0/1     Completed   0          4m5s
...
```

If the status is **Completed**, the job is complete.

Step 4 View the pod logs.

kubectl logs <pod_name>

```
# kubectl logs myjob-29qlw
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019893809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151557485724245415069595082953311686172785588907509838175463746493931925506040092770167113900984882401285836160356370766010471018194295559619894676783744944825537977472684710404753464620804668425906949129331367702898915210475216205696602405803815019351125338243003558764024749647326391419927260426992279678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295532116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321721477235014144197356854816136115735255213347574184946843852332390739414333454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438382796797668145410095388378636095068006422512520511739298489608412848862694560424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759015
```

----End

Related Operations

After a one-off job is created, you can perform operations listed in **Table 5-4**.

Table 5-4 Other operations

Operation	Description
Viewing a YAML file	Locate the row containing the target job and choose **More** > **View YAML** to view the YAML file of the job.
Deleting a job	1. Select the target job and choose **More** > **Delete** in the **Operation** column. 2. Click **Yes**. Deleted jobs cannot be restored. Exercise caution when deleting a job.

5.2.5 Creating a Cron Job

Scenario

A cron job runs on a repeating schedule. You can perform time synchronization for all active nodes at a fixed time point.

A cron job runs periodically at the specified time. It is similar with Linux crontab. A cron job has the following characteristics:

- Runs only once at the specified time.
- Runs periodically at the specified time.

The typical usage of a cron job is as follows:

- Schedules jobs at the specified time.
- Creates jobs to run periodically, for example, database backup and email sending.

Prerequisites

Resources have been created. For details, see **Creating a Node**.

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **Cron Job**. For details about workload types, see **Overview**.
- **Workload Name**: Enter the name of the workload. Enter 1 to 63 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Container Runtime**: A CCE standard cluster uses a common runtime by default, whereas a CCE Turbo cluster supports both common and secure runtimes. For details about the differences, see **Secure Runtime and Common Runtime**.

Container Settings

- Container Information

  Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

  – **Basic Info**: Configure basic information about the container.

Parameter	Description
Container Name	Name the container.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist, the image is pulled from the image repository.

Parameter	Description
Image Name	Click **Select Image** and select the image used by the container. ■ **SWR Shared Edition**: easy-to-use, secure, and reliable image management ○ **My Images**: You can select a private image that you have uploaded. For details about how to upload images, see **Uploading an Image Through a Container Engine Client**. ○ **Open Source Images**: Select a public image provided by SWR. For details, see **Image Center**. ○ **Shared Images**: You can select an image shared by another account. For details, see **Sharing a Private Image**. To use a third-party image, directly enter image path. Ensure that the **image access credential** can be used to access the image repository. For details, see **Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores that can be used by a container. This prevents containers from using excessive resources. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. If **Request** and **Limit** are not specified, the quota is not limited. For more information and suggestions about **Request** and **Limit**, see **Configuring Container Specifications**.

Parameter	Description
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **CCE AI Suite (NVIDIA GPU)** add-on has been installed. ■ **Do not use**: No GPU will be used. ■ **GPU card**: The GPU is dedicated for the container. ■ **GPU Virtualization**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container will use 10% of GPU resources. For details about how to use GPUs in the cluster, see **Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required NPU chips. The value must be an integer and the **CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **NPU Scheduling**.
(Optional) Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
(Optional) Init Container	Whether to use the container as an init container. An init container does not support health check. An init container is a special container that runs before other app containers in a pod are started. Each pod can contain multiple containers. In addition, a pod can contain one or more init containers. Application containers in a pod are started and run only after the running of all init containers completes. For details, see **Init Containers**.

  - (Optional) **Lifecycle**: Configure operations to be performed in a specific phase of the container lifecycle, such as Startup Command, Post-Start, and Pre-Stop. For details, see **Configuring Container Lifecycle Parameters**.
  - (Optional) **Environment Variables**: Configure variables for the container running environment using key-value pairs. These variables transfer external information to containers running in pods and can be flexibly modified after application deployment. For details, see **Configuring Environment Variables**.

- **Image Access Credential**: Select the credential used for accessing the image repository. The default value is **default-secret**. You can use default-secret to

access images in SWR Shared Edition. For details about **default-secret**, see **default-secret**.

- (Optional) **GPU**: **All** is selected by default. The workload instance will be scheduled to the node of the specified GPU type.

Schedule

- **Concurrency Policy**: The following three modes are supported:
  - **Forbid**: A new job cannot be created before the previous job is completed.
  - **Allow**: The cron job allows concurrently running jobs, which preempt cluster resources.
  - **Replace**: A new job replaces the previous job when it is time to create a job but the previous job is not completed.
- **Policy Settings**: specifies when a new cron job is executed. Policy settings in YAML are implemented using cron expressions.
  - A cron job is executed at a fixed interval. The unit can be minute, hour, day, or month. For example, if a cron job is executed every 30 minutes and the corresponding cron expression is ***/30 * * * ***, the execution time starts from 0 in the unit range, for example, **00:00:00**, **00:30:00**, **01:00:00**, and **...**.
  - The cron job is executed at a fixed time (by month). For example, if a cron job is executed at 00:00 on the first day of each month, the cron expression is **0 0 1 */1 ***, and the execution time is ******-01-01 00:00:00**, ******-02-01 00:00:00**, and **...**.
  - The cron job is executed by week. For example, if a cron job is executed at 00:00 every Monday, the cron expression is **0 0 * * 1**, and the execution time is ******-**-01 00:00:00 on Monday**, ******-**-08 00:00:00 on Monday**, and **...**.
  - **Custom Cron Expression**: For details about how to use cron expressions, see **CronJob**.

    NOTE

    - If a cron job is executed at a fixed time (by month) and the number of days in a month does not exist, the cron job will not be executed in this month. For example, the execution will skip February if the date is set to 30.
    - Due to the definition of cron, the fixed period is not a strict period. The time unit range is divided from 0 by period. For example, if the unit is minute, the value ranges from 0 to 59. If the value cannot be exactly divided, the last period is reset. Therefore, an accurate period can be represented only when the period can be evenly divided.

      Take a cron job that runs hourly as an example. If an interval that can divide 24 hours exactly, such as **/2, /3, /4, /6, /8, and /12**, the period can be accurately represented. However, if a different period is used, the last period will reset at the beginning of each new day. For example, if the cron expression is *** */12 * * ***, the task will run at **00:00:00** and **12:00:00** every day. Similarly, if the cron expression is *** */13 * * ***, the task will run at **00:00:00** and **13:00:00** every day. Note that the last period will reset at 00:00:00 on the following day, even if it has not reached 13 hours.
- **Time Zone**: You can specify a time zone. If this parameter is not specified, the time zone of the master node will be used by default.

- **Job Records**: You can set the number of jobs that are successfully executed or fail to be executed. Setting a limit to **0** corresponds to keeping none of the jobs after they finish.

(Optional) Advanced Settings

- **Labels and Annotations**: Add labels or annotations for pods using key-value pairs. After entering the key and value, click **Confirm**. For details about how to use and configure labels and annotations, see **Configuring Labels and Annotations**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **Configuring QoS for a Pod**.
  - Whether to enable a specified container network configuration: available only for clusters that support this function. After you enable a specified container network configuration, the workload will be created using the container subnet and security group in the configuration. For details, see **Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration**.
  - Specify the container network configuration name: Only the custom container network configuration whose associated resource type is workload can be selected.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

A cron job has the following configuration parameters:

- **.spec.schedule**: takes a **Cron** format string, for example, **0 * * * *** or **@hourly**, as schedule time of jobs to be created and executed.
- **.spec.jobTemplate**: specifies jobs to be run, and has the same schema as when you are **Creating a Job Using kubectl**.
- **.spec.startingDeadlineSeconds**: specifies the deadline for starting a job.
- **.spec.concurrencyPolicy**: specifies how to treat concurrent executions of a job created by the CronJob. The following options are supported:
  - **Allow** (default value): allows concurrently running jobs.
  - **Forbid**: forbids concurrent runs, skipping next run if previous has not finished yet.
  - **Replace**: cancels the currently running job and replaces it with a new one.

The following is an example cron job, which is saved in the **cronjob.yaml** file.

NOTE

In clusters of v1.21 or later, CronJob apiVersion is **batch/v1**.

In clusters earlier than v1.21, CronJob apiVersion is **batch/v1beta1**.

```
apiVersion: batch/v1
kind: CronJob
```



```
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            command:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
          imagePullSecrets:
            - name: default-secret
```

Run the job.

Step 1 Create a CronJob.

kubectl create -f cronjob.yaml

Information similar to the following is displayed:

```
cronjob.batch/hello created
```

Step 2 Query the running status of the cron job:

kubectl get cronjob

```
NAME      SCHEDULE      SUSPEND   ACTIVE    LAST SCHEDULE   AGE
hello     */1 * * * *   False     0         <none>          9s
```

kubectl get jobs

```
NAME               COMPLETIONS   DURATION   AGE
hello-1597387980   1/1           27s        45s
```

kubectl get pod

```
NAME                     READY     STATUS      RESTARTS   AGE
hello-1597387980-tjv8f   0/1       Completed   0          114s
hello-1597388040-lckg9   0/1       Completed   0          39s
```

kubectl logs hello-1597387980-tjv8f

```
Fri Aug 14 06:56:31 UTC 2020
Hello from the Kubernetes cluster
```

kubectl delete cronjob hello

```
cronjob.batch "hello" deleted
```

NOTICE

When a CronJob is deleted, the related jobs and pods are deleted accordingly.

----End

Related Operations

After a CronJob is created, you can perform operations listed in **Table 5-5**.

Table 5-5 Other operations

Operation	Description
Editing a YAML file	Click **More** > **Edit YAML** next to the cron job name to edit the YAML file of the current job.
Stopping a CronJob	1. Select the job to be stopped and click **Stop** in the **Operation** column. 2. Click **Yes**.
Deleting a CronJob	1. Select the CronJob to be deleted and click **More** > **Delete** in the **Operation** column. 2. Click **Yes**. Deleted jobs cannot be restored. Exercise caution when deleting a job.

5.3 Configuring a Workload

5.3.1 Secure Runtime and Common Runtime

Compared with a common runtime, a secure runtime allows each container (pod) to run on its own micro-VM with a separate OS kernel. This ensures secure isolation at the virtualization layer. With a secure runtime, kernels, compute resources, and networks are isolated between containers to protect pod resources and data from being preempted and stolen by other pods.

CCE Turbo clusters allow you to create workloads using a common runtime or secure runtime as required. The differences between them are as follows.

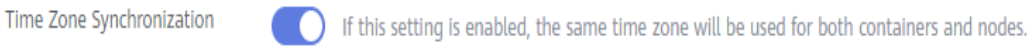
Category	Secure Runtime	Common Runtime
Node type used to run containers	ECS (PM)	ECS (VM) ECS (PM)
Container engine	containerd	Docker and containerd
Container runtime	Kata	runC
Container kernel	Exclusive kernel	Sharing the kernel with the host
Container isolation	Lightweight VMs	cgroups and namespaces

Category	Secure Runtime	Common Runtime
Container engine storage driver	Device Mapper	• Docker container: OverlayFS2 • containerd container: OverlayFS
Pod overhead	Memory: 100 MiB CPU: 0.1 cores Pod overhead is a feature for accounting for the resources consumed by the pod infrastructure on top of the container requests and limits. For example, if **limits.cpu** is set to 0.5 cores and **limits.memory** to 256 MiB for a pod, the pod will request 0.6 CPU cores and 356 MiB of memory.	None
Minimize flavor	Memory: 256 MiB CPU: 0.25 cores It is recommended that the ratio of CPU (unit: core) to memory (unit: GiB) be in the range of 1:1 to 1:8. For example, if CPU is 0.5 cores, the memory should range form 512 MiB to 4 GiB.	None
Container engine CLI	crictl	• Docker container: docker • containerd container: crictl
Pod computing resources	The request and limit values must be the same for both CPU and memory.	The request and limit values can be different for both CPU and memory.
hostNetwork	Not supported	Supported

For details about how to use containerd and Docker commands, see **Container Engines**.

5.3.2 Configuring Time Zone Synchronization

When creating a workload, you can configure containers to use the same time zone as the node. You can enable time zone synchronization when creating a workload.



The time zone synchronization function depends on the local disk (hostPath) mounted to the container. After time zone synchronization is enabled, **/etc/localtime** of the node is mounted to **/etc/localtime** of the container in HostPath mode, in this way, the node and container use the same time zone configuration file.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      volumes:
        - name: vol-162979628557461404
          hostPath:
            path: /etc/localtime
            type: ''
      containers:
        - name: container-0
          image: 'nginx:alpine'
          volumeMounts:
            - name: vol-162979628557461404
              readOnly: true
              mountPath: /etc/localtime
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

5.3.3 Configuring an Image Pull Policy

When a workload is created, the container image is pulled from the image repository to the node. The image is also pulled when the workload is restarted or upgraded.

By default, **imagePullPolicy** is set to **IfNotPresent**, indicating that if the image exists on the node, the existing image is used. If the image does not exist on the node, the image is pulled from the image repository.

The image pull policy can also be set to **Always**, indicating that the image is pulled from the image repository and overwrites the image on the node regardless of whether the image exists on the node.

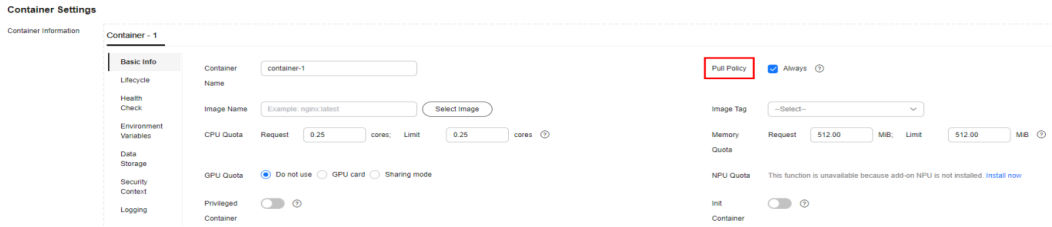
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
```



```
      memory: 200Mi
    imagePullPolicy: Always
  imagePullSecrets:
  - name: default-secret
```

An image pull policy can also be configured on the CCE console. When creating a workload, configure **Pull Policy**. If **Always** is selected, images are always pulled. If **Always** is not selected, images are pulled as needed.

Figure 5-5 Configuring an update policy



NOTICE

Use a new tag each time you create an image. If you do not update the tag but only update the image, when **Pull Policy** is set to **IfNotPresent**, CCE considers that an image with the tag already exists on the current node and will not pull the image again.

5.3.4 Using Third-Party Images

Scenario

CCE allows you to create workloads using images pulled from third-party image repositories.

Generally, a third-party image repository can be accessed only after authentication (using your account and password). CCE uses the secret-based authentication to pull images. Therefore, create a secret for an image repository before pulling images from the repository.

Prerequisites

The node where the workload is running is accessible from public networks.

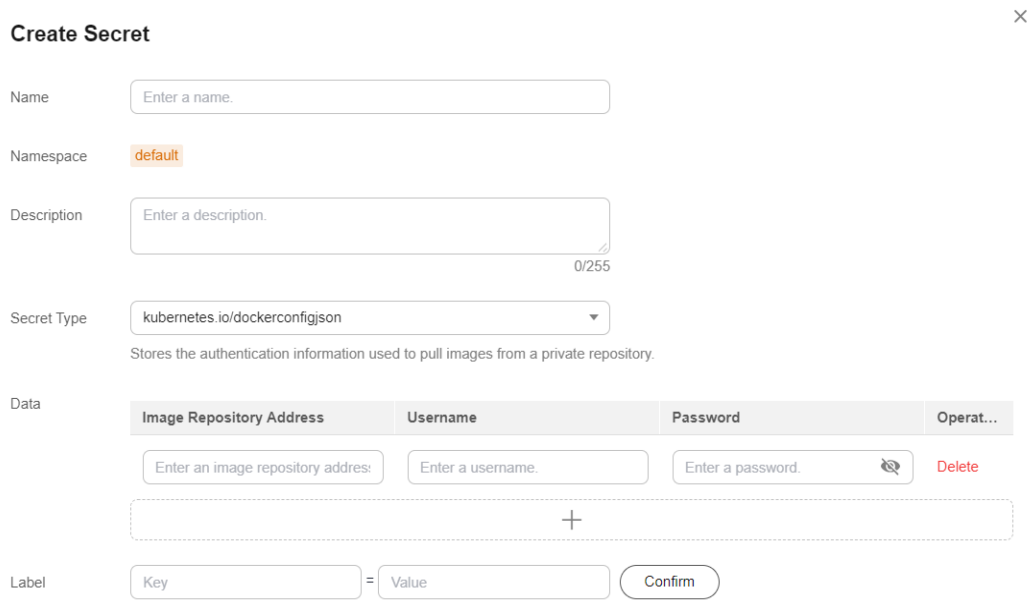
Using the Console

Step 1 Create a secret for accessing a third-party image repository.

Click the cluster name to access the cluster console. In the navigation pane, choose **ConfigMaps and Secrets**. On the **Secrets** tab page, click **Create Secret** in the upper right corner. Set **Secret Type** to **kubernetes.io/dockerconfigjson**. For details, see **Creating a Secret**.

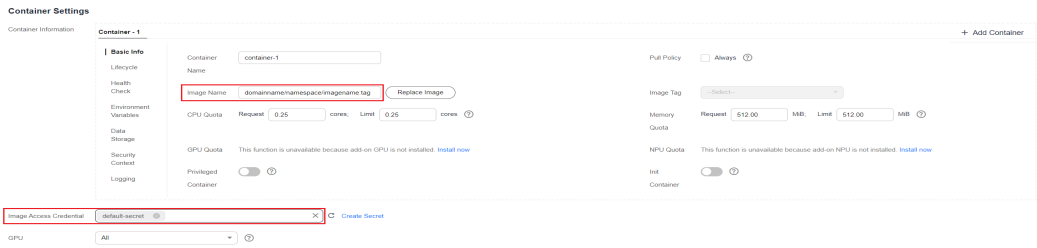
Enter the username and password used to access the third-party image repository.

Figure 5-6 Creating a secret



Step 2 When creating a workload, enter a private image path in the format of *domainname/namespace/imagename:tag* in **Image Name** and select the key created in **Step 1**.

Figure 5-7 Private image path



Step 3 Set other parameters and click **Create Workload**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Use kubectl to create a secret of the kubernetes.io/dockerconfigjson.

```
kubectl create secret docker-registry myregistrykey  -n default --docker-server=DOCKER_REGISTRY_SERVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
```

In the preceding command, *myregistrykey* indicates the key name, *default* indicates the namespace where the key is located, and other parameters are as follows:

- **DOCKER_REGISTRY_SERVER**: address of a third-party image repository, for example, **www.3rdregistry.com** or **10.10.10.10:443**

- **DOCKER_USER**: account used for logging in to a third-party image repository
- **DOCKER_PASSWORD**: password used for logging in to a third-party image repository
- **DOCKER_EMAIL**: email of a third-party image repository

Step 3 Use a third-party image to create a workload.

A kubernetes.io/dockerconfigjson secret is used for authentication when you obtain a private image. The following is an example of using the myregistrykey for authentication.

```
apiVersion: v1
kind: Pod
metadata:
  name: foo
  namespace: default
spec:
  containers:
    - name: foo
      image: www.3rdregistry.com/janedoe/awesomeapp:v1
  imagePullSecrets:
    - name: myregistrykey              # Use the created secret.
```

----End

5.3.5 Configuring Container Specifications

Scenario

CCE allows you to set resource requirements and limits, such as CPU and RAM, for added containers during workload creation. Kubernetes also allows using YAML to set requirements of other resource types.

Request and Limit

For **CPU** and **Memory**, the meanings of **Request** and **Limit** are as follows:

- **Request**: The system schedules a pod to the node that meets the requirements for workload deployment based on the request value.
- **Limit**: The system limits the resources used by the workload based on the limit value.

If a node has sufficient resources, the pod on this node can use more resources than requested, but no more than limited.

For example, if you set the memory request of a container to 1 GiB and the limit value to 2 GiB, a pod is scheduled to a node with 8 GiB CPUs with no other pod running. In this case, the pod can use more than 1 GiB memory when the load is heavy, but the memory usage cannot exceed 2 GiB. If a process in a container attempts to use more than 2 GiB resources, the system kernel attempts to terminate the process. As a result, an out of memory (OOM) error occurs.

NOTE

When creating a workload, you are advised to set the upper and lower limits of CPU and memory resources. If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

Configuration

In real-world scenarios, the recommended ratio of **Request** to **Limit** is about 1:1.5. For some sensitive services, the recommended ratio is 1:1. If the **Request** is too small and the **Limit** is too large, node resources are oversubscribed. During service peaks, the memory or CPU of a node may be used up. As a result, the node is unavailable.

- CPU quota: The unit of CPU resources is core, which can be expressed by quantity or an integer suffixed with the unit (m). For example, 0.1 core in the quantity expression is equivalent to 100m in the expression. However, Kubernetes does not allow CPU resources whose precision is less than 1m.

  Table 5-6 CPU quotas

Parameter	Description
CPU request	Minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available CPU on the node is greater than or equal to the number of containerized CPU applications.
CPU limit	Maximum number of CPU cores available for a container.

  Recommended configuration

  Actual available CPU of a node ≥ Sum of CPU limits of all containers on the current node ≥ Sum of CPU requests of all containers on the current node. You can view the actual available CPUs of a node on the CCE console (**Resource Management** > **Nodes** > **Allocatable**).

- Memory quota: The default unit of memory resources is byte. You can also use an integer with the unit suffix, for example, 100 Mi. Note that the unit is case-sensitive.

  Table 5-7 Description of memory quotas

Parameter	Description
Memory request	Minimum amount of memory required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available memory on the node is greater than or equal to the number of containerized memory applications.
Memory Limit	Maximum amount of memory available for a container. When the memory usage exceeds the configured memory limit, the instance may be restarted, which affects the normal use of the workload.

  Recommended configuration

Actual available memory of a node ≥ Sum of memory limits of all containers on the current node ≥ Sum of memory requests of all containers on the current node. You can view the actual available memory of a node on the CCE console (**Resource Management** > **Nodes** > **Allocatable**).

NOTE

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node (for details, see **Example of CPU and Memory Quota Usage**). The calculation formula is as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

Example of CPU and Memory Quota Usage

Assume that a cluster contains a node with 4 CPU cores and 8 GiB memory. Two pods (pod 1 and pod 2) have been deployed on the cluster. Pod 1 oversubscribes resources (that is **Limit** > **Request**). The specifications of the two pods are as follows.

Pod	CPU Request	CPU Limit	Memory Request	Memory Limit
Pod 1	1 core	2 cores	1 GiB	4 GiB
Pod 2	2 cores	2 cores	2 GiB	2 GiB

The CPU and memory usage of the node is as follows:

- Allocatable CPUs = 4 cores – (1 core requested by pod 1 + 2 cores requested by pod 2) = 1 core
- Allocatable memory = 8 GiB – (1 GiB requested by pod 1 + 2 GiB requested by pod 2) = 5 GiB

In this case, the remaining 1 core 5 GiB can be used by the next new pod.

If pod 1 is under heavy load during peak hours, it will use more CPUs and memory within the limit. Therefore, the actual allocatable resources are fewer than 1 core 5 GiB.

Quotas of Other Resources

Typically, nodes support local ephemeral storage, which is provided by locally mounted writable devices or RAM. EV does not ensure long-term data availability. Pods can use local EVs to buffer data and store logs, or mount emptyDir volumes to containers. For details, see **Local ephemeral storage**.

Kubernetes allows you to specify the requested value and limit value of ephemeral storage in container configurations to manage the local ephemeral storage. The following attributes can be configured for each container in a pod:

- spec.containers[].resources.limits.ephemeral-storage
- spec.containers[].resources.requests.ephemeral-storage

In the following example, a pod contains two containers. The requested value of each container for local ephemeral storage is 2 GiB, and the limit value is 4 GiB. Therefore, the requested value of the pod for local ephemeral storage is 4 GiB, the limit value is 8 GiB, and the emptyDir volume uses 500 MiB of the local ephemeral storage.

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: container-1
    image: <example_app_image>
    resources:
      requests:
        ephemeral-storage: "2Gi"
      limits:
        ephemeral-storage: "4Gi"
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  - name: container-2
    image: <example_log_aggregator_image>
    resources:
      requests:
        ephemeral-storage: "2Gi"
      limits:
        ephemeral-storage: "4Gi"
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  volumes:
    - name: ephemeral
      emptyDir:
        sizeLimit: 500Mi
```

5.3.6 Configuring Container Lifecycle Parameters

Scenario

CCE provides callback functions for the lifecycle management of containerized applications. For example, if you want a container to perform a certain operation before stopping, you can register a hook function.

CCE provides the following lifecycle callback functions:

- **Startup Command**: executed to start a container. For details, see **Startup Commands**.
- **Post-Start**: executed immediately after a container is started. For details, see **Post-Start Processing**.
- **Pre-Stop**: executed before a container is stopped. The pre-stop processing function helps you ensure that the services running on the pods can be completed in advance in the case of pod upgrade or deletion. For details, see **Pre-Stop Processing**.

Startup Commands

By default, the default command is executed during image start. To run a specific command or rewrite the default image setting, you must perform specific operations.

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, CCE runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

Table 5-8 Commands and arguments used to run a container

Image ENTRYPOINT	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

Step 1 Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Enter a command and arguments on the **Startup Command** tab page.

Table 5-9 Container startup command

Configuration Item	Procedure
Command	Enter an executable command, for example, **/run/server**. If there are multiple executable commands, write them in different lines. **NOTE** In the case of multiple commands, you are advised to run **/bin/sh** or other **shell** commands. Other commands are used as parameters.

Configuration Item	Procedure
Args	Enter the argument that controls the container running command, for example, **--port=8080**. If there are multiple arguments, separate them in different lines.

----End

Post-Start Processing

Step 1 Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Set the post-start processing parameters on the **Post-Start** tab page.

Table 5-10 Post-start processing parameters

Parameter	Description

CLI	Set commands to be executed in the container for post-start processing. The command format is **Command Args[1] Args[2]...**. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution. **Commands that are executed in the background or asynchronously are not supported.**
	Example command:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Enter **/install install_agent** in the script. This command indicates that **install.sh** will be executed after the container is created successfully.

HTTP request	Send an HTTP request for post-start processing. The related parameters are described as follows: • **Path**: (optional) request URL. • **Port**: (mandatory) request port. • **Host**: (optional) requested host IP address. The default value is the IP address of the pod.

----End

Pre-Stop Processing

Step 1 Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Set the pre-start processing parameters on the **Pre-Stop** tab page.

Table 5-11 Pre-stop processing parameters

Parameter	Description

CLI	Set commands to be executed in the container for pre-stop processing. The command format is **Command Args[1] Args[2]....** **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.
	Example command:
	`exec:`
	`  command:`
	`  - /uninstall.sh`
	`  - uninstall_agent`
	Enter **/uninstall uninstall_agent** in the script. This command indicates that **uninstall.sh** will be executed before the container completes its execution and stops running.

HTTP request	Send an HTTP request for pre-stop processing. The related parameters are described as follows: • **Path**: (optional) request URL. • **Port**: (mandatory) request port. • **Host**: (optional) requested host IP address. The default value is the IP address of the pod.

----End

YAML Example

This section uses Nginx as an example to describe how to set the container lifecycle.

In the following configuration file, the **postStart** command is defined to run the **install.sh** command in the **/bin/bash** directory. **preStop** is defined to run the **uninstall.sh** command.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        command:
```



```
    - sleep 3600                  # Startup command
    imagePullPolicy: Always
    lifecycle:
      postStart:
        exec:
          command:
          - /bin/bash
          - install.sh            # Post-start command
      preStop:
        exec:
          command:
          - /bin/bash
          - uninstall.sh          # Pre-stop command
    name: nginx
  imagePullSecrets:
  - name: default-secret
```

5.3.7 Configuring Container Health Check

Scenario

Health check regularly checks the health status of containers during container running. If the health check function is not configured, a pod cannot detect application exceptions or automatically restart the application to restore it. This will result in a situation where the pod status is normal but the application in the pod is abnormal.

Kubernetes provides the following health check probes:

- **Liveness probe** (livenessProbe): checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness check of a container fails, the cluster restarts the container. If the liveness check is successful, no operation is executed.
- **Readiness probe** (readinessProbe): checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some applications to start up before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, although the application process has started, the application cannot provide services. To address this issue, this health check probe is used. If the container readiness check fails, the cluster masks all requests sent to the container. If the container readiness check is successful, the container can be accessed.
- **Startup probe** (startupProbe): checks when a containerized application has started. If such a probe is configured, it disables liveness and readiness checks until it succeeds, ensuring that those probes do not interfere with the application startup. This can be used to adopt liveness checks on slow starting containers, avoiding them getting terminated by the kubelet before they are started.

Check Method

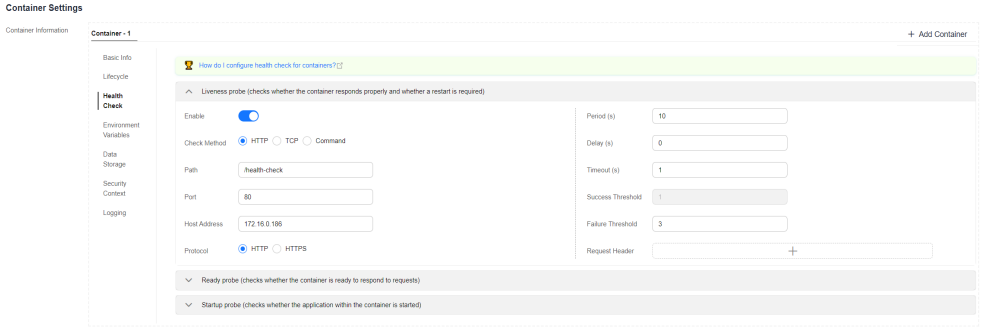
- **HTTP request**

  This health check mode applies to containers that provide HTTP/HTTPS services. The cluster periodically initiates an HTTP/HTTPS GET request to such containers. If the return code of the HTTP/HTTPS response is within 200–399,

the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port and an HTTP/HTTPS request path.

For example, for a container that provides HTTP services, the HTTP check path is **/health-check**, the port is 80, and the host address is optional (which defaults to the container IP address). Here, 172.16.0.186 is used as an example, and we can get such a request: GET http://172.16.0.186:80/health-check. The cluster periodically initiates this request to the container. You can also add one or more headers to an HTTP request. For example, set the request header name to **Custom-Header** and the corresponding value to **example**.

Figure 5-8 HTTP request-based check

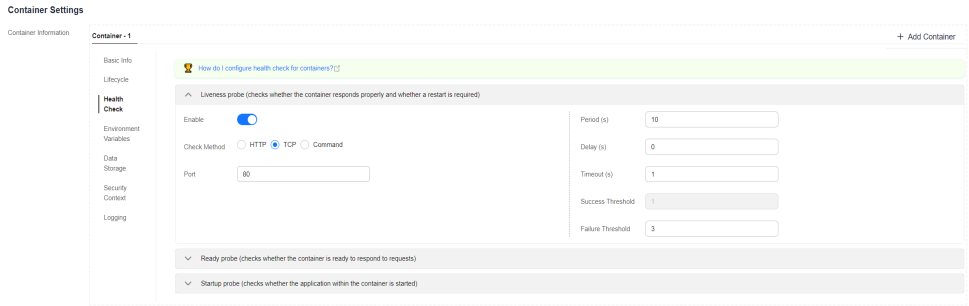


- **TCP port**

  For a container that provides TCP communication services, the cluster periodically establishes a TCP connection to the container. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port.

  For example, if you have an Nginx container with service port 80, after you specify TCP port 80 for container listening, the cluster will periodically initiate a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

Figure 5-9 TCP port-based check



- **CLI**

  CLI is an efficient tool for health check. When using the CLI, you must specify an executable command in a container. The cluster periodically runs the command in the container. If the command output is 0, the health check is successful. Otherwise, the health check fails.

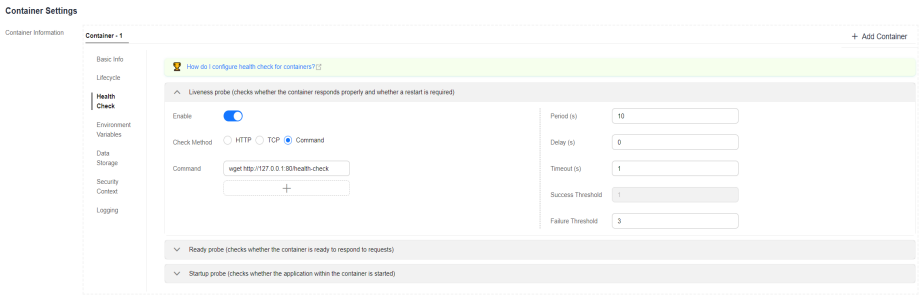
  The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

- For a TCP port, you can use a program script to connect to a container port. If the connection is successful, the script returns **0**. Otherwise, the script returns **–1**.
- For an HTTP request, you can use the script command to run the **wget** command to detect the container.

  wget http://127.0.0.1:80/health-check

  Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **–1**.

Figure 5-10 CLI-based check



NOTICE

- Put the program to be executed in the container image so that the program can be executed.
- If the command to be executed is a shell script, do not directly specify the script as the command, but add a script parser. For example, if the script is **/data/scripts/health_check.sh**, you must specify **sh/data/scripts/health_check.sh** for command execution.

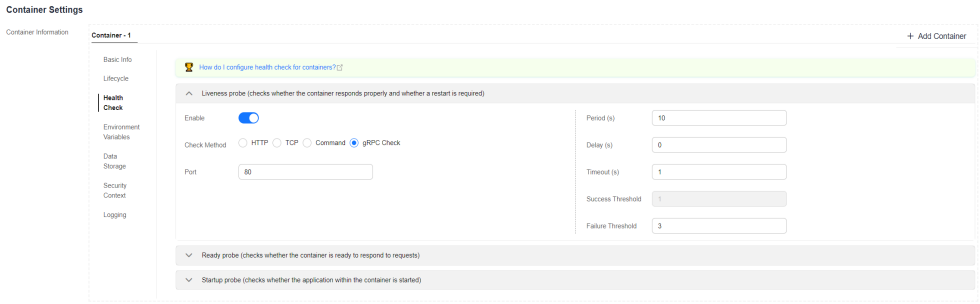
- **gRPC check**

  gRPC checks can configure startup, liveness, and readiness probes for your gRPC application without exposing any HTTP endpoint, nor do you need an executable. Kubernetes can connect to your workload via gRPC and obtain its status.

NOTICE

- The gRPC check is supported only in CCE clusters of v1.25 or later.
- To use gRPC for check, your application must support the **gRPC health checking protocol**.
- Similar to HTTP and TCP probes, if the port is incorrect or the application does not support the health checking protocol, the check fails.

Figure 5-11 gRPC check



Common Parameters

Table 5-12 Common parameters

Parameter	Description
Period (periodSeconds)	Indicates the probe detection period, in seconds. For example, if this parameter is set to **30**, the detection is performed every 30 seconds.
Delay (initialDelaySeconds)	Check delay time in seconds. Set this parameter according to the normal startup time of services. For example, if this parameter is set to **30**, the health check will be started 30 seconds after the container is started. The time is reserved for containerized services to start.
Timeout (timeoutSeconds)	Number of seconds after which the probe times out. Unit: second. For example, if this parameter is set to **10**, the timeout wait time for performing a health check is 10s. If the wait time elapses, the health check is regarded as a failure. If the parameter is left blank or set to **0**, the default timeout time is 1s.
Success Threshold (successThreshold)	Minimum consecutive successes for the probe to be considered successful after having failed. For example, if this parameter is set to **1**, the workload status is normal only when the health check is successful for one consecutive time after the health check fails. The default value is **1**, which is also the minimum value. The value of this parameter is fixed to **1** in **Liveness Probe** and **Startup Probe**.
Failure Threshold (failureThreshold)	Number of retry times when the detection fails. Giving up in case of liveness probe means to restart the container. In case of readiness probe the pod will be marked Unready. The default value is **3**, and the minimum value is **1**.

YAML Example

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: <image_address>
    args:
    - /server
    livenessProbe:              # Liveness probe
      httpGet:                  # Checking an HTTP request is used as an example.
        path: /healthz          # The HTTP check path is /healthz.
        port: 80                # The check port number is 80.
        httpHeaders:            # (Optional) The request header name is Custom-Header and the value is
Awesome.
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:             # Readiness probe
      exec:                     # Checking an execution command is used as an example.
        command:                # Command to be executed
          - cat
          - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
    startupProbe:               # Startup probe
      httpGet:                  # Checking an HTTP request is used as an example.
        path: /healthz          # The HTTP check path is /healthz.
        port: 80                # The check port number is 80.
      failureThreshold: 30
      periodSeconds: 10
```

5.3.8 Configuring Environment Variables

Scenario

An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

The function of setting environment variables on CCE is the same as that of specifying **ENV** in a Dockerfile.

NOTICE

After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

Environment variables can be set in the following modes:

- **Custom**: Enter the environment variable name and parameter value.
- **Added from ConfigMap**: Import all key values in a ConfigMap as environment variables.
- **Added from ConfigMap key**: Import the value of a key in a ConfigMap as the value of an environment variable. As shown in **Figure 5-12**, if you import **configmap_value** of **configmap_key** in **configmap-example** as the value of environment variable **key1**, an environment variable named **key1** whose value is **configmap_value** is available in the container.
- **Added from secret**: Import all key values in a secret as environment variables.
- **Added from secret key**: Import the value of a key in a secret as the value of an environment variable. As shown in **Figure 5-12**, if you import **secret_value** of **secret_key** in **secret-example** as the value of environment variable **key2**, an environment variable named **key2** whose value is **secret_value** is available in the container.
- **Variable Value/Reference**: Use the field defined by a pod as the value of the environment variable. As shown in **Figure 5-12**, if the pod name is imported as the value of environment variable **key3**, an environment variable named **key3** whose value is the pod name is available in the container.
- **Resource Reference**: The value of **Request** or **Limit** defined by the container is used as the value of the environment variable. As shown in **Figure 5-12**, if you import the CPU limit of container-1 as the value of environment variable **key4**, an environment variable named **key4** whose value is the CPU limit of container-1 is available in the container.

Adding Environment Variables

Step 1 Log in to the CCE console.

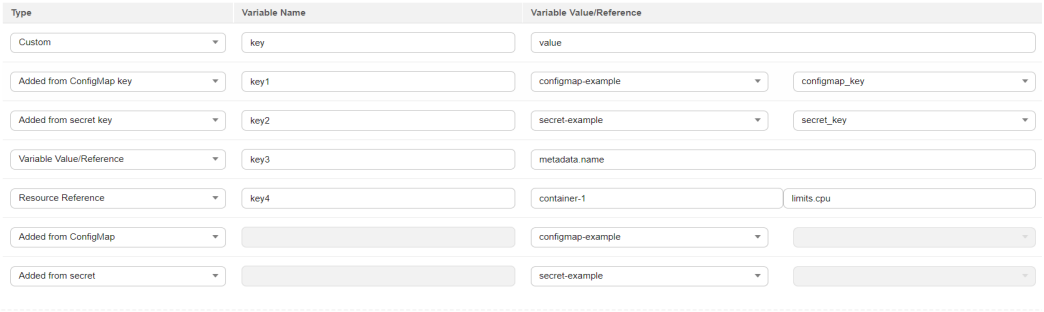
Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 When creating a workload, modify the container information in **Container Settings** and click the **Environment Variables** tab.

Step 4 Configure environment variables.

- To add environment variables one by one, click **Adding a Variable** and configure its parameters.
- To add environment variables in batches, click **Editing Custom Variables in Batches**. Then, in the displayed dialog box, enter environment variables in the format of "Variable name=Variable or variable reference".

Figure 5-12 Configuring environment variables



----End

YAML Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          env:
            - name: key                   # Custom
              value: value
            - name: key1                  # Added from ConfigMap key
              valueFrom:
                configMapKeyRef:
                  name: configmap-example
                  key: configmap_key
            - name: key2                  # Added from secret key
              valueFrom:
                secretKeyRef:
                  name: secret-example
                  key: secret_key
            - name: key3                  # Variable reference, which uses the field defined by a pod as the value of the environment variable.
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: key4                  # Resource reference, which uses the field defined by a container as the value of the environment variable.
              valueFrom:
                resourceFieldRef:
                  containerName: container1
```



```
            resource: limits.cpu
            divisor: 1
      envFrom:
        - configMapRef:                # Added from ConfigMap
            name: configmap-example
        - secretRef:                   # Added from secret
            name: secret-example
    imagePullSecrets:
      - name: default-secret
```

Viewing Environment Variables

If the contents of **configmap-example** and **secret-example** are as follows:

```
$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVl              # c2VjcmV0X3ZhbHVl is the value of secret_value in Base64 mode.
kind: Secret
...
```

The environment variables in the pod are as follows:

```
$ kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
env-example-695b759569-lx9jp   1/1     Running   0          17m

$ kubectl exec env-example-695b759569-lx9jp  -- printenv
/ # env
key=value                           # Custom environment variable
key1=configmap_value                # Added from ConfigMap key
key2=secret_value                   # Added from secret key
key3=env-example-695b759569-lx9jp     # metadata.name defined by the pod
key4=1                              # limits.cpu defined by container1. The value is rounded up, in unit of cores.
configmap_key=configmap_value         # Added from ConfigMap. The key value in the original ConfigMap key is directly imported.
secret_key=secret_value               # Added from key. The key value in the original secret is directly imported.
```

5.3.9 Configuring APM

Scenario

Application Performance Management (APM) allows you to monitor Java workloads through tracing and topology. You can install APM probes to locate and analyze problems for Java workloads.

You can configure Java workload monitoring when and after a workload is created.

NOTE

- Connect your Java application on CCE to APM through the Pinpoint probe. For details, see **Connecting a Huawei Cloud Containerized Application to APM**.

Prerequisites

If you have not enabled the APM service, go to the APM console and enable it as prompted.

Procedure

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 When creating a workload, click **APM Settings** in the **Advanced Settings** area. By default, the probe is disabled. You can choose **APM 1.0** as required. After the probe is enabled, APM can locate and analyze problems for Java programs.

NOTE

1. The APM 1.0 probe will be initialized in an auto created init container named init-pinpoint. The init container will be allocated 0.25-core CPU and 250 MiB memory.
2. Adding an APM probe will add the environment variables **PAAS_MONITORING_GROUP**, **JAVA_TOOL_OPTIONS**, and **PAAS_CLUSTER_ID** to all service containers.
3. Adding an APM probe will mount a local storage volume named **paas-apm** (for APM 1.0 probe) to all service containers.

Step 4 Set probe-related parameters.

APM 1.0 probe

- **Monitoring Group**: Enter a monitoring group name, for example, **testapp**.
- **Probe Version**: Select the probe version.
- **Probe Upgrade Policy**: By default, **Auto upgrade upon restart** is selected.
  - **Auto upgrade upon restart**: The system downloads the probe image each time the pod is restarted.
  - **Manual upgrade upon restart**: This policy means that if a local image is available, the local image will be used. The system downloads the probe image only when a local image is unavailable.

Step 5 Three minutes after the application is started, its data will be displayed on the APM console. You can log in to the APM console and optimize application performance through topology and tracing. For details, see **Topology**.

----End

Configuring APM Settings

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the desired workload name.

Step 3 On the page displayed, click the **APM Settings** tab and click **Edit** in the lower right corner.

For details about the parameters, see **Step 4**.

----End

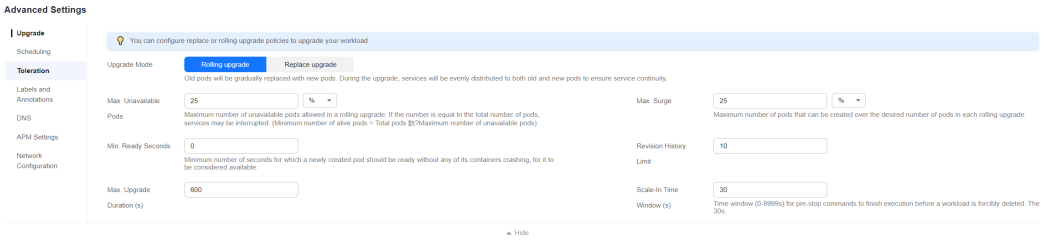
5.3.10 Configuring Workload Upgrade Policies

In actual applications, upgrade is a common operation. A Deployment, StatefulSet, or DaemonSet can easily support application upgrade.

You can set different upgrade policies:

- **Rolling upgrade**: New pods are created gradually and then old pods are deleted. This is the default policy.
- **Replace upgrade**: The current pods are deleted and then new pods are created.

Figure 5-13 Workload upgrade mode



Upgrade Parameters

Parameter	Description	Constraint
Max. Surge (maxSurge)	Specifies the maximum number of pods that can exist compared with **spec.replicas**. The default value is **25%**. For example, if **spec.replicas** is set to **4**, a maximum of five pods can exist during the upgrade. That is, the upgrade is performed at a step of 1. During the actual upgrade, the value is converted into a number and rounded up. The value can also be set to an absolute number.	This parameter is supported only by Deployments and DaemonSets.
Max. Unavailable Pods (maxUnavailable)	Specifies the maximum number of pods that can be unavailable compared with **spec.replicas**. The default value is **25%**. For example, if **spec.replicas** is set to **4**, at least three pods exist during the upgrade. That is, the deletion is performed at a step of 1. The value can also be set to an absolute number.	This parameter is supported only by Deployments and DaemonSets.
Min. Ready Seconds (minReadySeconds)	A pod is considered available only when the minimum readiness time is exceeded without any of its containers crashing. The default value is **0** (the pod is considered available immediately after it is ready).	None

Parameter	Description	Constraint
Revision History Limit (revisionHistory-Limit)	Specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of **kubectl get rs**. The configuration of each Deployment revision is stored in its ReplicaSets. Therefore, once the old ReplicaSet is deleted, you lose the ability to roll back to that revision of Deployment. By default, 10 old ReplicaSets will be kept, but the ideal value depends on the frequency and stability of the new Deployments.	None
Max. Upgrade Duration (progressDeadlineSeconds)	Specifies the number of seconds that the system waits for a Deployment to make progress before reporting a Deployment progress failure. It is surfaced as a condition with Type=Progressing, Status=False, and Reason=ProgressDeadlineExceeded in the status of the resource. The Deployment controller will keep retrying the Deployment. In the future, once automatic rollback will be implemented, the Deployment controller will roll back a Deployment as soon as it observes such a condition. If this parameter is specified, the value of this parameter must be greater than that of **.spec.minReadySeconds**.	None
Scale-In Time Window (terminationGracePeriodSeconds)	Graceful deletion time. The default value is 30 seconds. When a pod is deleted, a SIGTERM signal is sent and the system waits for the applications in the container to terminate. If the application is not terminated within the time specified by **terminationGracePeriodSeconds**, a SIGKILL signal is sent to forcibly terminate the pod.	None

Upgrade Example

The Deployment can be upgraded in a declarative mode. That is, you only need to modify the YAML definition of the Deployment. For example, you can run the **kubectl edit** command to change the Deployment image to **nginx:alpine**. After the modification, query the ReplicaSet and pod. The query result shows that a new ReplicaSet is created and the pod is re-created.

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME            DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd  2         2         2         1m
nginx-7f98958cdf  0         0         0         48m
```



```
$ kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

The Deployment can use the **maxSurge** and **maxUnavailable** parameters to control the proportion of pods to be re-created during the upgrade, which is useful in many scenarios. The configuration is as follows:

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

In the preceding example, the value of **spec.replicas** is **2**. If both **maxSurge** and **maxUnavailable** are the default value 25%, **maxSurge** allows a maximum of three pods to exist (2 x 1.25 = 2.5, rounded up to 3), and **maxUnavailable** does not allow a maximum of two pods to be unavailable (2 x 0.75 = 1.5, rounded up to 2). That is, during the upgrade process, there will always be two pods running. Each time a new pod is created, an old pod is deleted, until all pods are new.

Rollback

Rollback is to roll an application back to an earlier version when a fault occurs during an upgrade. A Deployment can be easily rolled back to the earlier version.

For example, if the upgraded image is faulty, you can run the **kubectl rollout undo** command to roll back the Deployment.

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

A Deployment can be easily rolled back because it uses a ReplicaSet to control a pod. After the upgrade, the previous ReplicaSet still exists. The Deployment is rolled back by using the previous ReplicaSet to re-create the pod. The number of ReplicaSets stored in a Deployment can be restricted by the **revisionHistoryLimit** parameter. The default value is **10**.

5.3.11 Configuring Tolerance Policies

Tolerations allow the scheduler to schedule pods to nodes with target taints. Tolerances work with **node taints**. Each node allows one or more taints. If no tolerance is configured for a pod, the scheduler will schedule the pod based on node taint policies to prevent the pod from being scheduled to an inappropriate node. For more examples, see **Taints and Tolerations**.

The following table shows how taint policies and tolerations affect pod running.

Taint Policy	No Taint Toleration Configured	Taint Toleration Configured
NoExecute	• Pods running on the node will be evicted immediately. • Inactive pods will not scheduled to the node.	• If the tolerance time window **tolerationSeconds** is not specified, pods can always run on this node. • If the tolerance time window **tolerationSeconds** is specified, pods still run on the node with taints within the time window. After the time expires, the pods will be evicted.
PreferNoSchedule	• Pods running on the node will not be evicted. • Inactive pods will not scheduled to the node **to the best extend**.	Pods can always run on this node.
NoSchedule	• Pods running on the node will not be evicted. • Inactive pods will not scheduled to the node.	Pods can always run on this node.

Configuring Tolerance Policies on the Console

Step 1 Log in to the CCE console.

Step 2 When creating a workload, click **Toleration** in the **Advanced Settings** area.

Step 3 Add a taint tolerance policy.

Table 5-13 Parameters for configuring a taint tolerance policy

Parameter	Description
Taint key	Key of a node taint
Operator	• **Equal**: **Exact match** for the specified taint key (mandatory) and taint value. If the taint value is left blank, all taints with the key the same as the specified taint key will be matched. • **Exists**: **matches only** the nodes with the specified taint key. In this case, the taint value cannot be specified. If the taint key is left blank, all taints will be tolerated.
Taint value	Taint value specified if the operator is set to **Equal**.

Parameter	Description
Taint Policy	• **All**: All taint policies are matched. • **NoSchedule**: Only the **NoSchedule** taint is matched. • **PreferNoSchedule**: Only the **PreferNoSchedule** taint is matched. • **NoExecute**: Only the **NoExecute** taint is matched.
Toleration Time Window	**tolerationSeconds**, which is configurable only when **Taint Policy** is set to **NoExecute**. Within the tolerance time window, pods still run on the node with taints. After the time expires, the pods will be evicted.

----End

Default Tolerance Policy

Kubernetes automatically adds tolerances for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints to pods, and sets the tolerance time window (**tolerationSeconds**) to 300s. These default tolerance policies indicate that when either of the preceding taint is added to the node where pods are running, the pods can still run on the node for 5 minutes.

NOTE

When a DaemonSet pod is created, no tolerance time window will be specified for the tolerances automatically added for the preceding taints. When either of the preceding taints is added to the node where the DaemonSet pod is running, the DaemonSet pod will never be evicted.

```
tolerations:
  - key: node.kubernetes.io/not-ready
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
  - key: node.kubernetes.io/unreachable
    operator: Exists
    effect: NoExecute
    tolerationSeconds: 300
```

5.3.12 Configuring Labels and Annotations

Pod Annotations

CCE allows you to add annotations to a YAML file to realize some advanced pod functions. The following table describes the annotations you can add.

Table 5-14 Pod annotations

Annotation	Description	Default Value

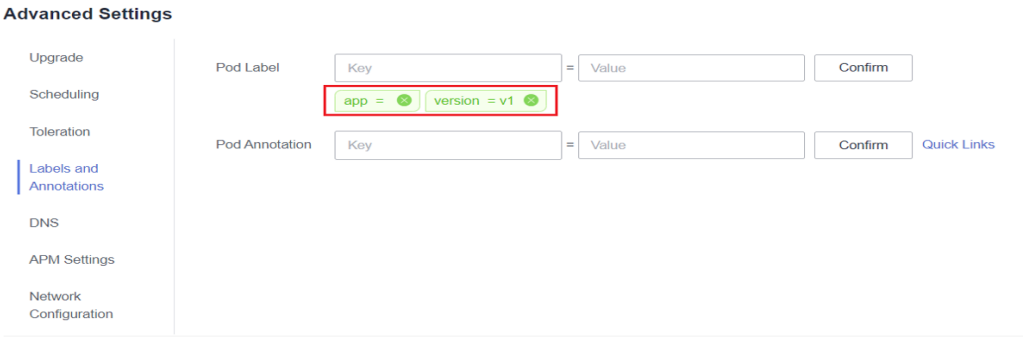
kubernetes.AOM.log.stdout	Standard output parameter. If not specified, the standard log output of all containers is reported to AOM. You can collect stdout logs from certain containers or ignore them at all.
	Example:
	• Collecting none of the stdout logs:
	`kubernetes.AOM.log.stdout: '[]'`
	• Collecting stdout logs of container-1 and container-2:
	`kubernetes.AOM.log.stdout: '["container-1","container-2"]'`
		None

metrics.alpha.kubernetes.io/custom-endpoints	Parameter for reporting AOM monitoring metrics that you specify. For details, see **Monitoring Custom Metrics on AOM**.	None
prometheus.io/scrape	Parameter for reporting Prometheus metrics. If the value is **true**, the current workload reports the monitoring metrics. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	None
prometheus.io/path	URL for Prometheus to collect data. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	/metrics
prometheus.io/port	Endpoint port number for Prometheus to collect data. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	None
prometheus.io/scheme	Protocol used by Prometheus to collect data. The value can be **http** or **https**. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	None
kubernetes.io/ingress-bandwidth	Ingress bandwidth of a pod. For details, see **Configuring QoS for a Pod**.	None

Annotation	Description	Default Value
kubernetes.io/egress-bandwidth	Egress bandwidth of a pod. For details, see **Configuring QoS for a Pod**.	None

Pod Labels

When you create a workload on the console, the following labels are added to the pod by default. The value of **app** is the workload name.



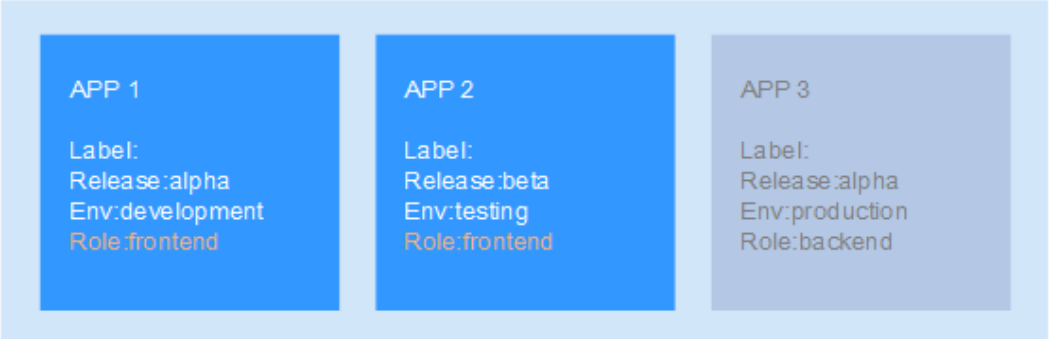
Example YAML:

```
...
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      ...
```

You can also add other labels to the pod for affinity and anti-affinity scheduling. In the following figure, three pod labels (release, env, and role) are defined for workload APP 1, APP 2, and APP 3. The values of these labels vary with workload.

- APP 1: [release:alpha;env:development;role:frontend]
- APP 2: [release:beta;env:testing;role:frontend]
- APP 3: [release:alpha;env:production;role:backend]

Figure 5-14 Label example



For example, if **key/value** is set to **role/backend**, APP 3 will be selected for affinity scheduling. For details, see **Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**.

5.4 Scheduling a Workload

5.4.1 Overview

Kubernetes schedules workloads based on pods. After you create a workload, the scheduler automatically assigns pods. For example, the scheduler distributes pods to nodes that have enough resources.

While the default scheduler behavior is sufficient for most needs, there may be situations where you require more precise control over where pods are deployed. To address this, Kubernetes enables you to configure scheduling policies when creating workloads. For example, you may need to:

- Deploy a frontend application and a backend application on the same node. This reduces latency because the pods of both applications can use the physical resources of the node.
- Deploy a certain type of applications on specific nodes to ensure that these critical applications always run on the best hardware or configuration.
- Deploy different applications on separate nodes to isolate them from each other. This prevents any issues with one application from affecting others.

Use the methods listed in the following table to select a pod scheduling policy in Kubernetes.

Table 5-15 Workload scheduling policies

Scheduling Policy	YAML Field	Description	Reference
Node selector	nodeSelector	A basic scheduling mode, in which Kubernetes selects the target node according to node labels. This means that pods are only scheduled to the node that has the specific label.	**Configuring Specified Node Scheduling (nodeSelector)**
Node affinity	nodeAffinity	Node affinity is more expressive than nodeSelector. It allows you to use **Label Selectors** to filter nodes that require affinity based on their labels and choose between **Required** and **Preferred** for **Affinity Rules**. **NOTE** When both nodeSelector and nodeAffinity are specified, a pod can only be scheduled to a candidate node if both conditions are met.	**Configuring Node Affinity Scheduling (nodeAffinity)**
Workload affinity or anti-affinity scheduling	podAffinity/ podAntiAffinity	You can use **Label Selectors** to filter pods that require affinity or anti-affinity based on workload labels. You can then determine whether to schedule new workload pods to the same node (or node group) as the target pod or not. Additionally, you can choose between **Required** and **Preferred** for **Affinity Rules**. **NOTE** Workload affinity and anti-affinity require a certain amount of computing time, which significantly slows down scheduling in large-scale clusters. Do not enable workload affinity and anti-affinity in a cluster that contains hundreds of nodes.	**Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**

Affinity Rules

Scheduling policies that use node affinity or workload affinity/anti-affinity can include both hard and soft constraints to meet complex scheduling requirements. Hard constraints must be met, while soft constraints should be met as much as possible.

Table 5-16 Affinity rules

Rule Type	YAML Field	Description	Example
Required	requiredDuringSchedulingIgnoredDuringExecution	Hard constraint that **must be met**. The scheduler can perform scheduling only when the rule is met.	• Configuring Node Affinity Scheduling (nodeAffinity) • Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)
Preferred	preferredDuringSchedulingIgnoredDuringExecution	Soft constraint. The scheduler tries to locate the target object that satisfies the target rule. The scheduler will schedule the pod even if it cannot find a matching target object that satisfies the target rule. When using preferred affinity, you can set a weight field ranging from 1 to 100 for each pod. Assigning a higher weight to a pod will increase its priority in the scheduling process.	

NOTE

The YAML field **requiredDuringScheduling** or **preferredDuringScheduling** in the affinity rules above indicates that a label rule must be forcibly met or needs to be met as much as possible during scheduling. **IgnoredDuringExecution** indicates that any changes to the node label after Kubernetes schedules the pod will not affect the pod's running or cause it to be rescheduled. However, if kubelet on the node is restarted, kubelet will recheck the node affinity rule, and the pod will still be scheduled to another node.

Label Selectors

When creating a scheduling policy use the logical operators of a label selector to filter label values and identify the objects that need affinity or anti-affinity.

Table 5-17 Label selectors

Parameter	Description	YAML Example
key	Label key. The objects that meet the filter criteria must contain the label of the key, and the label value must meet the operation relationship between the label value list (**values** field) and logical operators.	In the example below, objects that meet the filter criteria must have a label with a key of **topology.kubernetes.io/zone** and a value of either **az1** or **az2**. matchExpressions: - key: **topology.kubernetes.io/zone** operator: **In** values: - **az1** - **az2**
operator	Logical operator that can be used to determine filtering rules for label values. Options: • **In**: The label of the affinity or anti-affinity object **is** in the label value list (**values** field). • **NotIn**: The label of the affinity or anti-affinity object is **not** in the label value list (**values** field). • **Exists**: The affinity or anti-affinity object **has** a specified label key. There is no need to configure the label value list (**values** field). • **DoesNotExist**: The affinity or anti-affinity object does **not** have a specified label key. There is no need to configure the label value list (**values** field). • **Gt**: (available only for node affinity) The label value of the scheduled node is **greater** than what is listed (string comparison). • **Lt**: (available only for node affinity) The label value of the scheduled node is **less** than what is listed (string comparison).	
values	Label values	

5.4.2 Configuring Specified Node Scheduling (nodeSelector)

To select a node for scheduling in Kubernetes, simply configure the **nodeSelector** field in the workload. This field allows you to configure the label of the desired node to be scheduled. Kubernetes schedules pods only to nodes with specified labels.

Prerequisites

A custom label has been added to the target node so that workload pods can be scheduled based on the node label. For details, see **Adding or Deleting a Node Label**.

Creating a Workload Scheduled to a Specified Node

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **nginx.yaml**. The file name can be customized.

Configure nodeSelector for the workload. For example, if the key is **deploy_qa** and the value is **true**, the pod will be scheduled to the node with the **deploy_qa=true** label. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      nodeSelector:
        deploy_qa: "true"
      containers:
      - image: nginx:latest
        imagePullPolicy: IfNotPresent
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

Step 3 Create a workload.

```
kubectl apply -f nginx.yaml
```

Step 4 Verify that all pods run on the target node.

```
kubectl get pod -o wide
```

In the following command output, node 192.168.0.103 is labeled with **deploy_qa=true**:

```
NAME                     READY   STATUS    RESTARTS   AGE     IP            NODE
NOMINATED NODE   READINESS GATES
nginx-66859f4f48-xgp2g   1/1     Running   0          6h57m   172.16.3.0    192.168.0.103   <none>
<none>
nginx-66859f4f48-t9gqj   1/1     Running   0          6h57m   172.16.3.1    192.168.0.103   <none>
<none>
nginx-66859f4f48-2grhq   1/1     Running   0          6h57m   172.16.3.2    192.168.0.103   <none>
<none>
```

----End

5.4.3 Configuring Node Affinity Scheduling (nodeAffinity)

Kubernetes can schedule workload pods to affinity nodes based on their labels and label values. For example, some nodes support GPU computing, and node affinity scheduling can guarantee that high-performance computing pods run on these GPU nodes.

Configuring a Node Affinity Scheduling Policy

There are various ways to configure node affinity scheduling policies to schedule pods to nodes that meet specific requirements.

Using the CCE Console

In this example, a GPU node labeled with **gpu=true** has been created in the cluster. You can use this label to schedule pods to this GPU node.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 In **Advanced Settings**, choose **Scheduling** and select a policy for **Node Affinity**. In this example, **Custom policies** is selected. For details about how to create a workload, see **Creating a Workload**.

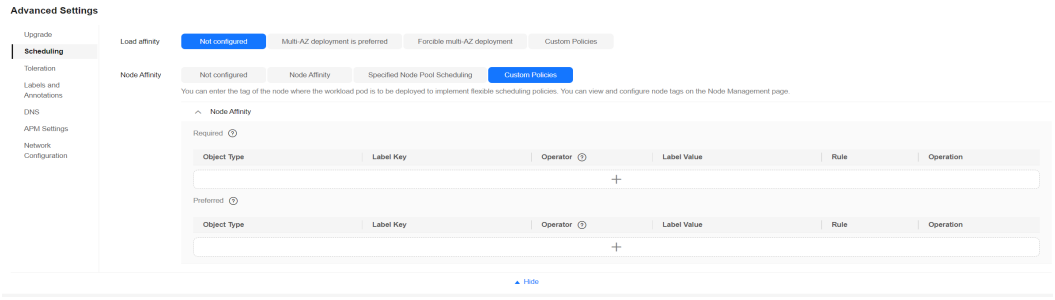


Table 5-18 Scheduling policies

Parameter	Description	Example
Node Affinity	• **Not configured**: No node affinity policy is configured. • **Node Affinity**: Specify the nodes where workload pods are to be deployed. If no nodes are specified, the pods will be randomly scheduled according to the default cluster scheduling policy. • **Specified Node Pool Scheduling**: Specify the node pools where workload pods are to be deployed. If no node pools are specified, the pods will be randomly scheduled according to the default cluster scheduling policy. • **Custom policies**: allow flexible scheduling of workload pods based on node labels. For details about the supported affinity policies, see **Table 5-19**. Select a proper policy type and add a policy. For details about the parameters, see **Table 5-20**.	Custom policies

Step 4 Select a proper node affinity rule and click + to add a scheduling policy. In this example, a scheduling policy is added in the **Required** area under **Node Affinity**.

This policy specifies that the newly created workload can only be scheduled to a specific node.

Table 5-19 Affinity rule

Parameter	Description	Example
Node Affinity	• **Required**: indicates the hard constraint (**requiredDuringSchedulingIgnoredDuringExecution**) that must be met. If multiple rules that must be met are added, scheduling will be performed when only one rule is met. • **Preferred**: indicates the soft constraint (**preferredDuringSchedulingIgnoredDuringExecution**) that need to be met as much as possible. If multiple rules that are preferentially met are added, scheduling will be performed even if one or none of the rules is met.	Required

Step 5 In the dialog box that is displayed on the right, click **Add Policy** to configure rules for filtering node labels.

You can also click **Specify Node** or **Specify AZ** to quickly select a node or AZ on the console for scheduling.

Specifying a node or AZ is also implemented through labels. The console frees you from manually entering node labels. The **kubernetes.io/hostname** label is used when you specify a node, and the **failure-domain.beta.kubernetes.io/zone** label is used when you specify an AZ.

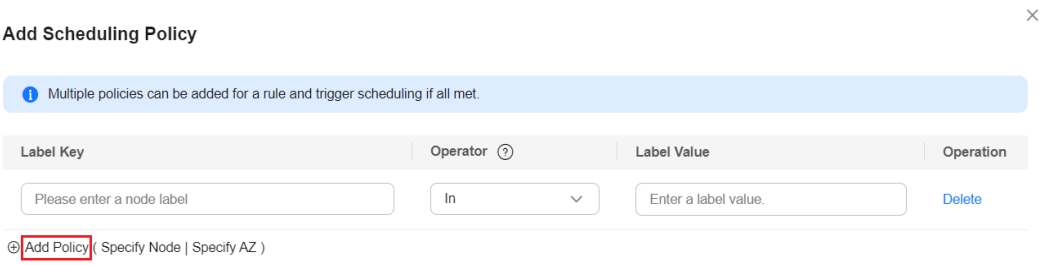


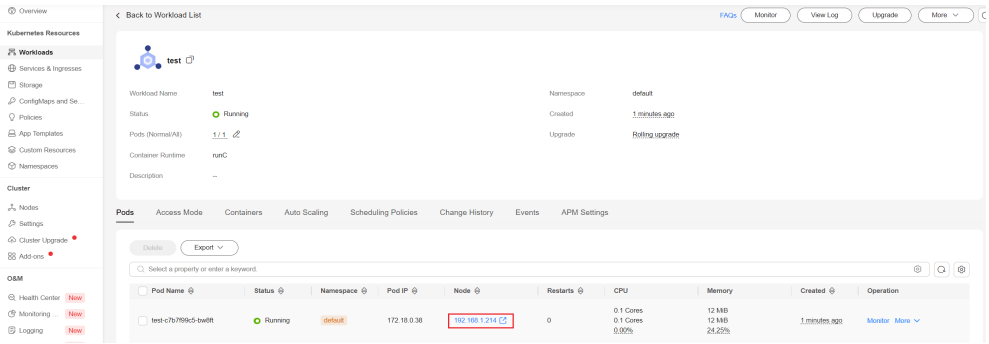
Table 5-20 Parameters for configuring node affinity scheduling policies

Parameter	Description	Example
Weight	This parameter is available only in a **preferred** scheduling policy. Weights range from 1 to 100 and are taken into account as an extra scoring factor during scheduling. The scheduler combines the weight with other priority functions of the node to determine the final score and then assigns pods to the node with the highest total score.	None
Label Key	When configuring node affinity, enter the node label to be matched. Both default labels and custom labels are supported.	gpu
Operator	The following operators are supported: • **In**: The label of the affinity or anti-affinity object is in the label value list (**values** field). • **NotIn**: The label of the affinity or anti-affinity object is not in the label value list (**values** field). • **Exists**: The affinity or anti-affinity object has a specified label key. • **DoesNotExist**: The affinity or anti-affinity object does not have a specified label key. • **Gt**: The label value of the scheduling node is greater than what is listed (character string comparison). • **Lt**: The label value of the scheduling node is less than what is listed (character string comparison).	In
Label Value	When configuring node affinity, enter the value of the node label.	true

Step 6 After the scheduling policy is added, click **Create Workload**.

Step 7 Verify that all pods run on the target node.

1. In the navigation pane, choose **Workloads**.
2. Click the workload name to enter its details page. On the **Pods** tab page, verify that all pods run on the target node, which is labeled with **gpu=true**.



----End

Using YAML

Workload node affinity rules are implemented using node labels. When a node is created in a CCE cluster, it is automatically assigned certain labels. Here are some examples of commonly used node labels (for more labels, see **Inherent Label of a Node**):

- **topology.kubernetes.io/zone**: the AZ where the node is located, which can be used for scheduling in a specified AZ.
- **kubernetes.io/hostname**: hostname of a node, which can be used for specified node scheduling.
- **cce.cloud.com/cce-nodepool**: node pool to which a node belongs, which can be used for scheduling in a specified node pool.

In this example, the rule that **must be met** indicates that the node to be scheduled must be labeled with key **gpu** and value **true**. The rule that needs to be **met as much as possible** indicates that pods are preferentially scheduled to nodes in AZ 1 based on **topology.kubernetes.io/zone**. The following is an example of configuring node affinity:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:  # Configure a scheduling policy.
        nodeAffinity:  # Node affinity scheduling
          requiredDuringSchedulingIgnoredDuringExecution:  # Scheduling policy that must be met
            nodeSelectorTerms:     # Select a node that meets the requirements according to the node label.
              - matchExpressions:    # Node label matching rule
                - key: gpu   # The key of the node label is gpu.
                  operator: In  # The rule is met if a value exists in the value list.
                  values:   # The value of the node label is true.
                  - "true"
          preferredDuringSchedulingIgnoredDuringExecution:  # Scheduling policy that is met as much as
possible
            - weight: 100  # Priority that can be configured when the best-effort policy is used. The value
ranges from 1 to 100. A larger value indicates a higher priority.
```



```
        preference:  # Preferred node label matching rule when the best-effort policy is used
          matchExpressions:   # Node label matching rule
            - key: topology.kubernetes.io/zone   # Nodes' AZs
              operator: In  # The rule is met if a value exists in the value list.
              values:   # The value of the node label is az1.
              - "az1"
```

NOTE

There is no anti-affinity policy for node affinity scheduling. For node anti-affinity, you can filter nodes by label using the **NotIn** and **DoesNotExist** operators.

5.4.4 Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)

Kubernetes offers workload affinity and anti-affinity scheduling, which allows for flexible scheduling of new workloads on either related or unrelated nodes. This results in improved cluster resource utilization.

For example, frontend workload pods and backend workload pods that frequently communicate with each other can be preferentially scheduled to the same node or AZ to minimize network latency. The process of workload affinity/anti-affinity is as follows:

1. Nodes are classified using node labels based on topology keys (**topologyKey**).

   For example, if **topologyKey** is **prefer**, nodes will be classified using node label **prefer**. Nodes labeled with **prefer=true** will be assigned to topology key 1, while those labeled with **prefer=false** will be assigned to topology key 2. Nodes without a **prefer** label will be assigned to topology key 3.

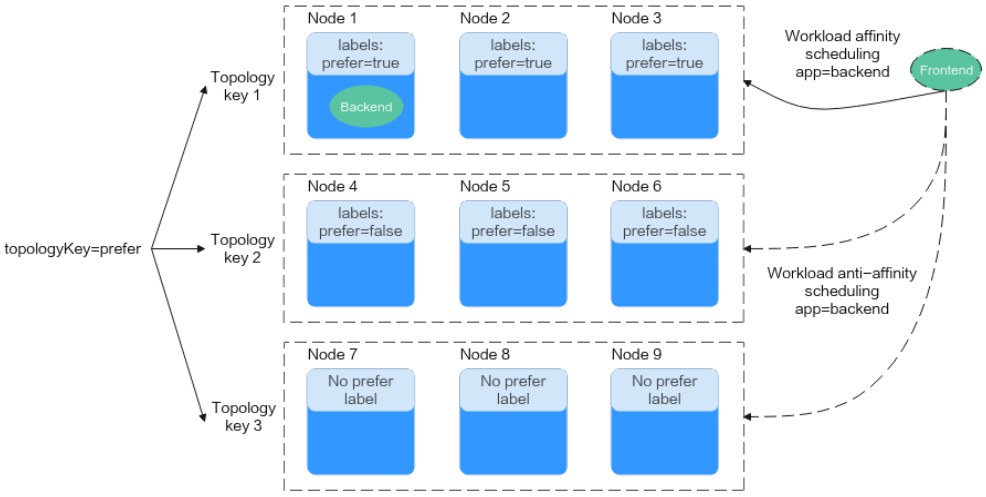
2. Identify the affinity/anti-affinity workloads based on the workload labels and operators.

   For example, the label selector filters the workloads that have been labeled with **app=backend**.

3. For affinity scheduling, the scheduler chooses the topology key where the target workload is located, while for anti-affinity scheduling, it selects a topology key where the target workload is not present.

   In this example, the workload labeled with **app=backend** is assigned to topology key 1. Therefore, workloads that have an affinity with the **app=backend** workload can be scheduled to topology key 1. Conversely, workloads that have an anti-affinity with the **app=backend** workload can only be scheduled to topology key 2 or 3.

Figure 5-15 Workload affinity or anti-affinity scheduling



Configuring Load Affinity/Anti-affinity

There are various ways to configure load affinity or anti-affinity scheduling policies to schedule pods to nodes that meet specific requirements.

Using the CCE Console

In this example, a backend workload with the **app=backend** label has been created in the cluster. This label can be used for workload affinity or anti-affinity scheduling, allowing the newly created frontend workload labeled with **app=frontend** to be deployed on the same node as the backend workload. Both workloads run in topology key **kubernetes.io/hostname**. When you use **kubernetes.io/hostname** for topology classification, only one node can be assigned to each topology key because each node has a unique hostname. This enables workloads to be scheduled on the same node when workload affinity is enabled.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 In **Advanced Settings**, choose **Scheduling** and select a policy for **Load Affinity**. In this example, **Custom policies** is selected. For details about how to create a workload, see **Creating a Workload**.

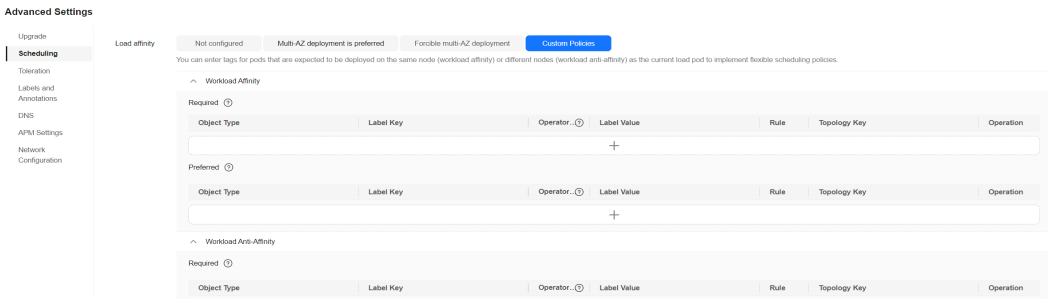


Table 5-21 Scheduling policies

Parameter	Description	Example
Load affinity	• **Not configured**: No load affinity policy is configured. • **Multi-AZ deployment preferred**: Workload pods are **preferentially** scheduled to nodes in different AZs through pod anti-affinity. The AZs serve as topology keys in this process. • **Forcible multi-AZ deployment**: Workload pods are **forcibly** scheduled to nodes in different AZs through pod anti-affinity. The AZs serve as topology keys in this process. When this scheduling policy is used, if there are fewer nodes than pods or node resources are insufficient, the extra pods will fail to run. • **Custom policies**: allow flexible scheduling of workload pods based on pod labels. For details about the supported scheduling policies, see **Table 5-22**. Select a proper policy type and add a policy. For details about the parameters, see **Table 5-23**.	Custom policies

Step 4 Select a proper load affinity rule and click + to add a scheduling policy. In this example, a scheduling policy is added in the **Required** area under **Workload Affinity**. This policy specifies that the newly created workload can only be scheduled to a node if a workload with specific labels is already running on that node.

Table 5-22 Load affinity policies

Policy	Rule Type	Description	Example
Workload affinity	Required	Hard constraint, which corresponds to **requiredDuringSchedulingIgnoredDuringExecution** in YAML for specifying the conditions that must be met. Select pods that require affinity by label. If such pods already run on a node in the topology key, the scheduler will **forcibly** schedule the created pods to that topology key. **NOTE** If multiple affinity rules are configured, multiple labels will be used to filter pods that require affinity, and the newly created pods must be affinity with all pods that meet the label filtering conditions. In this way, all pods that meet the label filtering conditions locate in the same topology key for scheduling.	Required

Policy	Rule Type	Description	Example
	Preferred	Soft constraint, which corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in YAML for specifying the conditions that need to be met as much as possible. Select pods that require affinity by label. If such pods already run on a node in the topology key, the scheduler will **preferentially** schedule the created pods to that topology key. **NOTE** If multiple affinity rules are configured, multiple labels will be used to filter pods that require affinity, and the newly created pods will be preferentially to be affinity with multiple pods that meet the label filtering conditions. However, even if no pod meets the label filter conditions, a topology key will be selected for scheduling.	
Workload anti-affinity	Required	Hard constraint, which corresponds to **requiredDuringSchedulingIgnoredDuringExecution** in YAML for specifying the conditions that must be met. Select one or more pods that require anti-affinity by label. If such pods already run on a node in the topology key, the scheduler will **not** schedule the created pods to that topology key. **NOTE** If multiple anti-affinity rules are configured, multiple labels will be used to filter pods that require anti-affinity, and the newly created pods must be anti-affinity with all pods that meet the label filtering conditions. In this way, all the topology keys where the pods that meet the label filtering conditions locate will not be scheduled.	None

Policy	Rule Type	Description	Example
	Preferred	Soft constraint, which corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in YAML for specifying the conditions that need to be met as much as possible. Select one or more pods that require anti-affinity by label. If such pods already run on a node in the topology key, the scheduler will **preferentially** schedule the created pods to other topology keys. **NOTE** If multiple anti-affinity rules are configured, multiple labels will be used to filter pods that require anti-affinity, and the newly created pods will be preferentially to be anti-affinity with multiple pods that meet the label filtering conditions. However, even if all topology keys involve the pods that require anti-affinity, a topology key will be selected for scheduling.	

Step 5 In the dialog box that is displayed on the right, click **Add Policy** to configure rules for filtering node labels.

Table 5-23 Parameters for configuring load affinity/anti-affinity scheduling policies

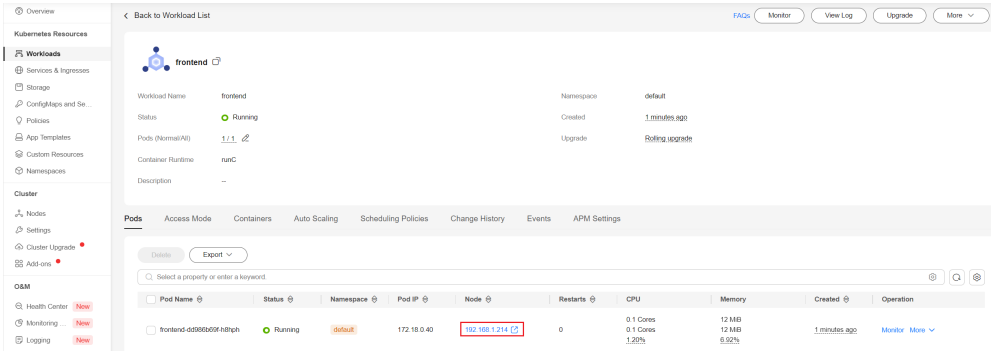
Parameter	Description	Example
Weight	This parameter is available only in a **preferred** scheduling policy. Weights range from 1 to 100 and are taken into account as an extra scoring factor during scheduling. The scheduler combines the weight with other priority functions of the node to determine the final score and then assigns pods to the node with the highest total score.	None
Namespace	Namespace for which the scheduling policy takes effect.	default

Parameter	Description	Example
Topology Key	A topology key (**topologyKey**) determines the range of nodes to be scheduled based on node labels, identifies affinity/anti-affinity objects based on the labels and operators, and performs scheduling based on the topology key where the target object is located. • For example, if the node label is **kubernetes.io/hostname**, the label value will be a node name. Nodes with different names are assigned to different topology keys. This allows for workload affinity scheduling on a single node, as each topology key contains only one node. • If the specified label is **kubernetes.io/os**, the label value will be a node OS. Nodes running different OSs are assigned to different topology keys. This allows for workload affinity scheduling on multiple nodes, as each topology key contains multiple nodes. For example, if pods that meet the load affinity rule are running on a node in a topology key, all nodes in the topology key can be scheduled.	kubernetes.io/ hostname
Label Key	When configuring a workload affinity or anti-affinity policy, enter the workload label to be matched. Both default labels and custom labels are supported.	app
Operator	The following operators are supported: • **In**: The label of the affinity or anti-affinity object is in the label value list (**values** field). • **NotIn**: The label of the affinity or anti-affinity object is not in the label value list (**values** field). • **Exists**: The affinity or anti-affinity object has a specified label key. • **DoesNotExist**: The affinity or anti-affinity object does not have a specified label key.	In
Label Value	When configuring a workload affinity or anti-affinity policy, enter the value of the workload label.	backend

Step 6 After the scheduling policy is added, click **Create Workload**.

Step 7 Verify that all pods run on the target node.

1. In the navigation pane, choose **Workloads**.
2. Click the workload name to enter its details page. On the **Pods** tab page, verify that the new pod and the existing backend pod run on the same node.



----End

Using YAML

- Workload affinity

  Kubernetes supports affinity between pods, which allows the frontend and backend pods of an application to be deployed together to minimize access latency.

  Assume that the backend pods of an application have been created with label **app=backend**. You can use **.spec.affinity.podAffinity** to configure workload affinity so that the frontend pods (labeled **app=frontend**) and backend pods (labeled **app=backend**) can be deployed together.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - image: nginx:alpine
        name: frontend
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:  # Configure a scheduling policy.
        podAffinity:  # Workload affinity scheduling rule
          requiredDuringSchedulingIgnoredDuringExecution:   # Scheduling policy that must be met
          - topologyKey: prefer    # Topology keys are divided based on node labels, among which prefer is a custom label.
```



```
          labelSelector:  # Select workloads that meet the requirements based on workload labels.
            matchExpressions: # Workload label matching rule
            - key: app # The key of the workload label is app.
              operator: In # The rule is met if a value exists in the value list.
              values: # Workload label values
              - backend
        preferredDuringSchedulingIgnoredDuringExecution:    # Scheduling policy that is met as much as possible
        - weight: 100  # Priority that can be configured when the best-effort policy is used. The value ranges from 1 to 100. A larger value indicates a higher priority.
          podAffinityTerm:  # Affinity configuration when the best-effort policy is used
            topologyKey: topology.kubernetes.io/zone   # Topology keys are divided based on node labels by node AZ.
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```

During workload scheduling in the preceding example, node topology keys are divided based on the **prefer** label using the rule that must be met. If backend pods (labeled **app=backend**) are running on a node in the topology key, frontend pods (labeled **app=frontend**) will also be deployed in that topology key, even if not all nodes in the topology key are running the backend pods. According to the best-effort rule, topology keys are divided based on **topology.kubernetes.io/zone** by node AZ. This ensures that the frontend and backend pods are deployed on nodes within the same AZ as much as possible.

NOTE

For workload affinity, **topologyKey** cannot be left blank when **requiredDuringSchedulingIgnoredDuringExecution** and **preferredDuringSchedulingIgnoredDuringExecution** are used.

topologyKey is used to divide topology keys based on the labels of nodes. Nodes with the same labels are grouped into the same topology key. The scheduler then selects the topology key to be scheduled based on the workload label. A topology key can consist of multiple nodes. If a workload that meets a label selection rule runs on a node in a topology key, all nodes in the topology key can be scheduled.

For example, if the **topologyKey** label is set to **topology.kubernetes.io/zone**, nodes' AZs will be used as the topology keys, and workloads will be scheduled by AZ during deployment.

- Workload anti-affinity

  In some cases, pods need to be deployed separately. This is because deploying them together can negatively impact performance.

  Assume that the frontend pods of an application have been created with label **app=frontend**. To ensure that pods are deployed on different nodes and multiple AZs are preferred, you can use **.spec.affinity.podAntiAffinity** to configure workload anti-affinity.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
```



```
  metadata:
    labels:
      app:  frontend
  spec:
    containers:
    - image:  nginx:alpine
      name:  frontend
      resources:
        requests:
          cpu:  100m
          memory:  200Mi
        limits:
          cpu:  100m
          memory:  200Mi
    imagePullSecrets:
    - name: default-secret
    affinity:
      podAntiAffinity:  # Workload anti-affinity scheduling rule
        requiredDuringSchedulingIgnoredDuringExecution:   # Scheduling policy that must be met
        - topologyKey: kubernetes.io/hostname    # Topology keys are divided based on node labels.
          labelSelector:    # Pod label matching rule
            matchExpressions:  # The key of the workload label is app.
            - key: app  # The key of the workload label is app.
              operator: In  # The rule is met if a value exists in the value list.
              values:  # Workload label values
              - frontend
        preferredDuringSchedulingIgnoredDuringExecution:    # Scheduling policy that is met as much
as possible
        - weight: 100  # Priority that can be configured when the best-effort policy is used. The value
ranges from 1 to 100. A larger value indicates a higher priority.
          podAffinityTerm:  # Affinity configuration when the best-effort policy is used
            topologyKey: topology.kubernetes.io/zone    # Topology keys are divided based on node
labels.
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - frontend
```

In the preceding example, anti-affinity rules are configured. The rule that must be met indicates that node topology keys are divided based on **kubernetes.io/hostname**. Nodes with the **kubernetes.io/hostname** label have different label values. Therefore, there is only one node in each topology key. If a topology key contains only one node where a **frontend** pod already exists, pods with the same label will not be scheduled to that topology key. According to the best-effort rule, topology keys are divided based on **topology.kubernetes.io/zone** by node AZ. This ensures that the pods are deployed on nodes in different AZs as much as possible.

NOTE

For workload anti-affinity, when **requiredDuringSchedulingIgnoredDuringExecution** is used, the default access controller **LimitPodHardAntiAffinityTopology** of Kubernetes requires that **topologyKey** can only be **kubernetes.io/hostname**. To use other custom topology logic, modify or disable the access controller.

5.5 Logging In to a Container

Scenario

If you encounter unexpected problems when using a container, you can log in to the container to debug it.

Notes and Constraints

When using CloudShell to access a CCE cluster or container, you can open a maximum of 15 instances simultaneously.

Using CloudShell

NOTICE

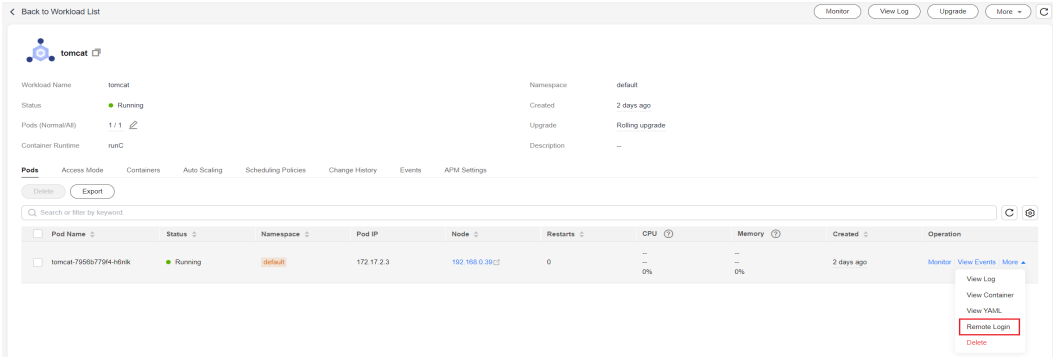
- CloudShell is implemented based on VPC Endpoint (VPCEP). To use kubectl to access a cluster, configure the security group (*Cluster name*-cce-control-*Random number*) on the master node of the cluster to allow access to port 5443. By default, port 5443 allows access from all CIDR blocks. If you have hardened security groups and any cluster cannot be accessed in CloudShell, check whether port 5443 allows access from .
- Using CloudShell to access containers is available only in the following regions: CN North-Beijing1, CN North-Beijing4, CN North-Ulanqab1, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and AP-Singapore.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the name of the target workload to view its pods.

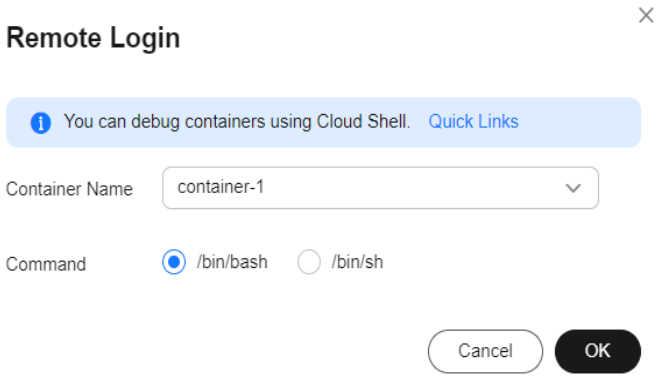
Step 3 Locate the target pod and choose **More** > **Remote Login** in the **Operation** column.

Figure 5-16 Accessing a container



Step 4 In the displayed dialog box, select the container you want to access and the command, and click **OK**.

Figure 5-17 Selecting a container and login command

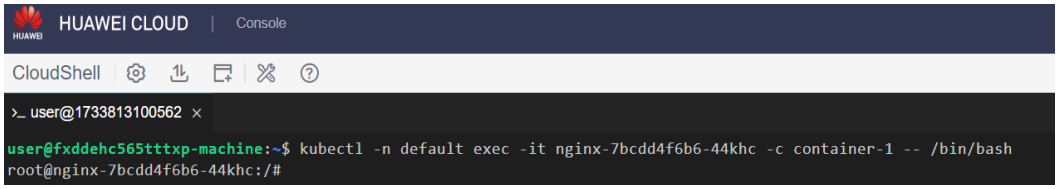


Step 5 You will be automatically redirected to CloudShell. Then, the system initializes kubectl and runs the **kubectl exec** command to log in to the container.

NOTE

Wait for 5 to 10 seconds until the **kubectl exec** command is automatically executed.

Figure 5-18 CloudShell page



----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following command to view the created pod:

```
kubectl get pod
```

The example output is as follows:

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-59d89cb66f-mhljr   1/1     Running   0          11m
```

Step 3 Query the container name in the pod.

```
kubectl get po nginx-59d89cb66f-mhljr -o jsonpath='{range .spec.containers[*]}{.name}{end}{"\n"}'
```

The example output is as follows:

```
container-1
```

Step 4 Run the following command to log in to the **container-1** container in the **nginx-59d89cb66f-mhljr** pod:

```
kubectl exec -it nginx-59d89cb66f-mhljr -c container-1 -- /bin/sh
```

Step 5 To exit the container, run the **exit** command.

----End

5.6 Managing Workloads

Scenario

After a workload is created, you can upgrade, log, monitor, roll back, or delete the workload, as well as edit its YAML file.

Table 5-24 Workload/Job management

Operation	Description
Monitor	You can view the CPU and memory usage of workloads and pods on the CCE console.
View Log	You can view the logs of workloads.
Upgrade	You can replace images or image tags to quickly upgrade Deployments, StatefulSets, and DaemonSets without interrupting services.
Edit YAML	You can modify and download YAML files of Deployments, StatefulSets, DaemonSets, CronJobs, and pods on the CCE console. YAML files of jobs can only be viewed, copied, and downloaded. **NOTE** If an existing CronJob is modified, the new configuration takes effect for the new pods, and the existing pods continue to run without any change.
Roll Back	Only Deployments can be rolled back.
Redeploy	You can redeploy a workload. After the workload is redeployed, all pods in the workload will be restarted.
Enabling/ Disabling the Upgrade	Only Deployments support this operation.
Manage Label	Labels are attached to workloads as key-value pairs to manage and select workloads. Jobs and Cron Jobs do not support this operation.
Delete	You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered.
View Events	You can view event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time.
Stop/Start	You can only start or stop a cron job.

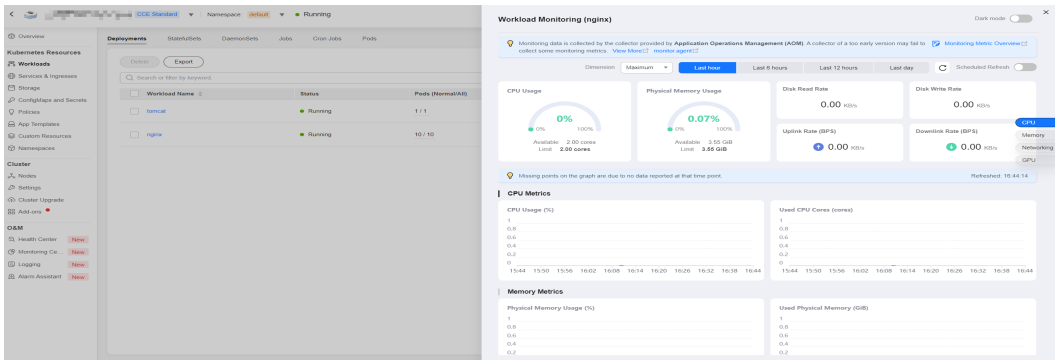
Monitoring a Workload

You can view the CPU and memory usage of Deployments and pods on the CCE console to determine the resource specifications you may need. This section uses a Deployment as an example to describe how to monitor a workload.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and click **Monitor** of the target workload. On the page that is displayed, you can view CPU usage and memory usage of the workload.

Figure 5-19 Viewing monitoring information



Step 3 Click the workload name. On the **Pods** tab page, click the **Monitor** of the target pod to view its CPU and memory usage.

----End

Viewing Logs

You can view logs of Deployments, StatefulSets, DaemonSets, and jobs. This section uses a Deployment as an example to describe how to view logs.

NOTICE

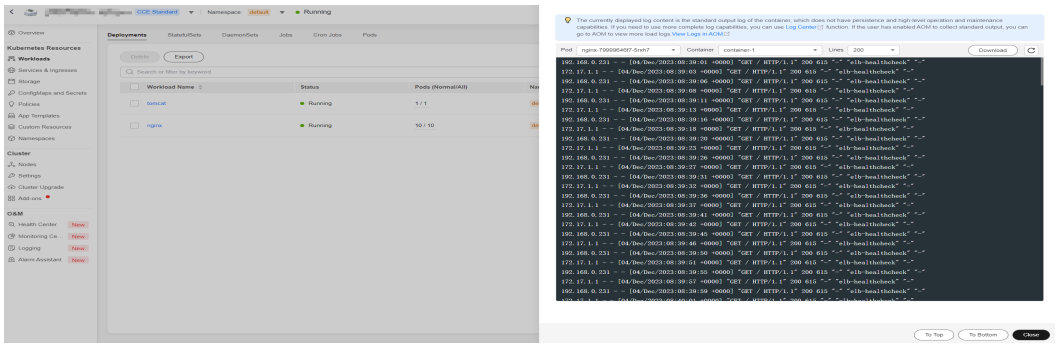
Before viewing logs, ensure that the time of the browser is the same as that on the backend server.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and click **View Log** of the target workload.

In the displayed **View Log** window, you can view logs.

Figure 5-20 Viewing logs of a workload



NOTE

The displayed logs are standard output logs of containers and do not have persistence and advanced O&M capabilities. To use more comprehensive log capabilities, see **Logs**. If the function of collecting standard output is enabled for the workload (enabled by default), you can go to AOM to view more workload logs. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.

----End

Upgrading a Workload

You quickly upgrade Deployments, StatefulSets, and DaemonSets on the CCE console.

This section uses a Deployment as an example to describe how to upgrade a workload.

Before replacing an image or image version, upload the new image to the SWR service. For details, see **Uploading an Image Through a Container Engine Client**.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and click **Upgrade** of the target workload.

NOTE

- Workloads cannot be upgraded in batches.
- Before performing an in-place StatefulSet upgrade, you must manually delete old pods. Otherwise, the upgrade status is always displayed as **Processing**.

Step 3 Upgrade the workload based on service requirements. The method for setting parameter is the same as that for creating a workload.

Step 4 After the update is complete, click **Upgrade Workload**, manually confirm the YAML file, and submit the upgrade.

----End

Editing a YAML file

You can modify and download YAML files of Deployments, StatefulSets, DaemonSets, CronJobs, and pods on the CCE console. YAML files of jobs can only

be viewed, copied, and downloaded. This section uses a Deployment as an example to describe how to edit the YAML file.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and choose **More** > **Edit YAML** in the **Operation** column of the target workload. In the dialog box that is displayed, modify the YAML file.

Step 3 Click **OK**.

Step 4 (Optional) In the **Edit YAML** window, click **Download** to download the YAML file.

----End

Rolling Back a Workload (Available Only for Deployments)

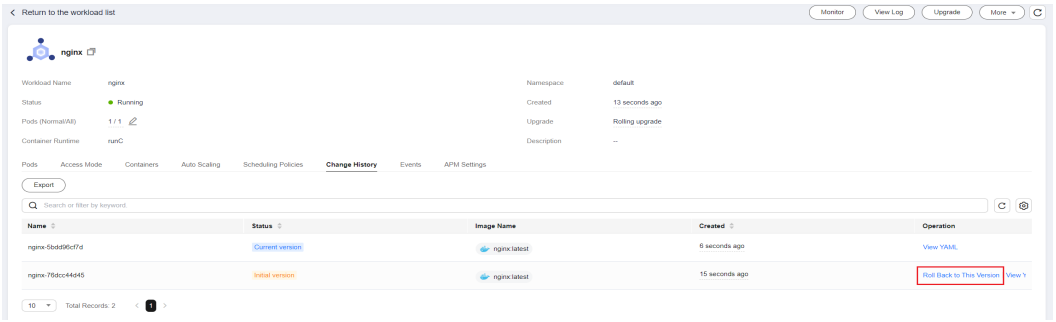
CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and choose **More > Roll Back** in the **Operation** column of the target workload.

Step 3 Switch to the **Change History** tab page, click **Roll Back to This Version** of the target version, manually confirm the YAML file, and click **OK**.

Figure 5-21 Rolling back a workload version



----End

Redeploying a Workload

After you redeploy a workload, all pods in the workload will be restarted. This section uses Deployments as an example to illustrate how to redeploy a workload.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and choose **More** > **Redeploy** in the **Operation** column of the target workload.

Step 3 In the dialog box that is displayed, click **Yes** to redeploy the workload.

----End

Disabling/Enabling Upgrade (Available Only for Deployments)

Only Deployments support this operation.

- After the upgrade is disabled, the upgrade command can be delivered but will not be applied to the pods.

  If you are performing a rolling upgrade, the rolling upgrade stops after the disabling upgrade command is delivered. In this case, the new and old pods co-exist.
- After the upgrade is enabled, a Deployment can be upgraded or rolled back. Its pods will inherit the latest updates of the Deployment. If they are inconsistent, the pods will be upgraded automatically according to the latest information of the Deployment.

NOTICE

Deployments in the disable upgrade state cannot be rolled back.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and choose **More** > **Disable/Enable Upgrade** in the **Operation** column of the workload.

Step 3 In the dialog box that is displayed, click **Yes**.

----End

Managing Labels

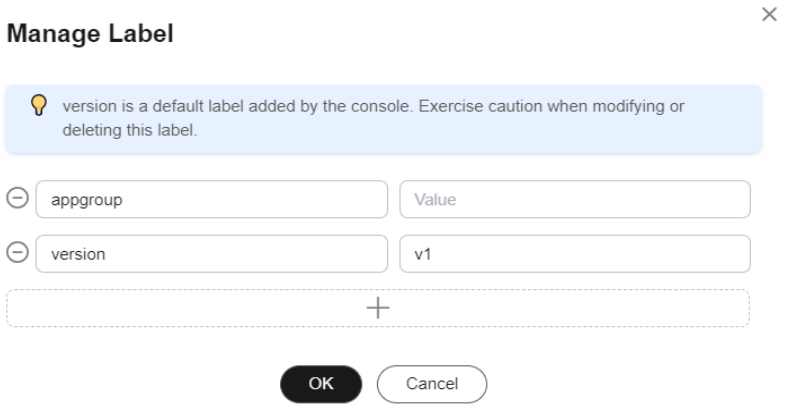
Labels are key-value pairs and can be attached to workloads. You can manage and select workloads by labels. You can add labels to multiple workloads or a specified workload.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab and choose **More** > **Manage Label** in the **Operation** column of the target workload.

Step 3 Click +, enter a key and a value, and click **OK**.

Figure 5-22 Managing labels



NOTE

A key-value pair must contain 1 to 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.

----End

Deleting a Workload/Job

You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered. This section uses a Deployment as an example to describe how to delete a workload.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 In the same row as the workload you will delete, choose **Operation** > **More** > **Delete**.

Read the system prompts carefully. A workload cannot be recovered after it is deleted. Exercise caution when performing this operation.

Step 3 Click **Yes**.

NOTE

- If the node where the pod is located is unavailable or shut down and the workload cannot be deleted, you can forcibly delete the pod from the pod list on the workload details page.
- Ensure that the storage volumes to be deleted are not used by other workloads. If these volumes are imported or have snapshots, you can only unbind them.

----End

Events

This section uses a Deployment as an example to describe how to view events of a workload. To view the event of a job or CronJob, click **View Event** in the **Operation** column of the target workload.

Step 1 Log in to the CCE console, go to an existing cluster, and choose **Workloads** in the navigation pane.

Step 2 On the **Deployments** tab page, click the target workload. In the **Pods** tab page, click the **View Events** to view the event name, event type, number of occurrences, Kubernetes event, first occurrence time, and last occurrence time.

NOTE

Event data will be retained for one hour and then automatically deleted.

----End

5.7 Managing Custom Resources

Custom Resource Definition (CRD) is an extension of Kubernetes APIs. When default Kubernetes resources cannot meet service requirements, you can use CRDs to define new resource types. According to CRD, you can create custom resources in a cluster to meet service requirements. CRD allows you to create new resource types without adding new Kubernetes API servers. This makes cluster management more flexible.

Creating a CRD

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Custom Resources** in the navigation pane, and click the **Create from YAML** in the upper right corner.

Step 3 Customize the YAML file to create a CRD based on service requirements. For details, see **Extend the Kubernetes API with CustomResourceDefinitions**.

Step 4 Click **OK**.

----End

Viewing CRDs and Their Resources

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Custom Resources** in the navigation pane.

Step 3 On the **Custom Resources** page, view CRDs and their resources.

- View a CRD and its YAML.

  All CRDs in the cluster as well as their API groups, API versions, and resource application scopes are listed. Click **View YAML** in the **Operation** column of a CRD to view its YAML.

  You can enter a keyword in the search box to search for target resource types.

- View the resources of a CRD.

  Locate a CDR in the list and click **View Details** in the **Operation** column to view the resources.

----End

5.8 Pod Security

5.8.1 Configuring a Pod Security Policy

A pod security policy (PSP) is a cluster-level resource that controls sensitive security aspects of the pod specification. The **PodSecurityPolicy** object in Kubernetes defines a group of conditions that a pod must comply with to be accepted by the system, as well as the default values of related fields.

By default, the PSP access control component is enabled for clusters of v1.17.17 and a global default PSP named **psp-global** is created. You can modify the default policy (but not delete it). You can also create a PSP and bind it to the RBAC configuration.

NOTE

- In addition to the global default PSP, the system configures independent PSPs for system components in namespace kube-system. Modifying the psp-global configuration does not affect pod creation in namespace kube-system.
- Starting from Kubernetes v1.21, PSP has been deprecated and will be fully removed in v1.25. Therefore, the information in this section applies only to clusters of a version earlier than v1.25. You can use pod security admission as a substitute for PSP. For details, see **Configuring Pod Security Admission**.

Modifying the Global Default PSP

Before modifying the global default PSP, ensure that a CCE cluster has been created and connected by using kubectl.

Step 1 Run the following command:

kubectl edit psp psp-global

Step 2 Modify the required parameters, as shown in **Table 5-25**.

Table 5-25 PSP configuration

Item	Description
privileged	Starts the privileged container.
hostPID hostIPC	Uses the host namespace.
hostNetwork hostPorts	Uses the host network and port.
volumes	Specifies the type of the mounted volume that can be used.

Item	Description
allowedHostPaths	Specifies the host path to which a hostPath volume can be mounted. The **pathPrefix** field specifies the host path prefix group to which a hostPath volume can be mounted.
allowedFlexVolumes	Specifies the FlexVolume driver that can be used.
fsGroup	Configures the supplemental group ID used by the mounted volume in the pod.
readOnlyRootFilesystem	Pods can only be started using a read-only root file system.
runAsUser runAsGroup supplementalGroups	Specifies the user ID, primary group ID, and supplemental group ID for starting containers in a pod.
allowPrivilegeEscalation defaultAllowPrivilegeEsca-lation	Specifies whether **allowPrivilegeEscalation** can be set to **true** in a pod. This configuration controls the use of Setuid and whether programs can use additional privileged system calls.
defaultAddCapabilities requiredDropCapabilities allowedCapabilities	Controls the Linux capabilities used in pods.
seLinux	Controls the configuration of SELinux used in pods.
allowedProcMountTypes	Controls the ProcMountTypes that can be used by pods.
annotations	Configures AppArmor or Seccomp used by containers in a pod.
forbiddenSysctls allowedUnsafeSysctls	Controls the configuration of sysctl used by containers in a pod.

----End

Example of Enabling Unsafe sysctls in a PSP

You can configure allowed-unsafe-sysctls for a node pool. For CCE clusters of **v1.17.17** and later versions, add configurations in **allowedUnsafeSysctls** of the pod security policy to make the configuration take effect. For details, see **Table 5-25**.

In addition to modifying the global pod security policy, you can add new pod security policies. For example, enable the **net.core.somaxconn** unsafe sysctls. The following is an example of adding a pod security policy:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
```



```
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  name: sysctl-psp
spec:
  allowedUnsafeSysctls:
  - net.core.somaxconn
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  fsGroup:
    rule: RunAsAny
  hostIPC: true
  hostNetwork: true
  hostPID: true
  hostPorts:
  - max: 65535
    min: 0
  privileged: true
  runAsGroup:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - '*'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: sysctl-psp
rules:
  - apiGroups:
      - "*"
    resources:
      - podsecuritypolicies
    resourceNames:
      - sysctl-psp
    verbs:
      - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: sysctl-psp
roleRef:
  kind: ClusterRole
  name: sysctl-psp
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

Restoring the Original PSP

If you have modified the default pod security policy and want to restore the original pod security policy, perform the following operations.

Step 1 Create a policy description file named **policy.yaml**. **policy.yaml** is an example file name. You can rename it as required.

vi policy.yaml

The content of the description file is as follows:


```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp-global
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
    - '*'
  volumes:
    - '*'
  hostNetwork: true
  hostPorts:
    - min: 0
      max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'

---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp-global
rules:
  - apiGroups:
      - "*"
    resources:
      - podsecuritypolicies
    resourceNames:
      - psp-global
    verbs:
      - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp-global
roleRef:
  kind: ClusterRole
  name: psp-global
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

Step 2 Run the following command:

kubectl apply -f policy.yaml

----End

5.8.2 Configuring Pod Security Admission

Before using **pod security admission**, understand Kubernetes **Pod Security Standards**. These standards define different isolation levels for pods. They let you define how you want to restrict the behavior of pods in a clear, consistent fashion.

Kubernetes offers a built-in pod security admission controller to enforce the pod security standards. Pod security restrictions are applied at the namespace level when pods are created.

The pod security standard defines three security policy levels:

Table 5-26 Pod security policy levels

Level	Description
privileged	Unrestricted policy, providing the widest possible level of permissions, typically aimed at system- and infrastructure-level workloads managed by privileged, trusted users, such as CNIs and storage drivers.
baseline	Minimally restrictive policy that prevents known privilege escalations, typically targeted at non-critical workloads. This policy disables capabilities such as hostNetwork and hostPID.
restricted	Heavily restricted policy, following current Pod hardening best practices.

Pod security admission is applied at the namespace level. The controller restricts the security context and other parameters in the pod or container in the namespace. The privileged policy does not verify the **securityContext** field of the pod and container. The baseline and restricted policies have different requirements on **securityContext**. For details, see **Pod Security Standards**.

Setting security context: **Configure a Security Context for a Pod or Container**

Pod Security Admission Labels

Kubernetes defines three types of labels for pod security admission (see **Table 5-27**). You can set these labels in a namespace to define the pod security standard level to be used. However, do not change the pod security standard level in system namespaces such as kube-system. Otherwise, pods in the system namespace may be faulty.

Table 5-27 Pod security admission labels

Mode	Target Object	Description
enforce	Pods	Policy violations will cause the pod to be rejected.
audit	Workloads (such as Deployment and job)	Policy violations will trigger the addition of an audit annotation to the event recorded in the audit log, but are otherwise allowed.
warn	Workloads (such as Deployment and job)	Policy violations will trigger a user-facing warning, but are otherwise allowed.

> **NOTE**
>
> Pods are often created indirectly, by creating a workload object such as a Deployment or job. To help catch violations early, both the audit and warning modes are applied to the workload resources. However, the enforce mode is applied only to the resulting pod objects.

Enforcing Pod Security Admission with Namespace Labels

You can label namespaces to enforce pod security standards. Assume that a namespace is configured as follows:

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: privileged
    pod-security.kubernetes.io/enforce-version: v1.25
    pod-security.kubernetes.io/audit: baseline
    pod-security.kubernetes.io/audit-version: v1.25
    pod-security.kubernetes.io/warn: restricted
    pod-security.kubernetes.io/warn-version: v1.25

    # The label can be in either of the following formats:
    # pod-security.kubernetes.io/<MODE>: <LEVEL>
    # pod-security.kubernetes.io/<MODE>-version: <VERSION>
    # The audit and warn modes inform you of which security behaviors are violated by the load.
```

Namespace labels indicate which policy level to apply for the mode. For each mode, there are two labels that determine the policy used:

- pod-security.kubernetes.io/<MODE>: <LEVEL>
  - <MODE>: must be **enforce**, **audit**, or **warn**. For details about the modes, see **Table 5-27**.
  - <LEVEL>: must be **privileged**, **baseline**, or **restricted**. For details about the levels, see **Table 5-26**.
- pod-security.kubernetes.io/<MODE>-version: <VERSION>

  Optional, which pins the policy to a given Kubernetes version.
  - <MODE>: must be **enforce**, **audit**, or **warn**. For details about the modes, see **Table 5-27**.
  - <VERSION>: Kubernetes version number. For example, v1.25. You can also use **latest**.

If pods are deployed in the preceding namespace, the following security restrictions apply:

1. The verification in the enforce mode is skipped (enforce mode + privileged level).
2. Restrictions related to the baseline policy are verified (audit mode + baseline level). That is, if the pod or container violates the policy, the corresponding event is recorded into the audit log.
3. Restrictions related to the restricted policy are verified (warn mode + restricted level). That is, if the pod or container violates the policy, the user will receive an alarm when creating the pod.

Migrating from Pod Security Policy to Pod Security Admission

If you use pod security policies in a cluster earlier than v1.25 and need to replace them with pod security admission in a cluster of v1.25 or later, follow the guide in **Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller**.

NOTICE

1. Pod security admission supports only three isolation modes, less flexible than pod security policies. If you require more control over specific constraints, you will need to use a Validating Admission Webhook to enforce those policies.
2. Pod security admission is a non-mutating admission controller, meaning it will not modify pods before validating them. If you were relying on this aspect of PSP, you will need to either modify the security context in your workloads, or use a Mutating Admission Webhook to make those changes.
3. PSP lets you bind different policies to different service accounts. This approach has many pitfalls and is not recommended, but if you require this feature anyway you will need to use a third-party webhook instead.
4. Do not apply pod security admission to namespaces where CCE components, such as kube-system, kube-public, and kube-node-lease, are deployed. Otherwise, CCE components and add-on functions will be abnormal.

Documentation

- **Pod Security Admission**
- **Mapping PodSecurityPolicies to Pod Security Standards**
- **Enforce Pod Security Standards with Namespace Labels**
- **Enforce Pod Security Standards by Configuring the Built-in Admission Controller**

6 Scheduling

6.1 Overview

CCE supports different types of resource scheduling and task scheduling, improving application performance and overall cluster resource utilization. This section describes the main functions of CPU resource scheduling, GPU/NPU heterogeneous resource scheduling, and Volcano scheduling.

CPU Scheduling

CCE provides CPU policies to allocate complete physical CPU cores to applications, improving application performance and reducing application scheduling latency.

Function	Description	Documentation
CPU policy	When many CPU-intensive pods are running on a node, workloads may be migrated to different CPU cores. Many workloads are not sensitive to this migration and work fine without any intervention. For CPU-sensitive applications, you can use the CPU policy provided by Kubernetes to allocate dedicated cores to applications, improving application performance and reducing application scheduling latency.	**CPU Policy**
Enhanced CPU policy	Based on the Kubernetes static core binding policy, the enhanced CPU policy (enhanced-static) supports burstable pods (whose CPU requests and limits must be positive integers) and allows them to preferentially use certain CPUs to ensure application stability.	**Enhanced CPU Policy**

GPU Scheduling

CCE schedules heterogeneous GPU resources in clusters and allows GPUs to be used in containers.

Function	Description	Documentation
Default GPU scheduling in Kubernetes	This function allows you to specify the number of GPUs that a pod requests. The value can be less than 1 so that multiple pods can share a GPU.	**Default GPU Scheduling in Kubernetes**
GPU virtualization	GPU virtualization dynamically divides the GPU memory and computing power. A single GPU can be virtualized into up to 20 virtual GPU devices. Virtualization is more flexible than static allocation. You can specify the number of GPUs on the basis of stable service running to improve GPU utilization.	**GPU Virtualization**

NPU Scheduling

CCE schedules heterogeneous NPU resources in a cluster to quickly and efficiently perform inference and image recognition.

Function	Description	Documentation
NPU scheduling	NPU scheduling allows you to specify the number of NPUs that a pod requests to provide NPU resources for workloads.	**NPU Scheduling**

Volcano Scheduling

Volcano is a Kubernetes-based batch processing platform that supports machine learning, deep learning, bioinformatics, genomics, and other big data applications. It provides general-purpose, high-performance computing capabilities, such as job scheduling, heterogeneous chip management, and job running management.

Function	Description	Documentation
Resource utilization-based scheduling	Scheduling policies are optimized for computing resources to effectively reduce resource fragments on each node and maximize computing resource utilization.	**Resource Usage-based Scheduling**

Function	Description	Documentation
Priority-based scheduling	Scheduling policies are customized based on service importance and priorities to guarantee the resources of key services.	Priority-based Scheduling
AI performance-based scheduling	Scheduling policies are configured based on the nature and resource usage of AI tasks to increase the throughput of cluster services and improve service performance.	AI Performance-based Scheduling
NUMA affinity scheduling	Volcano targets to lift the limitation to make scheduler NUMA topology aware so that: • Pods are not scheduled to the nodes that NUMA topology does not match. • Pods are scheduled to the most suitable node for NUMA topology.	NUMA Affinity Scheduling

Cloud Native Hybrid Deployment

The cloud native hybrid deployment solution focuses on the Volcano and Kubernetes ecosystems to help users improve resource utilization and efficiency and reduce costs.

Function	Description	Documentation
Dynamic resource oversubscription	Based on the types of online and offline jobs, Volcano scheduling is used to utilize the resources that are requested but not used in the cluster (the difference between the number of requested resources and the number of used resources) for resource oversubscription and hybrid deployment to improve cluster resource utilization.	Dynamic Resource Oversubscription
CPU Burst	CPU Burst is an elastic traffic limiting mechanism that allows temporarily exceeding the CPU limit to reduce the long-tail response time of services and improve the quality of latency-sensitive services.	CPU Burst
Egress network bandwidth guarantee	The egress network bandwidth used by online and offline services is balanced to ensure sufficient network bandwidth for online services.	Guaranteed Egress Network Bandwidth

6.2 CPU Scheduling

6.2.1 CPU Policy

Application Scenarios

By default, kubelet uses **CFS quotas** to enforce pod CPU limits. When a node runs many CPU-bound pods, the workload can move to different CPU cores depending on whether the pod is throttled and which CPU cores are available at scheduling time. Many workloads are not sensitive to this migration and work fine without any intervention. Some applications are CPU-sensitive. They are sensitive to:

- CPU throttling
- Context switching
- Processor cache misses
- Cross-socket memory access
- Hyperthreads that are expected to run on the same physical CPU card

If your workloads are sensitive to any of these items, you can use Kubernetes **CPU management policies** to allocate dedicated CPU cores (through CPU core binding) to these workloads. This will shorten scheduling latency and improve application performance. The CPU manager preferentially allocates resources on a socket and full physical cores to avoid interference.

A CPU management policy is specified by using kubelet *--cpu-manager-policy*. By default, Kubernetes supports the following policies:

- **none**: the default policy. The **none** policy explicitly enables the existing default CPU affinity scheme, providing no affinity beyond what the OS scheduler does automatically.
- **static**: The **static** policy allows containers in **guaranteed pods** with integer CPU requests to be use dedicated CPU resources on nodes through CPU core binding.

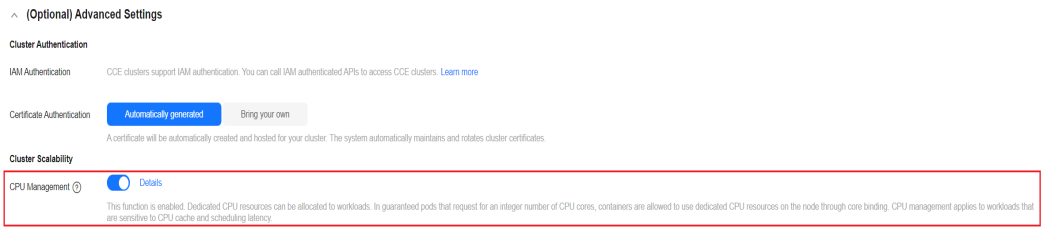
Notes and Constraints

The CPU management policy does not apply to ECS (PM) nodes in CCE Turbo clusters.

Enabling CPU Management for the Default Node Pool

When creating a cluster, you can enable CPU management in **Advanced Settings**.

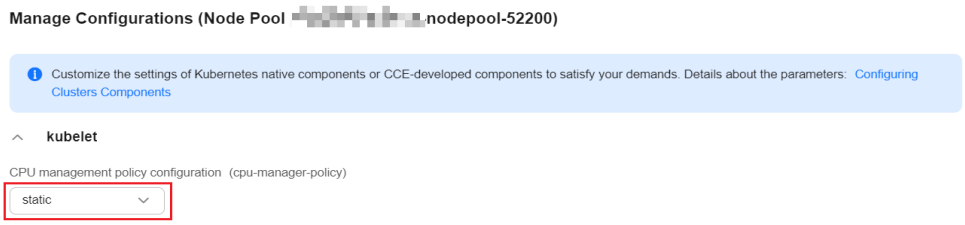
- If this function is enabled, the **static** Kubernetes policy is used, where CPU core binding is used.
- If this function is disabled, the **none** Kubernetes policy is used, where CPU core binding is not used.



Enabling CPU Management for a Custom Node Pool

You can configure a CPU management policy in a custom node pool. After the configuration, the kubelet parameter **--cpu-manager-policy** will be automatically modified on the node in the node pool.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. Choose **Nodes** in the navigation pane and click the **Node Pools** tab on the right. Locate the target node pool and choose **More** > **Manage**.
3. On the **Manage Configurations** page, change the **cpu-manager-policy** value to **static** in the **kubelet** area.



4. Click **OK**.

Allowing Pods to Exclusively Use the CPU Resources

Prerequisites:

- Enable the **static** policy on the node. For details, see **Enabling CPU Management for the Default Node Pool**.
- Both requests and limits must be configured in pods and their values must be the same integer.
- If an init container needs to exclusively use CPUs, set its requests to the same as that of the service container. Otherwise, the service container does not inherit the CPU allocation result of the init container, and the CPU manager reserves more CPU resources than supposed. For more information, see **App Containers can't inherit Init Containers CPUs - CPU Manager Static Policy**.

You can use **node affinity scheduling** to schedule the configured pods to the nodes where the **static** policy is enabled. In this way, the pods can exclusively use the CPU resources.

Example YAML:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: container-1
```



```
      image: nginx:alpine
      resources:
        requests:
          cpu: 2          # The value must be an integer and must be the same as that in limits.
          memory: 2048Mi
        limits:
          cpu: 2          # The value must be an integer and must be the same as that in requests.
          memory: 2048Mi
  imagePullSecrets:
    - name: default-secret
```

Verification

Take a node with 8 vCPUs and 16 GiB of memory as an example. Deploy a workload whose CPU request and limit are both **2** on the node beforehand.

Log in to the node where the workload is running and check the **/var/lib/kubelet/cpu_manager_state** output.

```
cat /var/lib/kubelet/cpu_manager_state
```

Command output:

```
{"policyName":"static","defaultCpuSet":"0-1,4-7","entries":{"de14506d-0408-411f-bbb9-822866b58ae2":{"container-1":"2-3"}},"checksum":3744493798}
```

- If the **policyName** is **static**, the policy has been configured.
- Value **2-3** indicates the set of CPUs that can be used by containers in the pod.

6.2.2 Enhanced CPU Policy

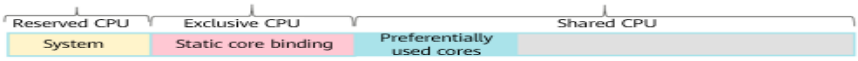
Kubernetes provides two **CPU policies**: none and static.

- **none**: The CPU policy is disabled by default, indicating the existing scheduling behavior.
- **static**: The static CPU core binding policy is enabled. This policy allows pods with certain resource characteristics to be granted enhanced CPU affinity and exclusivity on the node.

Based on the Kubernetes static core binding policy, the enhanced CPU policy (enhanced-static) supports burstable pods (whose CPU requests and limits must be positive integers) and allows them to preferentially use certain CPUs to ensure application stability. Example:

```
...
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "300Mi"
        cpu: "2"
      requests:
        memory: "200Mi"
        cpu: "1"
```

This feature is built on the optimized CPU scheduling in the Huawei Cloud EulerOS 2.0 kernel. When the CPU usage preferentially used by a container exceeds 85%, the container is automatically allocated to other CPUs with low usage to ensure the response capability of applications.



NOTE

- When enhanced CPU policy is enabled, the application performance is better than that of the **none** policy but worse than that of the **static** policy.
- CPU would not be exclusively used by burstable pods, it is still in the shared CPU pool. When the burstable pods are in the low tide, other pods can share this CPU.

Notes and Constraints

To use this feature, the following conditions must be met:

- The cluster version must be v1.23 or later.
- The node OS is Huawei Cloud EulerOS 2.0.
- The CPU management policy does not apply to ECS (PM) nodes in CCE Turbo clusters.

Procedure

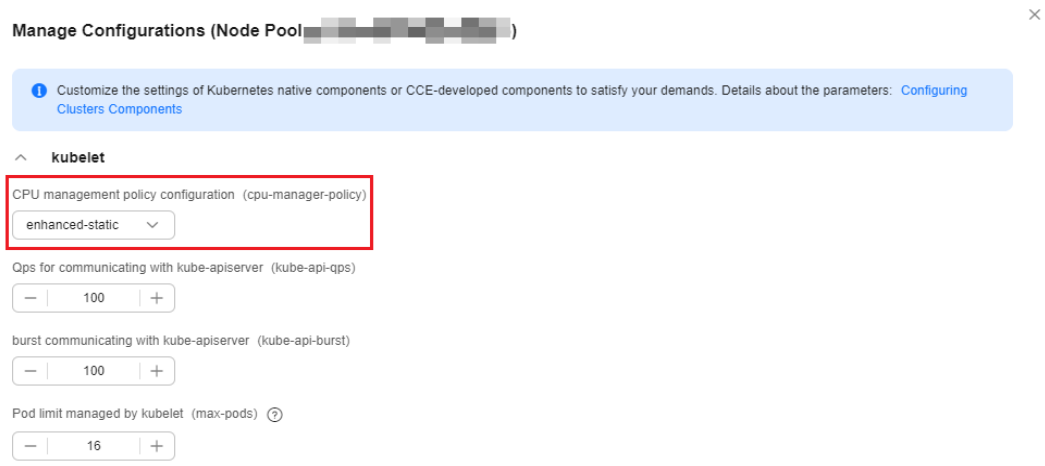
Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Select a node pool whose OS is Huawei Cloud EulerOS 2.0 and click **Manage** in the **Operation** column.

Step 4 On the **Manage Configurations** page, change the **cpu-manager-policy** value to **enhanced-static** in the **kubelet** area.

Figure 6-1 CPU policy



Step 5 Click **OK**.

----End

Verification

Take a node with 8 vCPUs and 32 GiB of memory as an example. Deploy a workload whose CPU request is 1 and limit is 2 in the cluster beforehand.

Step 1 Log in to a node in the node pool and view the **/var/lib/kubelet/cpu_manager_state** output.

```
cat /var/lib/kubelet/cpu_manager_state
```

Command output:

```
{"policyName":"enhanced-static","defaultCpuSet":"0,2-7","entries":{"6739f6f2-ebe5-48ae-945a-986d5d8919b9":{"container-1":"0-7,10001"}},"checksum":1638128523}
```

- If the value of **policyName** is **enhanced-static**, the policy is configured successfully.
- 10000 is used as the base for the CPU ID. In this example, 10001 indicates that the affinity CPU ID used by the container is CPU 1, and 0-7 indicates the set of CPUs that can be used by the container in the pod.

Step 2 Check the cgroup setting of **cpuset.preferred_cpus** of the container. The output is the ID of the CPU that is preferentially used.

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod {pod uid} / {Container ID} /cpuset.preferred_cpus
```

- *{pod uid}* indicates the pod UID, which can be obtained by running the following command on the host that has been connected to the cluster using kubectl:

  ```
  kubectl get po {pod name} -n {namespace} -ojsonpath='{.metadata.uid}{"\n"}'
  ```

  In the preceding command, *{pod name}* and *{namespace}* indicate the pod name and the namespace to which the pod belongs.

- *{Container id}* must be a complete container ID. You can run the following command on the node where the container is running to obtain the container ID:

  Docker node pool: In the command, *{pod name}* indicates the pod name.

  ```
  docker ps --no-trunc | grep {pod name} | grep -v cce-pause | awk '{print $1}'
  ```

  containerd node pool: In the command, *{pod name}* indicates the pod name, *{pod id}* indicates the pod ID, and *{container name}* indicates the container name.

  ```
  # Obtain the pod ID.
  crictl pods | grep {pod name} | awk '{print $1}'
  # Obtain the complete container ID.
  crictl ps --no-trunc | grep {pod id} | grep {container name} | awk '{print $1}'
  ```

A complete example is as follows:

```
cat /sys/fs/cgroup/cpuset/kubepods/burstable/pod6739f6f2-ebe5-48ae-945a-986d5d8919b9/5ba5603434b95fd22d36fba6a5f1c44eba83c18c2e1de9b52ac9b52e93547a13/cpuset.preferred_cpus
```

If the following command output is displayed, CPU 1 is preferentially used.

```
1
```

----End

6.3 GPU Scheduling

6.3.1 GPU Driver Version

6.3.1.1 Selecting a GPU Driver Version for Nodes

Before using GPU-accelerated ECSs, install the necessary NVIDIA infrastructure software to enable accelerated GPU computing. To use GPUs, select and install the appropriate software package that matches the GPU model.

This section describes how to select a driver version and CUDA Toolkit for GPU nodes.

Selecting a GPU Driver Version for Nodes

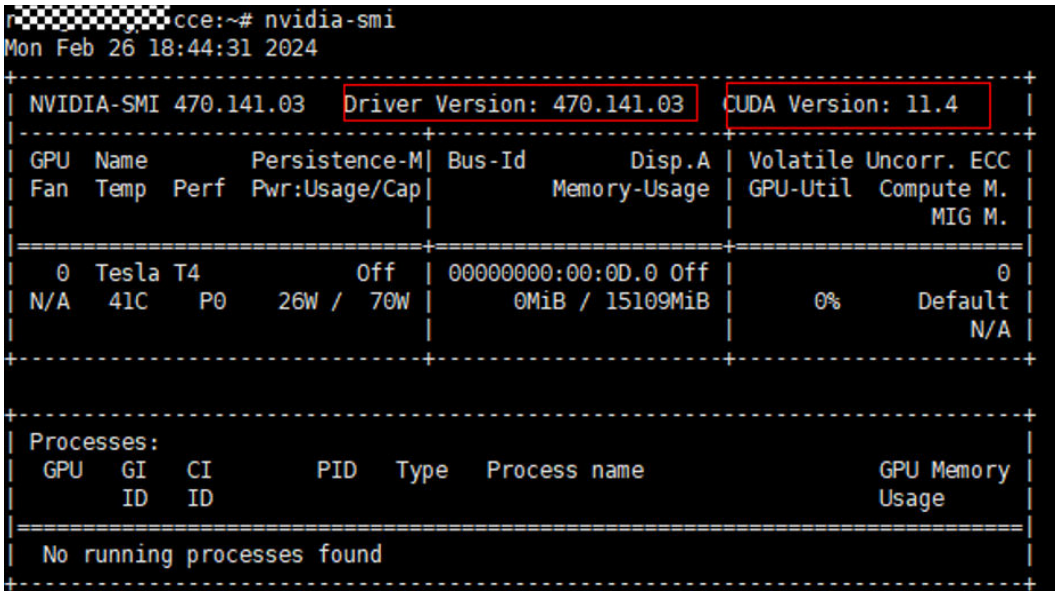
To use GPU resources, you typically need to install the following software packages of the desired version:

- Hardware driver for GPUs, such as a Tesla driver
- Libraries required by upper-layer applications, such as CUDA Toolkit

When using container applications, a node will have a GPU driver installed, and the CUDA Toolkit will be pre-installed during the building of an application's container image. Alternatively, you can use a **NVIDIA base image** that already has the CUDA Toolkit pre-installed to build the application's container image. To use GPU resources, the GPU driver version must match the CUDA Toolkit version.

You can run the **nvidia-smi** command to obtain the driver installed on the node and determine the mapping between the NVIDIA driver and CUDA Toolkit version. In the figure below, the driver version is 470.141.03, and the latest supported CUDA Toolkit version is 11.4.

Figure 6-2 Mapping between the NVIDIA driver and CUDA Toolkit version



Compatibility Between CUDA Toolkit and Driver Versions

When selecting a NVIDIA driver, ensure that the driver version is compatible with the CUDA Toolkit version. The following table lists the **mapping**. The table below

shows the earliest driver versions that are compatible with the CUDA Toolkit. For more detailed version mapping, see **CUDA Toolkit and Corresponding Driver Versions**. You can select a proper NVIDIA driver version based on the CUDA Toolkit version used by your application.

NOTE

If the CUDA and driver versions listed in the table below match, preferentially use the GPU driver provided in **Recommended GPU Driver Versions for CCE**. If the driver version recommended by CCE does not match your CUDA Toolkit version, you will need to use a non-recommended driver version. In such cases, ensure compatibility between the model, OS, and driver version.

CUDA Toolkit Version	Minimum Driver Version Required (Linux x86_64)
CUDA 12.x	≥ 525.60.13
CUDA 11.8.x CUDA 11.7.x CUDA 11.6.x CUDA 11.5.x CUDA 11.4.x CUDA 11.3.x CUDA 11.2.x CUDA 11.1.x	≥ 450.80.02
CUDA 11.0	≥ 450.36.06

6.3.1.2 Recommended GPU Driver Versions for CCE

This section describes the recommended driver versions for CCE clusters. If you use a non-recommended GPU driver version, make sure to check its compatibility with the model and OS. Select a proper NVIDIA driver version by checking the compatibility list for CUDA Toolkit and NVIDIA drivers based on the version used by your application.

Supported GPU Drivers

NOTICE

- The list of supported GPU drivers applies only to GPU add-ons of v1.2.28 and later versions.
- To use the latest GPU driver, upgrade your GPU add-on to the latest version.

Table 6-1 Supported GPU drivers

GPU Model	Supported Cluster Type	Specification	OS							
			Huawei Cloud EulerOS 2.0 (GPU Virtualization Supported)	Ubuntu 22.04.4	Ubuntu 22.04.3	CentOS Linux release 7.6	EulerOS release 2.9	EulerOS release 2.5	Ubuntu 18.04 (end of maintenance)	EulerOS release 2.3 (end of maintenance)
Tesla T4	CCE standard cluster	g6 pi2	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03
Volta V100	CCE standard cluster	p2s p2vs p2v	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03

Helpful Links

- Official NVIDIA drivers
- Version Mapping of Mainstream Tesla Series Drivers

6.3.1.3 Manually Upgrading the Driver Version of a GPU Node

You can use the CCE AI Suite (NVIDIA GPU) add-on to configure the driver file path for a node. After the node is restarted, the driver will be installed automatically. Alternatively, manually upgrade the driver version.

> **NOTE**
>
> Upgrading the driver version of a GPU node manually is a short-term fix for customizing node configurations. However, once the node is restarted, the driver version will revert back to the version specified in the GPU add-on configuration.
>
> For a stable, long-term upgrade of the driver version of a GPU node, see **Upgrading the Driver Version of a GPU Node Using a Node Pool**.

Prerequisites

The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Procedure

To use a specific NVIDIA driver version, perform the following operations to install the GPU driver of the latest version on the node:

Step 1 Take the node offline and manually evict pods on the node to prevent all programs from using GPUs.

1. Take the node offline.
   ```
   kubectl cordon <NODE_NAME>
   ```
   After the node is offline, it becomes unschedulable.
   ```
   NAME          STATUS                     ROLES    AGE   VERSION
   192.168.1.xx  Ready,SchedulingDisabled   <none>   20m   v1.25.5-r20-25.1.31.1
   ```
2. Manually evict pods on the node.
   ```
   kubectl drain 192.168.1.xx --ignore-daemonsets=true --delete-emptydir-data
   ```
   Expected result
   ```
   node/192.168.1.xx drained
   ```
3. Check whether there are any DaemonSet pods are using GPUs. If so, stop kubelet and the runtime.

   Log in to the node where the GPU driver needs to be upgraded, for example, the node with IP address 192.168.1.xx.

   - Stop containerd.
     ```
     systemctl stop kubelet kubelet-monit containerd containerd-monit
     ```
   - Stop Docker.
     ```
     systemctl stop kubelet kubelet-monit docker docker-monit
     ```
4. Check whether there are any programs are using GPUs.

   Run the **sudo fuser -v /dev/nvidia*** command.

   If this command is unavailable (for example, in an RPM-based Linux distribution), run the **yum install psmisc** command to install the Psmisc package.
   ```
   # sudo fuser -v /dev/nvidia*
                        USER        PID ACCESS COMMAND
   /dev/nvidia0:        root      12192 F.... nvidia-gpu-devi
   /dev/nvidiactl:      root      12192 F.... nvidia-gpu-devi
   ```
   Run the **sudo kill 12192** command to delete the process. In this example, process ID **12192** is for reference only.
   ```
   # sudo kill 12192
   # sudo fuser -v /dev/nvidia*      # Recheck whether there are any programs are using GPUs.
   ```

Step 2 Install the NVIDIA driver of a specified version on the node.

1. Download the driver of the specified version at the **official NVIDIA website**. For details about how to select a proper driver version, see **Selecting a GPU Driver Version for Nodes**.
2. Record the driver information of the current version. The command for obtaining a driver varies depending on the CCE AI Suite (NVIDIA GPU) add-on version.
   - Version 1.x.x: **/opt/cloud/cce/nvidia/bin/nvidia-smi**
   - Versions from 2.0.0 to 2.5.3: **/usr/local/nvidia/bin/nvidia-smi**
   - Versions later than 2.5.4: **nvidia-smi**

```
# nvidia-smi
Tue Feb 20 15:06:54 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   31C    P0    23W / 300W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3. Prepare for the installation.

   Obtain the kernel version of the current node and install its dependencies (such as gcc, make, and kernel-devel). The following is an example using a RPM-based Linux distribution (whose OS is EulerOS, HCE, or CentOS).

```
yum install -y perl kernel-devel-$(uname -r) zlib-devel binutils binutils-extra binutils-devel elfutils-libelf-devel gcc make
```

4. Install the downloaded driver and verify the installation.

   The **NVIDIA-Linux-x86_64-535.54.03.run** driver is used as an example.

```
# Obtain the kernel version.
kerVersion=`uname -r`

# Logs for installation details
touch /root/nvidia-installer.log

# Install the driver downloaded at the official NVIDIA website.
sh NVIDIA-Linux-x86_64-535.54.03.run --silent --kernel-source-path=/usr/src/kernels/$kerVersion --log-file-name=/root/nvidia-installer.log --no-install-compat32-libs --utility-prefix="/usr/local/nvidia" --opengl-prefix="/usr/local/nvidia"
```


NOTE

If the following information is displayed during the installation:

```
ERROR: An NVIDIA kernel module 'nvidia' appears to already be loaded in your kernel.  This may be because it is in use (for example, by an X server, a CUDA program, or the NVIDIA Persistence Daemon), but this may also happen if your kernel was configured without support for module unloading.  Please be sure to exit any programs that may be using the GPU(s) before attempting to upgrade your driver.  If no GPU-based programs are running, you know that your kernel supports module unloading, and you still receive this message, then an error may have occurred that has corrupted an NVIDIA kernel module's usage count, for which the simplest remedy is to reboot your computer.
ERROR: Installation has failed.  Please see the file '/root/nvidia-installer.log' for details.  You may find suggestions on fixing installation problems in the README available on the Linux driver download page at www.nvidia.com.
```

Perform the following operations to handle this issue:

1. Disable kubelet and the runtime.
   - Disable containerd.
     ```
     systemctl disable kubelet kubelet-monit containerd containerd-monit
     ```
   - Disable Docker.
     ```
     systemctl disable kubelet kubelet-monit docker docker-monit
     ```
2. Restart the affected node.
3. Run the driver installation command again.
4. Enable kubelet and the runtime.
   - Enable containerd.
     ```
     systemctl enable kubelet kubelet-monit containerd containerd-monit
     ```
   - Enable Docker.
     ```
     systemctl enable kubelet kubelet-monit docker docker-monit
     ```

5. Run the **nvidia-smi** command to check whether the new-version GPU driver has been installed.

```
# nvidia-smi
Tue Feb 20 15:13:58 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.54.03              Driver Version: 535.54.03    CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla V100-SXM2-16GB           Off | 00000000:21:01.0 Off |                    0 |
| N/A   34C    P0              38W / 300W |      0MiB / 16384MiB |      2%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

6. Delete driver files and historical commands.
   ```
   rm NVIDIA-Linux-x86_64-535.54.03.run /root/nvidia-installer.log && history -c
   ```

Step 3 Restart the node.

1. Restart kubelet and the runtime.

- Restart containerd.
  ```
  systemctl start kubelet kubelet-monit containerd containerd-monit
  ```
- Restart Docker.
  ```
  systemctl start kubelet kubelet-monit docker docker-monit
  ```

2. Remove scheduling restrictions on the node.
   ```
   kubectl uncordon <NODE_NAME>
   ```

----End

6.3.1.4 Upgrading the Driver Version of a GPU Node Using a Node Pool

To ensure proper functioning of GPU nodes, upgrade the NVIDIA driver version if it does not match the CUDA library you use. It is recommended that you use node pools to effectively manage node NVIDIA driver versions. This allows you to schedule applications to a specific node pool with a designated driver version. Additionally, when upgrading drivers, you can perform the upgrade in batches by node pool.

NOTICE

When upgrading the NVIDIA driver of a node by node pool, note that the driver will be reinstalled during the node restart. To prevent any issues, ensure that there are no running tasks on the node before upgrading the driver.

Step 1: Specify the Driver Version of a Node Pool

Step 1 Log in to the target node and check its driver version. In this example, the driver version is 510.47.03.

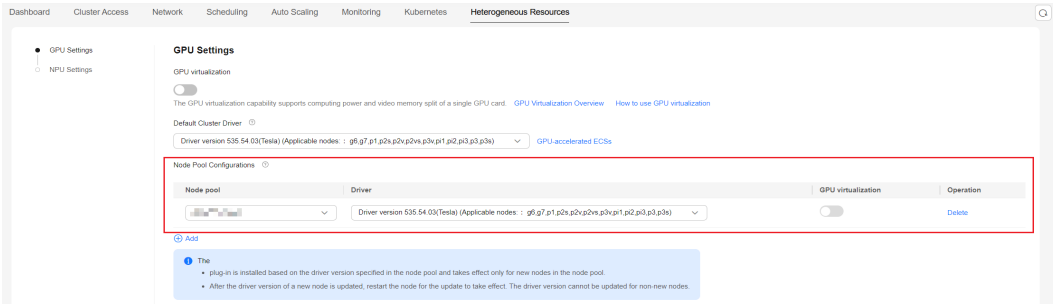
```
# If the add-on version is earlier than 2.0.0, run the following command:
cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi
# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:
cd /usr/local/nvidia/bin && ./nvidia-smi
```

```
[root@test-gpu-nodepool-g8nyo ~]# cd /usr/local/nvidia/bin && ./nvidia-smi
Fri Feb 23 16:00:56 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P4            Off  | 00000000:00:0D.0 Off |                    0 |
| N/A   25C    P8     6W /  75W |      0MiB /  7680MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[root@test-gpu-nodepool-g8nyo bin]#
```

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings**.

Step 3 Click the **Heterogeneous Resources** tab page. In the **Node Node Pool Configurations** pane, select the target node pool and driver, or enter the link to the custom driver.

In this section, the driver will be upgraded to 535.54.03.



Step 4 Click **Confirm Configuration**.

----End

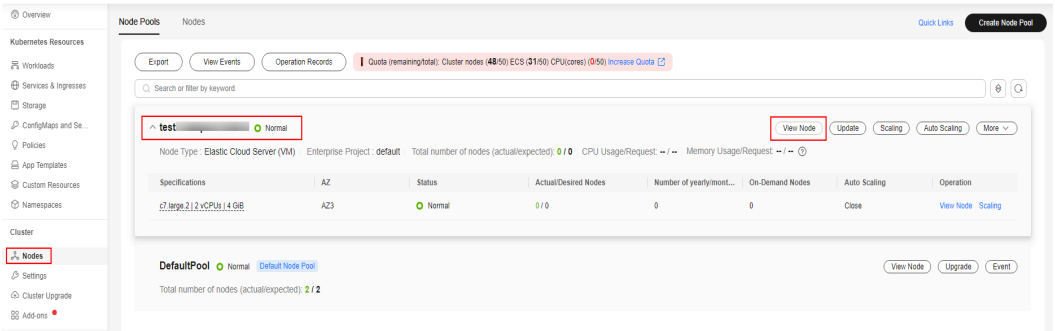
Step 2: Restart the Nodes in the Node Pool

NOTICE

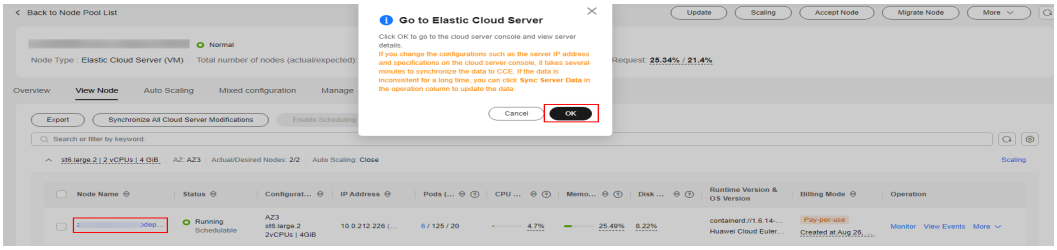
Evict pods on a node before restarting the node. For details, see **Draining a Node**. Make sure to reserve GPU resources to avoid pod scheduling failure during node drainage. Insufficient resources can affect service running.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

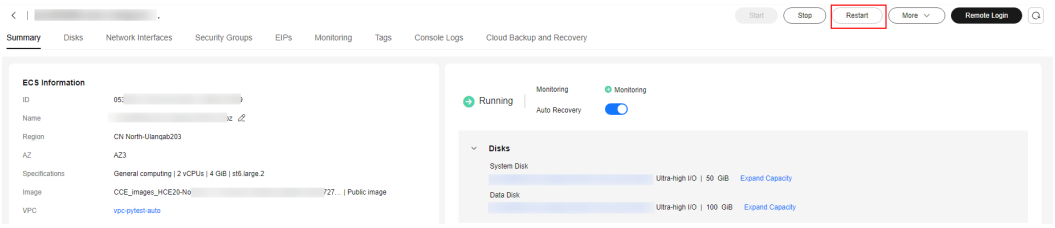
Step 2 Choose **Nodes**, locate the target node pool, and click **View Node**.



Step 3 Click the node name and navigate to the ECS page.



Step 4 In the upper right corner, click **Restart**.



----End

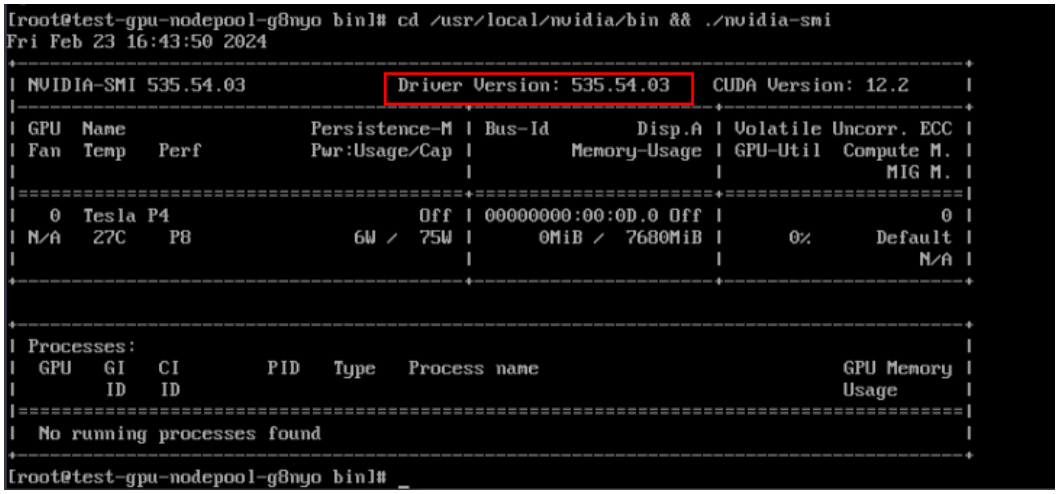
Step 3: Verify the Driver Upgrade

Step 1 Wait a few minutes after restarting the node for the driver to install.

Step 2 Log in to the node and check whether the driver on the node has been updated.

```
# If the add-on version is earlier than 2.0.0, run the following command:
 cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi
# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:
 cd /usr/local/nvidia/bin && ./nvidia-smi
```

Verify the driver version on the node. The figure shows an updated version of 535.54.03.



Step 3 Confirm that the node and its services are running correctly. Then, perform the same operations on the remaining nodes in the node pool individually.

----End

6.3.2 Default GPU Scheduling in Kubernetes

You can use GPUs in CCE containers.

Prerequisites

- A GPU node has been created. For details, see **Creating a Node**.
- The CCE AI Suite (NVIDIA GPU) add-on has been installed. During the installation, select the driver corresponding to the GPU model on the node. For details, see **CCE AI Suite (NVIDIA GPU)**.

- When the default GPU scheduling is used in clusters of v1.27 or earlier, the CCE AI Suite (NVIDIA GPU) add-on mounts the driver directory to **/usr/local/nvidia/lib64**, so you need to add **/usr/local/nvidia/lib64** to the **LD_LIBRARY_PATH** environment variable to use GPUs in a container. You can skip this step for clusters of v1.28 or later.

  You can add environment variables in any of the following ways:

  - Configure the **LD_LIBRARY_PATH** environment variable in the Dockerfile used for creating an image. (Recommended)
    ```
    ENV LD_LIBRARY_PATH /usr/local/nvidia/lib64:$LD_LIBRARY_PATH
    ```
  - Configure the **LD_LIBRARY_PATH** environment variable in the image startup command.
    ```
    /bin/bash -c "export LD_LIBRARY_PATH=/usr/local/nvidia/lib64:$LD_LIBRARY_PATH && ..."
    ```
  - Define the **LD_LIBRARY_PATH** environment variable when creating a workload. (Ensure that this variable is not configured in the container. Otherwise, it will be overwritten.)
    ```
    ...
            env:
              - name: LD_LIBRARY_PATH
                value: /usr/local/nvidia/lib64
    ...
    ```

Using GPUs

Create a workload and request GPUs. You can specify the number of GPUs as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 1   # Maximum number of GPUs that can be used
      imagePullSecrets:
      - name: default-secret
```

nvidia.com/gpu specifies the number of GPUs to be requested. The value can be smaller than **1**. For example, **nvidia.com/gpu: 0.5** indicates that multiple pods share a GPU. In this case, all the requested GPU resources come from the same GPU card.

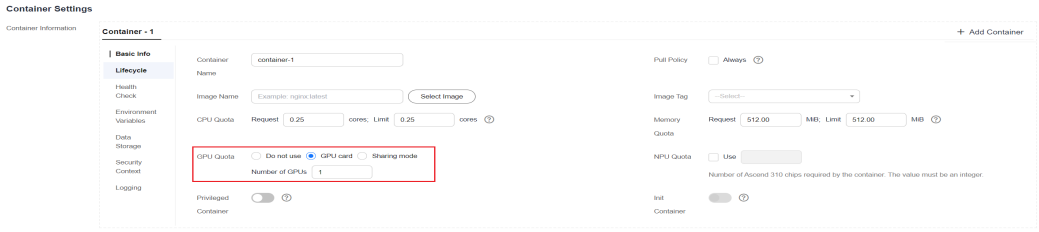
> **NOTE**
>
> When you use **nvidia.com/gpu** to specify the number of GPUs, the values of requests and limits must be the same.

After **nvidia.com/gpu** is specified, workloads will not be scheduled to nodes without GPUs. If the node is GPU-starved, Kubernetes events similar to the following are reported:

- 0/2 nodes are available: 2 Insufficient nvidia.com/gpu.
- 0/4 nodes are available: 1 InsufficientResourceOnSingleGPU, 3 Insufficient nvidia.com/gpu.

To use GPU resources on the CCE console, you only need to configure the GPU quota when creating a workload.

Figure 6-3 GPU quota



GPU Node Labels

CCE will label GPU-enabled nodes after they are created. Different types of GPU-enabled nodes have different labels.

```
$ kubectl get node -L accelerator
NAME          STATUS   ROLES    AGE     VERSION                                ACCELERATOR
10.100.2.179  Ready    <none>   8m43s   v1.19.10-r0-CCE21.11.1.B006-21.11.1.B006   nvidia-t4
```

When using GPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-test
  template:
    metadata:
      labels:
        app: gpu-test
    spec:
      nodeSelector:
        accelerator: nvidia-t4
      containers:
      - image: nginx:perl
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
```



```
        nvidia.com/gpu: 1   # Number of requested GPUs
      limits:
        cpu: 250m
        memory: 512Mi
        nvidia.com/gpu: 1   # Maximum number of GPUs that can be used
  imagePullSecrets:
  - name: default-secret
```

6.3.3 GPU Virtualization

6.3.3.1 Overview

CCE uses xGPU virtualization technologies to dynamically divide the GPU memory and computing power. A single GPU can be virtualized into up to 20 virtual GPU devices. Virtualization is more flexible than static allocation. You can specify the number of GPUs on the basis of stable service running to improve GPU utilization.

Advantages

The GPU virtualization function of CCE has the following advantages:

- **Flexible**: The GPU computing ratio and memory size are finely tuned. The allocation granularity for computing is 5% of GPUs, while the memory allocation is in MiB.
- **Isolated**: The memory of a single GPU can be isolated, and both GPU computing and memory can be isolated concurrently.
- **Compatible**: There is no need to recompile services or replace the CUDA library.

Prerequisites

Item	Supported Version
Cluster version	v1.23.8-r0, v1.25.3-r0, or later
OS	Huawei Cloud EulerOS 2.0
GPU type	T4 and V100
Driver version	470.57.02, 510.47.03, and 535.54.03
Runtime	containerd
Add-on	The following add-ons must be installed in the cluster: • **Volcano Scheduler**: 1.10.5 or later • **CCE AI Suite (NVIDIA GPU)**: 2.0.5 or later

Notes and Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- xGPUs cannot be used in init containers.

- GPU virtualization supports two isolation modes: GPU memory isolation and isolation between GPU memory and computing power. A single GPU can schedule only workloads in the same isolation mode.
- Autoscaler cannot be used to automatically scale in or out GPU virtualization nodes in clusters of v1.27 or earlier.
- xGPU isolation does not allow you to request for GPU memory by calling CUDA API cudaMallocManaged(), which is also known as using UVM. For more information, see **NVIDIA official documents**. Use other methods to request for GPU memory, for example, by calling cudaMalloc().
- When a containerized application is initializing, the real-time compute monitored by the nvidia-smi may exceed the upper limit of the available compute of the container.
- When GPU virtualization is enabled on a node with multiple GPUs, insufficient GPU resources will not result in the preempting of GPU resources from other pods.

6.3.3.2 Preparing xGPU Resources

CCE uses xGPU virtualization technologies to dynamically divide the GPU memory and computing power. A single GPU can be virtualized into up to 20 virtual GPU devices. This section describes how to implement GPU scheduling and isolation capabilities on GPU nodes.

Prerequisites

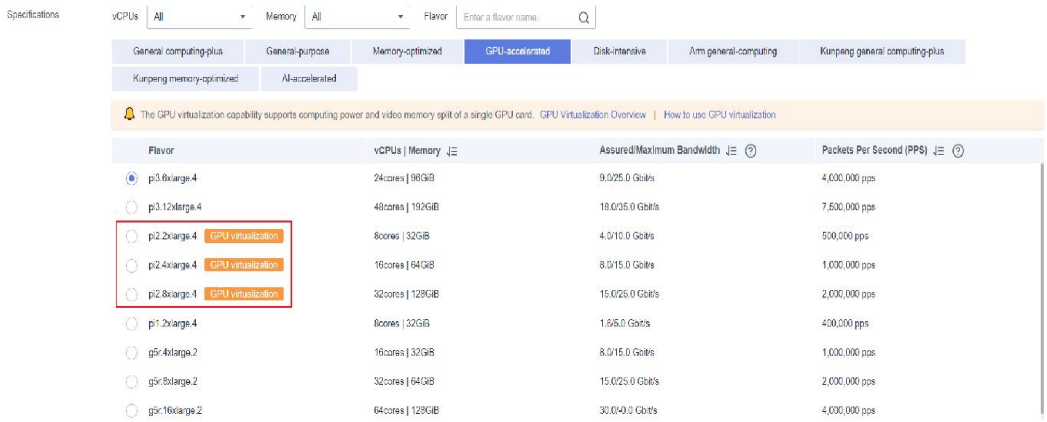
Item	Supported Version
Cluster version	v1.23.8-r0, v1.25.3-r0, or later
OS	Huawei Cloud EulerOS 2.0
GPU type	T4 and V100
Driver version	470.57.02, 510.47.03, and 535.54.03
Runtime	containerd
Add-on	The following add-ons must be installed in the cluster: • **Volcano Scheduler**: 1.10.5 or later • **CCE AI Suite (NVIDIA GPU)**: 2.0.5 or later

Step 1: Enable GPU Virtualization

Both **CCE AI Suite (NVIDIA GPU)** and **Volcano Scheduler** must be installed in the cluster.

Step 2: Create a GPU Node

Create nodes that support GPU virtualization in the cluster to use the GPU virtualization function. For details, see **Creating a Node** or **Creating a Node Pool**.



NOTE

If your cluster already has GPU nodes that meet the **Prerequisites**, skip this step.

Step 3 (Optional): Modifying the Volcano Scheduling Policy

The default scheduling policy of Volcano for GPU nodes is **Spread**. If the node configurations are the same, Volcano selects the node with the minimum number of running containers, so that containers can be evenly allocated to each node. In contrast, the bin packing policy attempts to schedule all containers to one node to avoid resource fragmentation.

If the bin packing policy is required when the GPU virtualization feature is used, you can modify the policy in the advanced settings of the Volcano add-on. The procedure is as follows:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings**.

Step 2 On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.

Step 3 Modify the Volcano scheduling configuration.

1. In the nodeorder add-on, add the **arguments** parameter and set **leastrequested.weight** to **0**. That is, set the priority of the node with the fewest allocated resources to **0**.
2. Add the bin packing add-on, and specify the weights of xGPU customized resources (**volcano.sh/gpu-core.percentage** and **volcano.sh/gpu-mem.128Mi**).

Example:

```
...
default_scheduler_conf:
```



```
actions: allocate, backfill, preempt
tiers:
  - plugins:
      - name: priority
      - enablePreemptable: false
        name: gang
      - name: conformance
  - plugins:
      - enablePreemptable: false
        name: drf
      - name: predicates
      - name: nodeorder
        arguments:
          leastrequested.weight: 0 # Set the priority of the node with the least allocated resources to 0.
  - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: xgpu
      - name: binpack # Add the bin packing add-on and specify the weights of xGPU resources.
        arguments:
          binpack.resources: volcano.sh/gpu-core.percentage,volcano.sh/gpu-mem.128Mi
          binpack.resources.volcano.sh/gpu-mem.128Mi: 10
          binpack.resources.volcano.sh/gpu-core.percentage: 10
  - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
...
```

Step 4 Click **Save**.

----End

6.3.3.3 Using GPU Virtualization

This section describes how to use the GPU virtualization capability to isolate the computing power from the GPU memory and efficiently use GPU device resources.

Prerequisites

- You have **prepared GPU virtualization resources**.
- If you want to create a cluster using commands, use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- xGPUs cannot be used in init containers.
- GPU virtualization supports two isolation modes: GPU memory isolation and isolation between GPU memory and computing power. A single GPU can schedule only workloads in the same isolation mode.
- Autoscaler cannot be used to automatically scale in or out GPU virtualization nodes in clusters of v1.27 or earlier.
- xGPU isolation does not allow you to request for GPU memory by calling CUDA API cudaMallocManaged(), which is also known as using UVM. For more information, see **NVIDIA official documents**. Use other methods to request for GPU memory, for example, by calling cudaMalloc().

- When a containerized application is initializing, the real-time compute monitored by the nvidia-smi may exceed the upper limit of the available compute of the container.
- When GPU virtualization is enabled on a node with multiple GPUs, insufficient GPU resources will not result in the preempting of GPU resources from other pods.

Creating a GPU Virtualization Application

Using the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

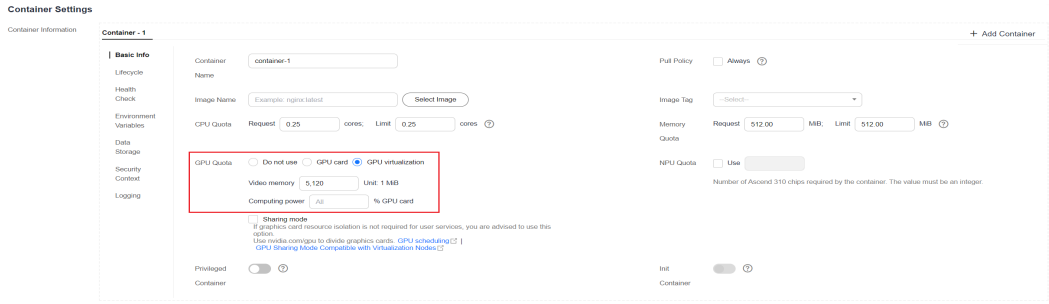
Choose **Container Settings** > **Basic Info** and configure the GPU quota.

- **Video memory**: The unit is MiB. The value must be a positive integer that is a multiple of 128. If the value exceeds the memory of a single GPU, scheduling cannot be performed.
- **Computing power**: The unit is %. The value must be a multiple of 5 and cannot exceed 100.

NOTE

- If the GPU memory is set to the capacity upper limit of a single GPU or the computing power is set to 100%, the entire GPU will be used.
- When GPU virtualization is used, the workload scheduler defaults to Volcano and cannot be changed.

Figure 6-4 Configuring the xGPU quota



This section describes how to use GPU virtualization. For details about other parameters, see **Workloads**.

After completing the setting, click **Create**.

Step 4 After a workload is created, you can try to verify the isolation capability of GPU virtualization.

1. Log in to the pod and check the total GPU memory allocated to the pod.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:


```
Wed Apr 12 07:54:59 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4912MiB /  5120MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The expected output indicates that the total GPU memory allocated to the pod is 5120 MiB, and 4912 MiB is used.

2. Run the following command on the node to check the isolation status of the GPU memory:

```
nvidia-smi
```

Expected output:

```
Wed Apr 12 09:31:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4957MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                          4835MiB |
+-----------------------------------------------------------------------------+
```

The expected output indicates that the total GPU memory on the node is 16160 MiB, and the example pod uses 4957 MiB.

----End

Using kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create an application that uses GPU virtualization.

NOTE

The GPU memory isolation and isolation between GPU memory and computing power are supported. The computing power cannot be isolated only. **volcano.sh** and **gpu-core.percentage** cannot be set separately.

Create a **gpu-app.yaml** file with the following content.

- Isolate GPU memory only:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  labels:
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
      - name: container-1
        image: <your_image_address>     # Replace it with your image address.
        resources:
          limits:
            volcano.sh/gpu-mem.128Mi: 40  # GPU memory allocated to the pod. The value is a multiple of 128 MiB (40 x 128 = 5120 MiB).
      imagePullSecrets:
        - name: default-secret
      schedulerName: volcano
```

- Isolate the GPU memory from computing power:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  labels:
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
      - name: container-1
        image: <your_image_address>     # Replace it with your image address.
        resources:
          limits:
            volcano.sh/gpu-mem.128Mi: 40  # GPU memory allocated to the pod. The value is a multiple of 128 MiB (40 x 128 = 5120 MiB).
            volcano.sh/gpu-core.percentage: 25    # Computing power allocated to the pod
      imagePullSecrets:
        - name: default-secret
      schedulerName: volcano
```

Table 6-2 Key parameters

Parameter	Mandatory	Description
volcano.sh/gpu-mem.128Mi	No	The value is a positive integer that is a multiple of 128 in the unit of MiB. If the value exceeds the memory of a single GPU, scheduling cannot be performed.

Parameter	Mandatory	Description
volcano.sh/gpu-core.percentage	No	The unit of the computing power is %. The value must be a multiple of 5 and cannot exceed 100.

NOTE

- If the GPU memory is set to the capacity upper limit of a single GPU or the computing power is set to 100%, the entire GPU will be used.
- When GPU virtualization is used, the workload scheduler defaults to Volcano and cannot be changed.

Step 3 Run the following command to create an application:

```
kubectl apply -f gpu-app.yaml
```

Step 4 Verify the isolation capability of GPU virtualization.

1. Log in to the pod and check the total GPU memory allocated to the pod.

   ```
   kubectl exec -it gpu-app -- nvidia-smi
   ```

   Expected output:

   ```
   Wed Apr 12 07:54:59 2023
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |                               |                      |               MIG M. |
   |===============================+======================+======================|
   |   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
   | N/A   27C    P0    37W / 300W |   4912MiB /  5120MiB |      0%      Default |
   |                               |                      |                  N/A |
   +-------------------------------+----------------------+----------------------+

   +-----------------------------------------------------------------------------+
   | Processes:                                                                  |
   |  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
   |        ID   ID                                                   Usage      |
   |=============================================================================|
   +-----------------------------------------------------------------------------+
   ```

   The expected output indicates that the total GPU memory allocated to the pod is 5120 MiB, and 4912 MiB is used.

2. Run the following command on the node to check the isolation status of the GPU memory:

   ```
   nvidia-smi
   ```

   Expected output:

   ```
   Wed Apr 12 09:31:10 2023
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |                               |                      |               MIG M. |
   |===============================+======================+======================|
   |   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
   | N/A   27C    P0    37W / 300W |   4957MiB / 16160MiB |      0%      Default |
   ```



```
|                               |                      |               N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                           4835MiB |
+-----------------------------------------------------------------------------+
```

The expected output indicates that the total GPU memory on the node is 16160 MiB, and the example pod uses 4957 MiB.

----End

6.3.3.4 Supporting Kubernetes' Default GPU Scheduling

After GPU virtualization is enabled, you are advised to configure **volcano.sh/gpu-mem.128Mi** for GPU memory isolation and both **volcano.sh/gpu-mem.128Mi** and **volcano.sh/gpu-core.percentage** for compute-GPU memory isolation when scheduling GPUs for workloads. **Kubernetes' default GPU scheduling** is still available, allowing your workloads to use **nvidia.com/gpu** resources.

- Setting **nvidia.com/gpu** to a decimal fraction (for example, **0.5**) allows GPU virtualization to allocate the specified **nvidia.com/gpu** resources for GPU memory isolation in workloads. Containers will receive GPU memory based on the specified value, such as 8 GiB (0.5 x 16 GiB). The GPU memory value must be a multiple of 128 MiB, or it will be rounded down to the nearest integer automatically. If **nvidia.com/gpu** is set to an integer, the entire available GPUs will be used. If **nvidia.com/gpu** resources have been used in the workload before GPU virtualization is enabled, the resources will be from the entire GPUs but not GPU virtualization.
- When GPU virtualization is enabled, configuring **nvidia.com/gpu** for a workload enables GPU memory isolation. This allows the workload to share the same GPU with other workloads using **volcano.sh/gpu-mem.128Mi** resources. However, it cannot share the GPU with workloads that use both **volcano.sh/gpu-mem.128Mi** and **volcano.sh/gpu-core.percentage** resources. In addition, **Notes and Constraints** on GPU virtualization must be followed.

Notes and Constraints

To support Kubernetes' default GPU scheduling on GPU nodes, the CCE AI Suite (NVIDIA GPU) add-on must be of v2.0.10 or later, and the Volcano Scheduler add-on must be of v1.10.5 or later.

Configuration Example

Step 1 Use kubectl to access the cluster.

Step 2 Create a workload that uses nvidia.com/gpu resources.

Create a **gpu-app.yaml** file. The following shows an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  namespace: default
```



```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      schedulerName: volcano
      containers:
        image: <your_image_address>     # Replace it with your image address.
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 0.1   # Number of requested GPUs
          limits:
            cpu: 250m
            memory: 512Mi
            nvidia.com/gpu: 0.1   # Maximum number of GPUs that can be used
      imagePullSecrets:
      - name: default-secret
```

Step 3 Run the following command to create an application:

```
kubectl apply -f gpu-app.yaml
```

Step 4 Log in to the pod and check the total GPU memory allocated to the pod.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Thu Jul 27 07:53:49 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A30          Off  | 00000000:00:0D.0 Off |                    0 |
| N/A   47C    P0    34W / 165W |      0MiB /  2304MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The output shows that the total GPU memory that can be used by the pod is 2304 MiB.

In this example, the total GPU memory on the GPU node is 24258 MiB, but the number 2425.8 (24258 × 0.1) is not an integer multiple of 128 MiB. Therefore, the value 2425.8 is rounded down to 18 times of 128 MiB (18 × 128 MiB = 2304 MiB).

----End

6.3.4 Monitoring GPU Metrics

You can use Prometheus and Grafana to observe GPU metrics. This section uses Prometheus as an example to describe how to view the GPU memory usage of a cluster.

The process is as follows:

1. **Accessing Prometheus**

   (Optional) Bind a LoadBalancer Service to Prometheus so that Prometheus can be accessed from external networks.

2. **Monitoring GPU Metrics**

   After a GPU workload is deployed in the cluster, GPU metrics will be automatically reported.

3. **Accessing Grafana**

   View Prometheus monitoring data on Grafana, a visualization panel.

Prerequisites

- The **Cloud Native Cluster Monitoring** add-on has been installed in the cluster.
- The **CCE AI Suite (NVIDIA GPU)** add-on has been installed in the cluster, and the add-on version is 2.0.10 or later.
- To monitor GPU virtualization metrics, ensure **Volcano Scheduler** has been installed in the cluster and the add-on version is 1.10.5 or later.

Accessing Prometheus

After the Prometheus add-on is installed, you can deploy workloads and Services. The Prometheus server will be deployed as a StatefulSet in the **monitoring** namespace.

You can create a public network **LoadBalancer Service** so that Prometheus can be accessed from an external network.

Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

Step 2 Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service.

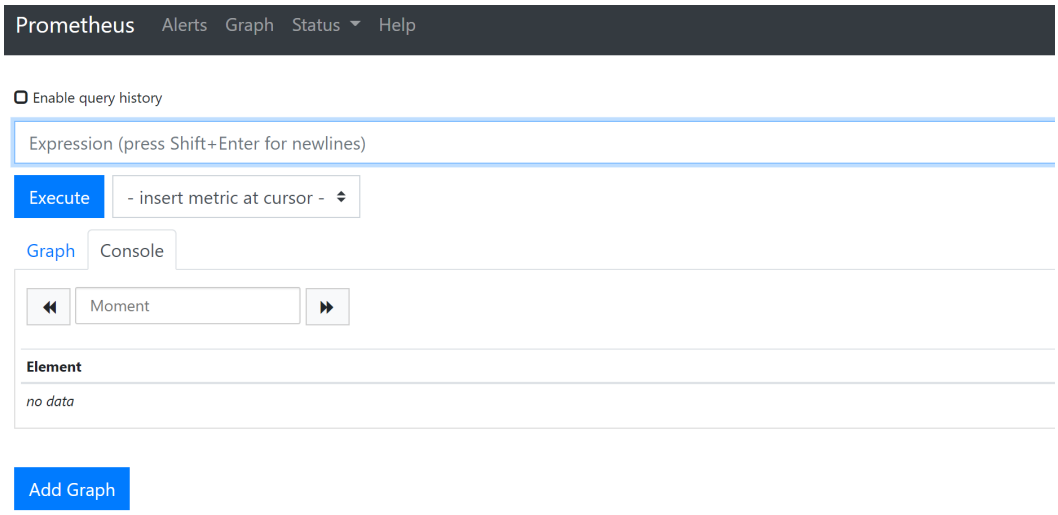
```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb     # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace it with the ID of the public network load balancer in the VPC that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
```



```
    port: 88              # Service port, which is customizable.
    targetPort: 9090      # Default Prometheus port. Retain the default value.
  selector:               # The label selector can be adjusted based on the label of a Prometheus server
instance.
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

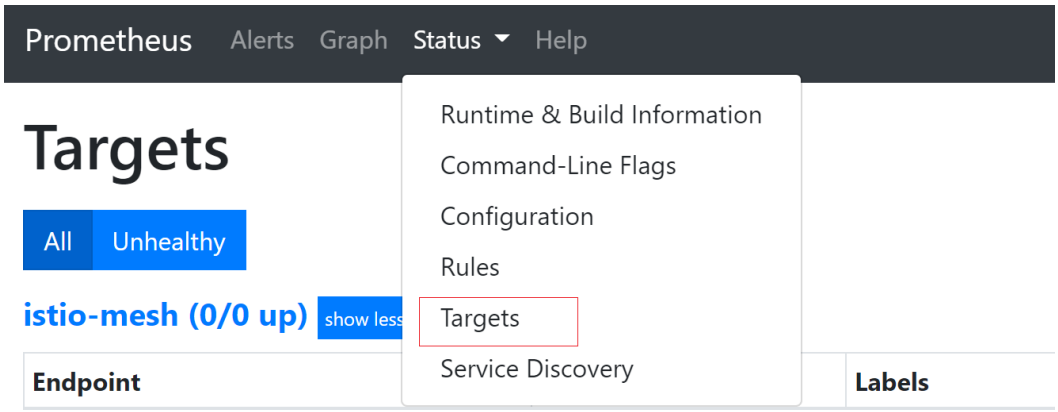
Step 3 After the Service is created, visit ***Public IP address of the load balancer:Service port*** to access Prometheus.

Figure 6-5 Accessing Prometheus



Step 4 Choose **Status** > **Targets** to view the targets monitored by Prometheus.

Figure 6-6 Viewing monitored targets



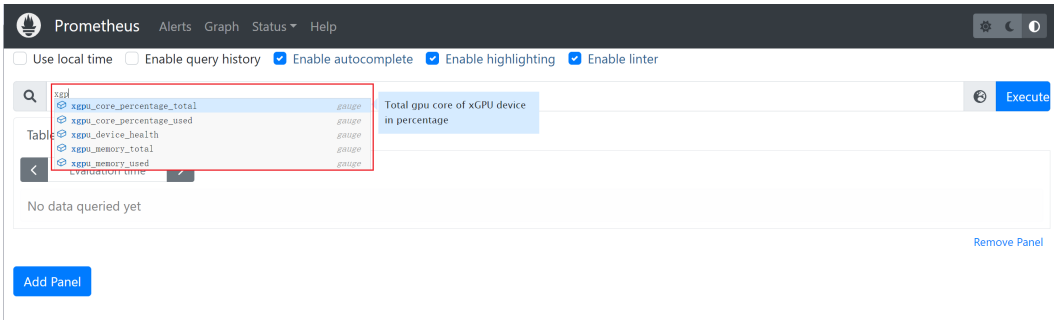
----End

Monitoring GPU Metrics

Create a GPU workload. After the workload runs properly, access Prometheus and view GPU metrics on the **Graph** page.

For more details, see **GPU Metrics**.

Figure 6-7 Viewing GPU metrics



Accessing Grafana

The Prometheus add-on has had **Grafana** (an open-source visualization tool) installed and interconnected. You can create a public network **LoadBalancer Service** so that you can access Grafana from the public network and view Prometheus monitoring data on Grafana.

Click the access address to access Grafana and select a proper dashboard to view the aggregated content.

Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

Step 2 Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service for Grafana.

```
apiVersion: v1
kind: Service
metadata:
  name: grafana-lb     # Service name, which is customizable
  namespace: monitoring
  labels:
    app: grafana
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace it with the ID of the public network load balancer in the VPC to which the cluster belongs.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80     # Service port, which is customizable
      targetPort: 3000     # Default Grafana port. Retain the default value.
  selector:
    app: grafana
  type: LoadBalancer
```

Step 3 After the Service is created, visit ***Public IP address of the load balancer:Service port*** to access Grafana and select a proper dashboard to view xGPU resources.

Figure 6-8 Viewing xGPU resources

----End

6.3.5 Configuring NVIDIA GPU to Use DCGM-Exporter for GPU Metric Monitoring

It is essential for O&M personnel to monitor large-scale Kubernetes-based GPU devices. By tracking GPU metrics, the O&M personnel can gain valuable insights into the overall utilization, health status, and workload performance of the entire cluster, thereby facilitating swift issue resolution, optimized GPU resource allocation, and enhanced resource efficiency. Moreover, data scientists and AI algorithm engineers can leverage relevant monitoring metrics to learn about the organization's GPU usage patterns, ultimately supporting informed capacity planning and task scheduling decisions.

The latest NVIDIA offerings enable the utilization of the Data Center GPU Manager (DCGM) for managing large-scale GPU clusters. The CCE AI Suite (NVIDIA GPU) add-on, version 2.7.32 or later, is built on NVIDIA DCGM, providing advanced GPU monitoring functionalities. DCGM offers a broad spectrum of GPU monitoring metrics, featuring:

- GPU behavior monitoring
- GPU configuration management
- GPU policy management
- GPU health diagnosis
- Statistics on GPUs and threads
- Configuration and monitoring for NVSwitches

This section leverages CCE Cloud Native Cluster Monitoring and DCGM Exporter to monitor GPUs in comprehensive scenarios. For details about commonly used metrics, see **GPU Metrics**. For more information about DCGM-Exporter, see **DCGM-Exporter**.

Prerequisites

A NVIDIA GPU node is running properly in the cluster.

Step 1: Enable DCGM-Exporter

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE AI Suite (NVIDIA GPU)** on the right, and click **Install**.

Step 2 Enable **Use DCGM-Exporter to Observe DCGM Metrics**. Then, DCGM-Exporter will be deployed as a DaemonSet on GPU nodes.

NOTICE

DCGM-Exporter is seamlessly integrated into NVIDIA GPU of version 2.1.24, 2.7.40, or later versions. However, it is not supported in earlier versions.

To send GPU monitoring data to AOM, enable **Report Monitoring Data to AOM** in Cloud Native Cluster Monitoring after enabling DCGM-Exporter. GPU metrics reported to AOM are custom metrics and you will be billed on a pay-per-use basis for them. For details, see **Pricing Details**.

Step 3 Configure other parameters for the add-on and click **Install**. For details about parameter settings, see **CCE AI Suite (NVIDIA GPU)**.

----End

Step 2: Collect DCGM Metrics

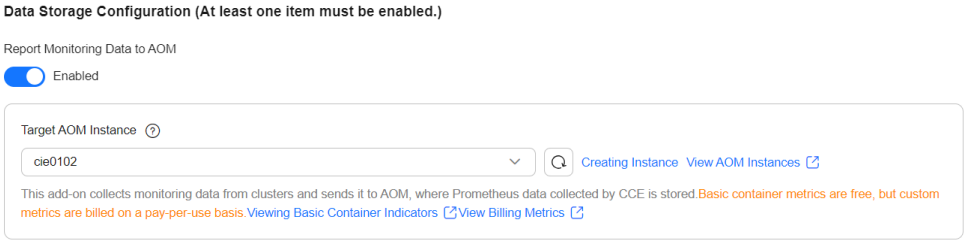
NOTICE

By default, the metrics exposed by DCGM-Exporter are not collected and reported by Prometheus. To use Prometheus or Grafana to view these metrics, enable ServiceMonitor for DCGM-Exporter.

ServiceMonitor for DCGM-Exporter has been preset in Cloud Native Cluster Monitoring (kube-prometheus-stack) of version 3.12.0 or later. You can enable it in **Settings**.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

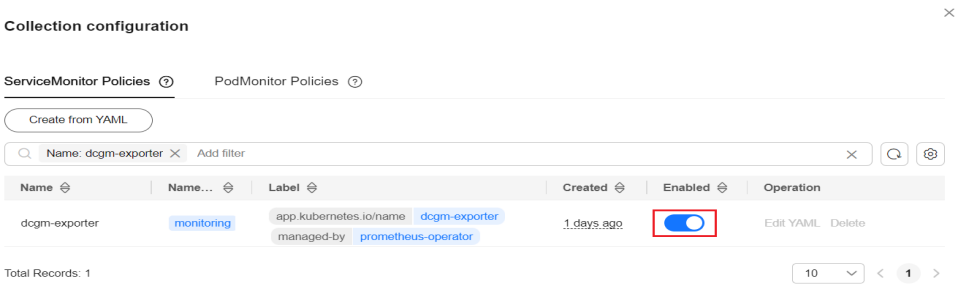
Step 2 In the navigation pane, choose **Add-ons** and install the **Cloud Native Cluster Monitoring**. Then, enable **Report Monitoring Data to AOM** and select the target AOM instance. For details about other settings, see **Cloud Native Cluster Monitoring**.



Step 3 In the navigation pane, choose **Settings**. Then, click the **Monitoring** tab.

Step 4 In the **Collection configuration** area, find ServiceMonitor and click **Manage**.

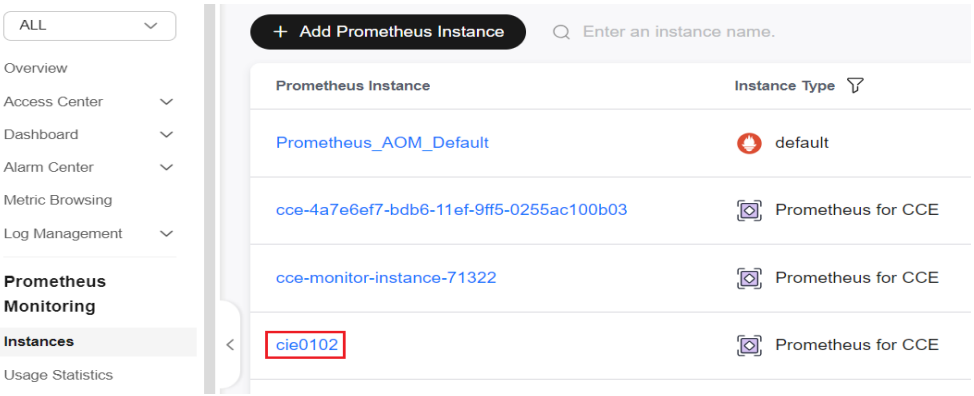
Step 5 Search for ServiceMonitor of DCGM-Exporter and enable it.



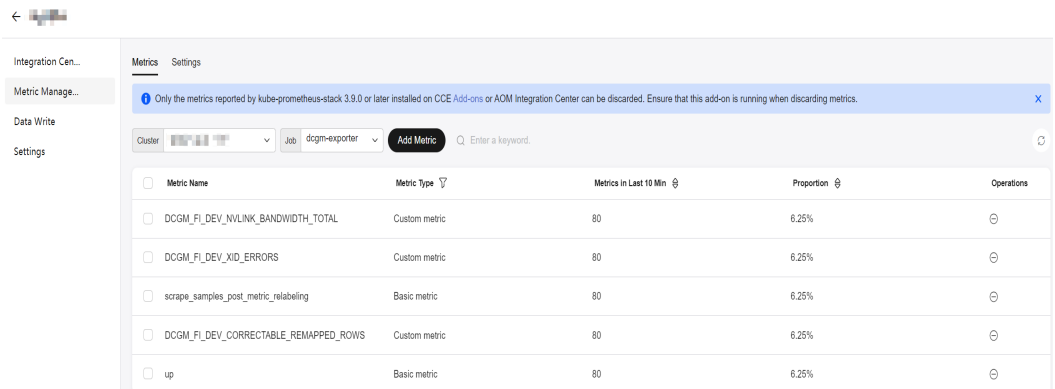
----End

Step 3: View DCGM Metrics on AOM

Step 1 Go to the AOM console and select the target AOM instance in the instance list.



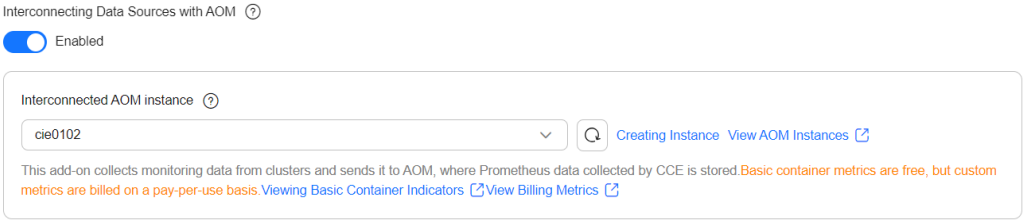
Step 2 Choose **Metric Management** in the navigation pane and check DCGM metrics.



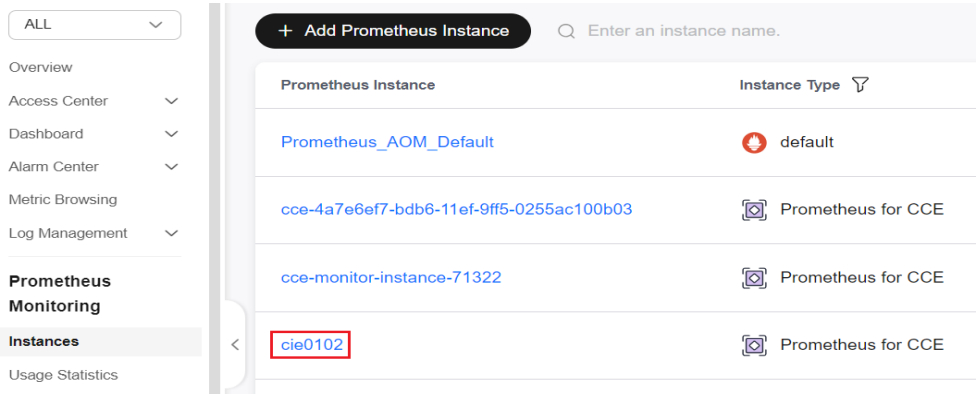
----End

Step 4: Use Grafana to View DCGM Metrics

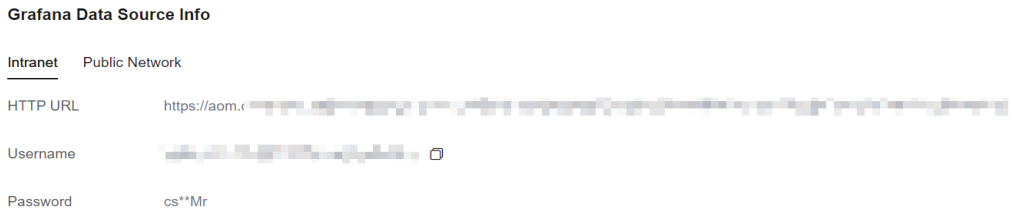
Step 1 In the navigation pane, choose **Add-ons**, install **Grafana**, enable **Interconnecting Data Sources with AOM**, and select the interconnected AOM instance.



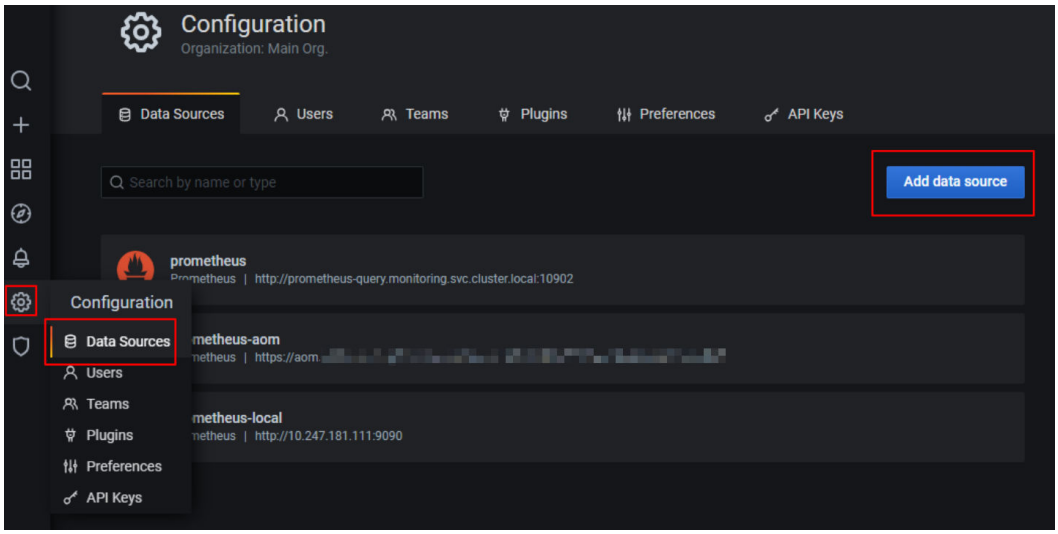
Step 2 Go to the AOM console and select the target AOM instance in the instance list.



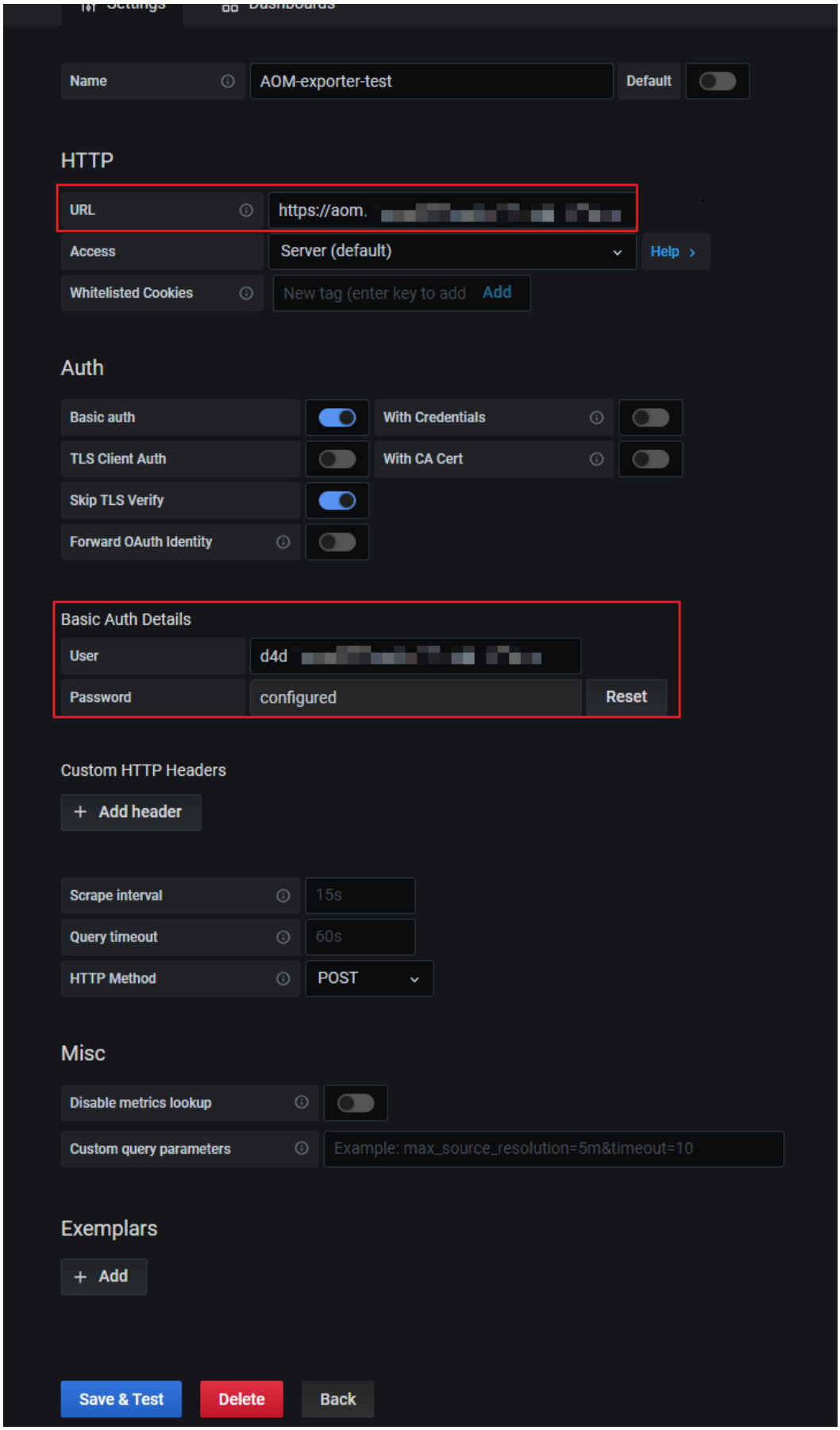
Step 3 Choose **Settings** in the navigation pane. In the **Grafana Data Source Info** area, obtain the AOM instance URL, username, and password. If **Grafana Data Source Info** is unavailable for the AOM instance, click **Add Access Code** in the **Credential** area to generate the Grafana data source configuration.



Step 4 Access Grafana and add the Prometheus data source to Grafana.



Step 5 Configure AOM settings, including enabling **Basic auth** and **Skip TLS Verify**.



Step 6 Import **NVIDIA DCGM Exporter Dashboard**, which is provided by NVIDIA for displaying DCGM metrics. For details about how to import dashboards to Grafana, see **Manage dashboards**.

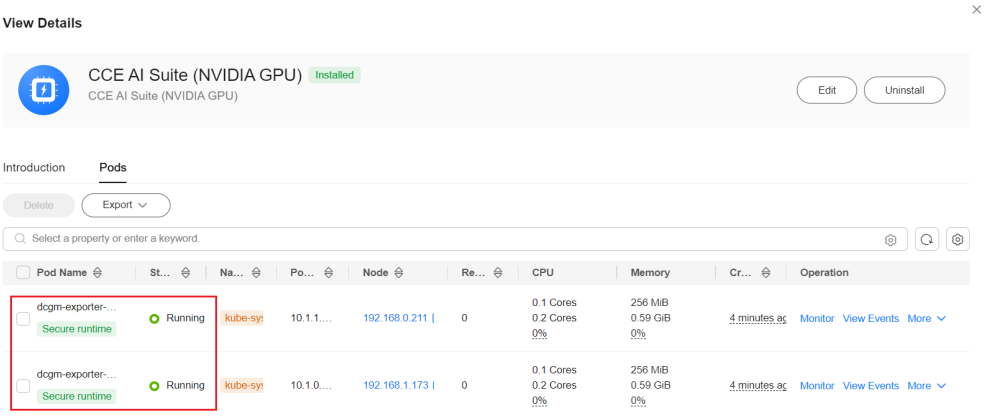
Step 7 View the imported dashboard.

----End

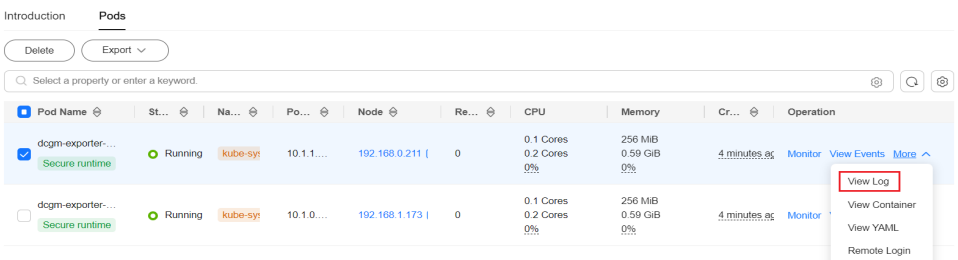
Appendix: Troubleshooting DCGM-Exporter Faults

Check the running status.

1. On the NVIDIA GPU details page, check whether the target pod is running.



2. Check pod logs for the HTTP server listening status.




```
2024/12/23 18:53:46 maxprocs: Updating GOMAXPROCS=1: using minimum allowed GOMAXPROCS
time="2024-12-23T18:53:46+08:00" level=info msg="Starting dcgm-exporter"
time="2024-12-23T18:53:46+08:00" level=info msg="DCGM successfully initialized!"
time="2024-12-23T18:53:47+08:00" level=info msg="Not collecting DCP metrics: This request is serviced by a module of DCGM that is
not currently loaded"
time="2024-12-23T18:53:47+08:00" level=info msg="Falling back to metric file '/etc/dcgm-exporter/default-counters.csv'"
time="2024-12-23T18:53:47+08:00" level=warning msg="Skipping line 6 ('DCGM_FI_PROF_PCIE_TX_BYTES'): metric not enabled"
time="2024-12-23T18:53:47+08:00" level=warning msg="Skipping line 7 ('DCGM_FI_PROF_PCIE_RX_BYTES'): metric not enabled"
time="2024-12-23T18:53:47+08:00" level=info msg="Initializing system entities of type: GPU"
time="2024-12-23T18:53:47+08:00" level=info msg="Not collecting NvSwitch metrics; no fields to watch for device type: 3"
time="2024-12-23T18:53:47+08:00" level=info msg="Not collecting NvLink metrics; no fields to watch for device type: 6"
time="2024-12-23T18:53:47+08:00" level=info msg="Not collecting CPU metrics; no fields to watch for device type: 7"
time="2024-12-23T18:53:47+08:00" level=info msg="Not collecting CPU Core metrics; no fields to watch for device type: 8"
time="2024-12-23T18:53:47+08:00" level=info msg="Kubernetes metrics collection enabled!"
time="2024-12-23T18:53:47+08:00" level=info msg="Pipeline starting"
time="2024-12-23T18:53:47+08:00" level=info msg="Starting webserver"
time="2024-12-23T18:53:47+08:00" level=info msg="Listening on" address="10.1.0.63:9400"
time="2024-12-23T18:53:47+08:00" level=info msg="TLS is disabled." address="10.1.0.63:9400" http2=false
```

3. Run the **curl** command in the cluster to access DCGM-Exporter and check whether data can be obtained.

   a. Check DCGM-Exporter's pod IP address.
   ```
   kubectl get po -A -owide | grep dcgm
   ```

   b. Check data. In the following command, *10.1.1.15* is the obtained pod IP address:
   ```
   curl 10.1.1.15:9400/metrics | head
   ```

```
[root@ecs-4f50 ~]# kubectl get po -A -owide|grep dcgm
default       dcgm-exporter-c2gmp                       0/1     Terminating       0               12d     <none>          192.168.0.121   <none>           <none>
default       dcgm-exporter-pvlxj                       0/1     Terminating       0               12d     <none>          192.168.7.124   <none>           <none>
kube-system   dcgm-exporter-s29f9                       1/1     Running           5 (18m ago)     19m     10.1.1.15       192.168.0.120   <none>           <none>
kube-system   dcgm-exporter-sjx95                       1/1     Running           0               5h51m   10.1.0.49       192.168.0.104   <none>           <none>
[root@ecs-4f50 ~]# curl 10.1.1.15:9400/metrics | head
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
# HELP DCGM_FI_DEV_SM_CLOCK SM clock frequency (in MHz).-:--:-- --:--:--     0
1# TYPE DCGM_FI_DEV_SM_CLOCK gauge
0DCGM_FI_DEV_SM_CLOCK{gpu="0",UUID="GPU-1cd2c45d-7e2b-ac29-c284-66ac92dfe1f3",pci_bus_id="00000000:00:0D.0",device="nvidia0",modelName="Tesla T4",Hostname="dcgm-exporter-s29f9",DCGM_FI_DRIVER_VERSION="535.5
4.03"} 585
0 # HELP DCGM_FI_DEV_MEM_CLOCK Memory clock frequency (in MHz).
 # TYPE DCGM_FI_DEV_MEM_CLOCK gauge
5DCGM_FI_DEV_MEM_CLOCK{gpu="0",UUID="GPU-1cd2c45d-7e2b-ac29-c284-66ac92dfe1f3",pci_bus_id="00000000:00:0D.0",device="nvidia0",modelName="Tesla T4",Hostname="dcgm-exporter-s29f9",DCGM_FI_DRIVER_VERSION="535.
54.03"} 5000
90# HELP DCGM_FI_DEV_MEMORY_TEMP Memory temperature (in C).
2# TYPE DCGM_FI_DEV_MEMORY_TEMP gauge
 DCGM_FI_DEV_MEMORY_TEMP{gpu="0",UUID="GPU-1cd2c45d-7e2b-ac29-c284-66ac92dfe1f3",pci_bus_id="00000000:00:0D.0",device="nvidia0",modelName="Tesla T4",Hostname="dcgm-exporter-s29f9",DCGM_FI_DRIVER_VERSION="53
5.54.03"} 0
 # HELP DCGM_FI_DEV_GPU_TEMP GPU temperature (in C).
  0  5902    0     0  5169k      0 --:--:-- --:--:-- --:--:-- 5763k
```

Helpful Links

GPU Metrics

6.3.6 Configuring Workload Scaling Based on GPU Monitoring Metrics

If there are GPU nodes in a cluster, you can view the GPU resource usage of the nodes through GPU metrics, such as the GPU usage and used GPU memory. After obtaining GPU monitoring metrics, you can configure auto scaling policies based on the GPU metrics of applications to adaptively adjust the number of nodes for the applications when services fluctuate.

Prerequisites

- A cluster is available, and there are GPU nodes and GPU related services running in the cluster.
- The **CCE AI Suite (NVIDIA GPU)** add-on has been installed in the cluster, and the add-on metrics API is working properly. You can log in to the GPU node and run the following command:
  ```
  curl {Pod IP}:2112/metrics
  ```
  In the preceding command, *{Pod IP}* indicates the pod IP address of the GPU add-on. If the metric result is returned, the GPU add-on is running properly.

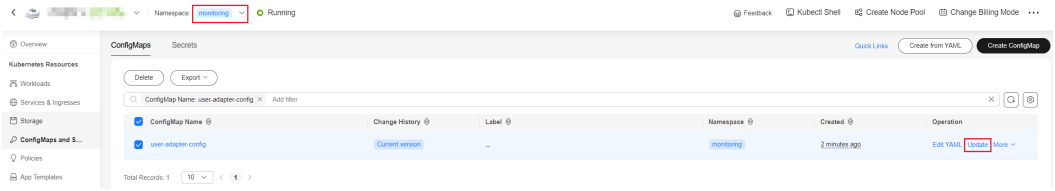
- The **Cloud Native Cluster Monitoring** add-on of v3.9.5 or later has been installed in the cluster, and it is deployed in local data storage mode.

Collecting GPU Metrics

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **ConfigMaps and Secrets**.

Step 2 Select the **monitoring** namespace. On the **ConfigMaps** tab, locate the row containing **user-adapter-config** and click **Update**.

Figure 6-9 Updating a ConfigMap



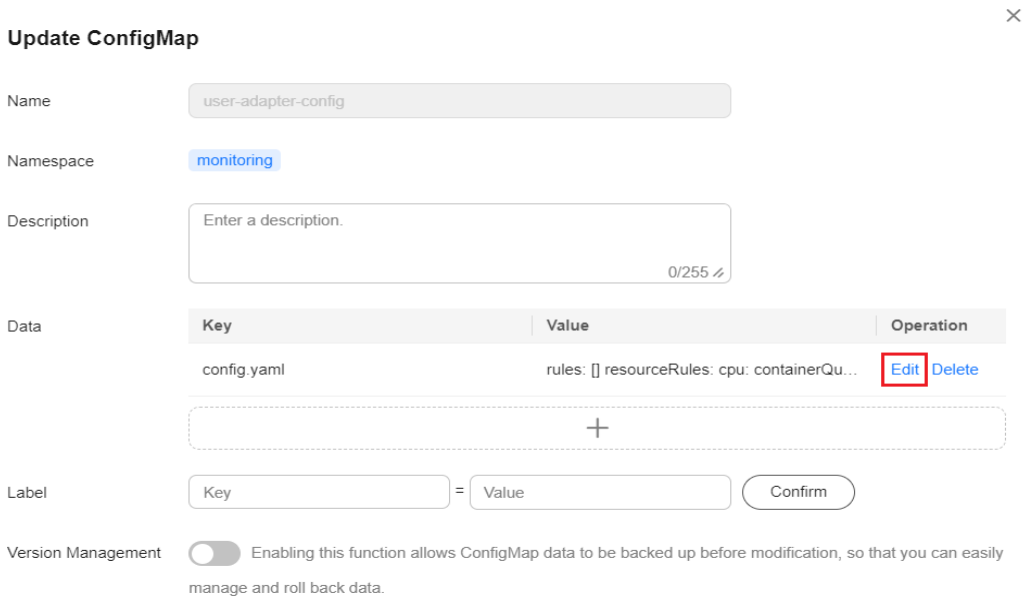
Step 3 On the **Update ConfigMap** page, click **Edit** in the **Operation** column of the **config.yaml** file in the **Data** pane. Then, add a custom metric collection rule under the **rules** field. Click **OK**.

You can add multiple collection rules by adding multiple configurations under the **rules** field. For details, see **Metrics Discovery and Presentation Configuration**.

The following is an example of a custom rule for collecting **cce_gpu_memory_utilization**. For details about more GPU metrics, see **Monitoring GPU Metrics**.

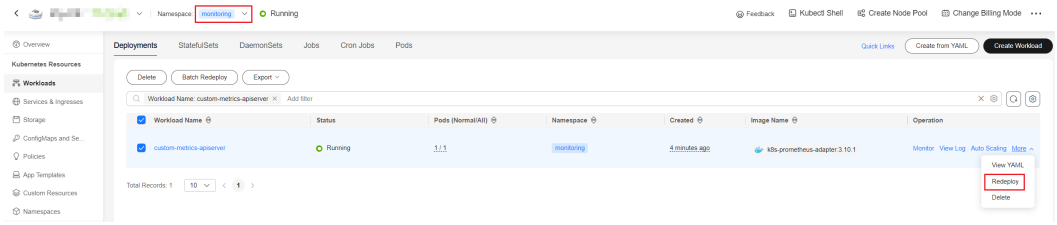
```
rules:
- seriesQuery: '{__name__=~"cce_gpu_memory_utilization",container!="",namespace!="",pod!=""}'
  seriesFilters: []
  resources:
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  metricsQuery: sum(last_over_time(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<<.GroupBy>>)
```


Figure 6-10 Customizing a collection rule



Step 4 Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

Figure 6-11 Redeploying custom-metrics-apiserver



Step 5 After the restart, check whether the metrics in the target pod are normal (replace the namespace and service pod names).

```
# Obtain metrics.
$ kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1"

{"kind":"APIResourceList","apiVersion":"v1","groupVersion":"custom.metrics.k8s.io/v1beta1","resources":
[{"name":"pods/
cce_gpu_memory_utilization","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},
{"name":"namespaces/
cce_gpu_memory_utilization","singularName":"","namespaced":false,"kind":"MetricValueList","verbs":
["get"]}]}

# Obtain workload metric values.
$ kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pods/test-gpu-
hpa-68667fdd94-grmd2/cce_gpu_memory_utilization"

{"kind":"MetricValueList","apiVersion":"custom.metrics.k8s.io/v1beta1","metadata":{},"items":
[{"describedObject":{"kind":"Pod","namespace":"default","name":"test-gpu-hpa-68667fdd94-
grmd2","apiVersion":"/
v1"},"metricName":"cce_gpu_memory_utilization","timestamp":"2024-01-10T08:36:44Z","value":"20","selecto
r":null}]}
```

----End

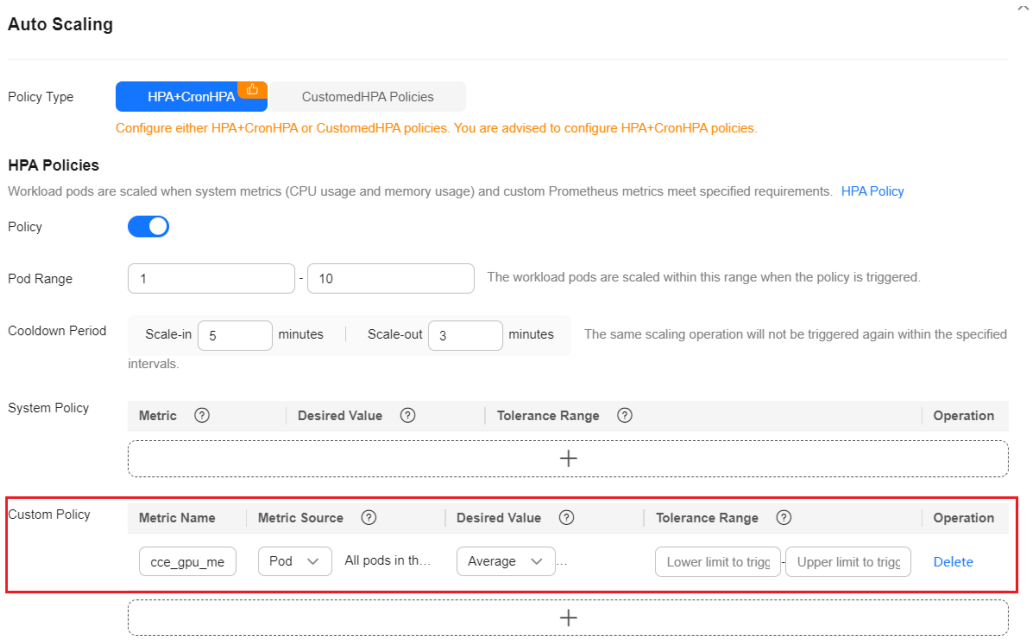
Creating an Auto Scaling Policy

Step 1 Choose **Workloads** in the navigation pane. Locate the target workload and choose **Auto Scaling** in the **Operation** column.

Step 2 Set **Policy Type** to **HPA+CronHPA** and enable HPA.

You can select GPU monitoring parameters in **Custom Policy** to create an auto scaling policy. The following is an example.

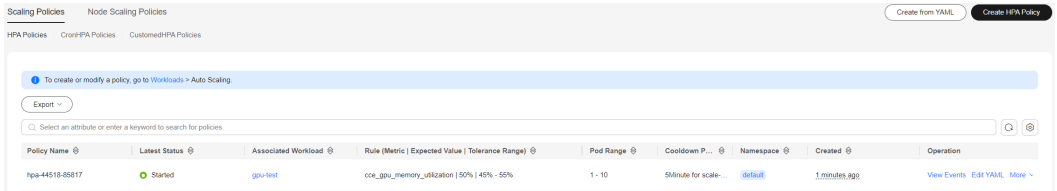
Figure 6-12 Selecting custom metrics



In this example, **cce_gpu_memory_utilization** (GPU memory usage) is used as the scaling metric. For details about how to configure other HPA parameters, see **Creating an HPA Policy**.

Step 3 Return to the **Scaling Policies** page and check whether the HPA policy has been created.

Figure 6-13 HPA policy created



----End

6.3.7 Configuring Auto Scaling for xGPU Nodes

If there are not enough GPU virtualization resources in a cluster, xGPU nodes can be scaled out automatically. This section describes how to create an auto scaling policy for xGPU nodes.

Prerequisites

- A cluster of v1.28 or v1.29 is available.
- **CCE AI Suite (NVIDIA GPU)** (v2.7.5 or later), **Volcano Scheduler**, and **CCE Cluster Autoscaler** (v1.28.78/v1.29.41 or later) have been installed in the cluster.

Step 1: Configure the Node Pool

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**.

Step 2 Click **Create Node Pool** to create an xGPU node pool. For details, see **Creating a Node Pool**.

For details about requirements on xGPU nodes, such as the specifications, OS, and runtime, see **Preparing xGPU Resources**.

Step 3 After the node pool is created, click **Auto Scaling**. In the **AS Object** area, enable **Auto Scaling** for the target specification and click **OK**.

----End

Step 2: Configure Heterogeneous Resources

Step 1 In the navigation pane, choose **Settings**. Then, click the **Heterogeneous Resources** tab.

Step 2 In the **GPU Settings** area, locate **Node Pool Configurations** and select the created node pool.

Step 3 Select a driver that meets GPU virtualization requirements and enable GPU virtualization based on **Preparing xGPU Resources**.

Figure 6-14 Heterogeneous Resources

Step 4 Click **Confirm configuration**.

----End

Step 3: Create a GPU Virtualization Workload and Trigger Capacity Expansion

Create a Deployment that uses GPU virtualization resources and requests a number of GPUs exceeding the current upper limit available in the cluster. For details, see **Using GPU Virtualization**. For example, there is a total of 16 GiB of

GPU memory available, with each pod requiring 1 GiB. Then, configure 17 pods, which need a total of 17 GiB of GPU memory.

After a short period of time, you can find GPU node scale-out on the node pool details page.

6.3.8 GPU Fault Handling

Prerequisites

Cloud Native Log Collection has been installed in the cluster so that GPU events can be synchronously reported to AOM.

GPU Isolation Event

When a GPU malfunctions, the system automatically isolates the faulty GPU. For details, see **Table 6-3**.

Table 6-3 GPU isolation event

Event Cause	Error Details	Description	Result
GPUMemoryError	Device=%s, UUID=%s, SN=%s has failed remapped rows; The device will go unhealthy.	Failed to obtain the number of remapped rows in NVML.	The faulty GPU device is isolated.
GPUMemoryError	Device=%s, UUID=%s, SN=%s has more than 60 retired pages caused by both multiple single bit ecc error and double bit ecc error, DBE error number: %d, SBE error number: %d; The device will go unhealthy.	The total number of DBE errors and SBE errors of the GPU device is greater than 60.	The faulty GPU device is isolated.
GPUMemoryError	Device=%s, UUID=%s, SN=%s has more than 4 SRAM uncorrectable ecc errors count; The device will go unhealthy.	The number of uncorrectable ECC errors of the GPU device is greater than 4.	The faulty GPU device is isolated.
GPUXidError	Failed to determine gpu device uuid for Xid=%d; Marking all devices as unhealthy.	Failed to obtain the UUID using NVML.	The GPU device of the faulty GPU node is isolated.
GPUXidError	Xid=%d on Device=%s, UUID=%s, SN=%s, the device will go unhealthy.	GPU Xid error occurred, and the affected Xids are 74 and 79.	The faulty GPU device is isolated.

Event Cause	Error Details	Description	Result
GPUHealthWarning	Device=%s, UUID=%s, SN=%s failed to get fan state.	The fan on the GPU device is not running properly.	The affected GPU device is not isolated.
GPUHealthWarning	Device=%s, UUID=%s, SN=%s failed to get power state.	Failed to obtain the power of the GPU device.	The affected GPU device is not isolated.

Fault Locating

- **Failed to obtain the number of remapped rows in NVML.**

  The GPU driver or GPU device malfunctions. Contact customer service based on the type of the node (ECS or BMS) where the GPU device resides.

- **The total number of DBE errors and SBE errors of the GPU device is high.**

  The GPU driver or GPU device malfunctions. Contact customer service based on the type of the node (ECS or BMS) where the GPU device resides.

- **There are uncorrectable ECC errors of the GPU device.**

  a. Log in to the node where the GPU isolation event occurred.

  b. Go to the **/usr/local/nvidia/bin** directory and run the **nvidia-smi -q** command.

  If the **nvidia-smi** command is unavailable or fails to be executed, the failure may be caused by the lack of GPU driver. Reinstall the GPU driver and try again.

  c. Check the **ECC ERROR** in the command output.

  - **Correctable Error**: Such an error will not interrupt services or trigger GPU isolation.
  - **Uncorrectable Error**: Such an error will interrupt services and trigger GPU isolation.

  d. If there are uncorrectable errors, perform the following operations to rectify the fault:

  i. Configure taints on the target node to evict the existing service load from the node.

  ii. Restart the target node.

  iii. If the fault persists, collect the output of the **nvidia-smi -q** command and contact customer service based on the type of the node (ECS or BMS) where the GPU device resides.

- **Failed to obtain the UUID using NVML.**

  a. Log in to the node where the GPU isolation event occurred.

  b. Access **/usr/local/nvidia/bin**.

c. Run the **nvidia-smi** command and check the device ID in the command output, for example, **00:0D.0**.

   If the **nvidia-smi** command is unavailable or fails to be executed, the failure may be caused by the lack of GPU driver. Reinstall the GPU driver and try again.

d. Run the **lspci | grep NVIDIA** command and check the device ID in the command output.

e. Compare the two results. If they do not match, contact customer service based on the type of the node (ECS or BMS) where the GPU device is located.

- **The Xid of the GPU device is incorrect.**

  a. Log in to the node where the GPU isolation event occurred.

  b. Run the **dmesg -T | grep -i NVRM** command and check the command output.

  c. If information in the "Xid(PCI:0000:00:0x): xx" format is displayed, collect the error code and identify the cause based on **NVIDIA Xid Errors**. Collect the error information and detailed cause and contact customer service based on the type of the node (ECS or BMS) where the GPU device resides.

- **The available memory of xGPU devices is far less than the physical GPU memory.**

  a. Log in to the xGPU node.

  b. Run the **/usr/local/nvidia/bin/nvidia-smi** command to obtain the physical GPU memory of the target GPU and record its serial number.

  c. Run the **cat /proc/xgpu/***{GPU serial number}***/meminfo** command to obtain the available xGPU memory. Replace *{GPU serial number}* with the one obtained in preceding step.

  d. Compare the obtained GPU memory.

     NOTE

     The driver of the GPU vendor occupies a certain amount of physical GPU memory, which is about 300 MiB. This is normal. For example, if Tesla T4 GPUs run with NVIDIA driver 510.47.03, the driver occupies the GPU memory of 280 MiB by default. The value varies depending on the driver version. For example, the 535 series driver occupies more memory than the 470 series driver.

     If the available xGPU memory is far less than the physical GPU memory, some containers that are not provisioned using GPU virtualization occupy the GPU memory.

  e. In this case, clear the GPU load on the target node through the CCE console or by using kubectl.

  f. Run the **rmmod xgpu_km** command to delete GPU virtualization.

  g. Delete the nvidia-gpu-device-plugin pods on the target node through the CCE console or by using kubectl.

  h. After the nvidia-gpu-device-plugin pods are rebuilt, perform steps **2** and **3** again to verify the result.

6.3.9 GPU Metrics

The CCE AI Suite (NVIDIA GPU) add-on provides GPU monitoring metrics and integrates DCGM-Exporter. To use DCGM-Exporter, make sure you have version 2.7.32 or later of the add-on installed. This add-on offers additional GPU observability options. This section describes the metrics provided by CCE AI Suite (NVIDIA GPU).

Billing

GPU metrics are custom ones. If you plan to have them reported to AOM, you will be billed on a pay-per-use basis. To avoid any extra fees, review **Pricing Details** carefully before enabling this function.

GPU Metrics Provided by CCE

NOTE

The CCE AI Suite (NVIDIA GPU) add-on version 2.1.24, 2.7.40, or later includes the ability to read the used xGPU compute, used xGPU memory, and total xGPU memory in addition to the basic GPU metrics.

- The **cce_gpu_memory_total** metric supports the collection of **xgpu_memory_total** data.
- The **cce_gpu_memory_used** metric supports the collection of **xgpu_memory_used** data.
- The **cce_gpu_utilization** metric supports the collection of **xgpu_core_percentage_used** data.

When CCE shows the GPU metric data, it also shows the xGPU metric data. The xGPU metric data is identified by the **gpu_index** label in the format of {**gpu_index**="*M*|*N*"}, where *M* represents the GPU serial number (**gpu_index**) and *N* represents the xGPU serial number (**xgpu_index**) of the GPU. You can use the **gpu_index** label to get xGPU metrics. For example:

```
cce_gpu_memory_used{gpu_index="0|1"} 16000
```

It indicates that the used memory of the xGPU whose **xgpu_index** is **1** on GPU0 is 16,000 bytes.

If you do not need to see xGPU metrics, you can filter them out using regular expressions. For example:

```
cce_gpu_memory_used{gpu_index=~"[^|]"}
```


Table 6-4 Basic GPU monitoring metrics

Type	Metric	Type	Unit	Monitoring Level	Description
Utilization	cce_gpu_utilization	Gauge	%	GPU cards	GPU compute usage **NOTE** If the add-on version is 2.1.24, 2.7.40, or later, this metric can also be used to obtain the corresponding xGPU metric (**xgpu_core_percentage_used**). You can use the **gpu_index** label of the metric to obtain the xGPU metric. For example, **gpu_index=0|0** indicates GPU 0 and xGPU 0.
	cce_gpu_memory_utilization	Gauge	%	GPU cards	GPU memory usage
	cce_gpu_encoder_utilization	Gauge	%	GPU cards	GPU encoding usage
	cce_gpu_decoder_utilization	Gauge	%	GPU cards	GPU decoding usage
	cce_gpu_utilization_process	Gauge	%	GPU processes	GPU compute usage of each process
	cce_gpu_memory_utilization_process	Gauge	%	GPU processes	GPU memory usage of each process
	cce_gpu_encoder_utilization_process	Gauge	%	GPU processes	GPU encoding usage of each process
	cce_gpu_decoder_utilization_process	Gauge	%	GPU processes	GPU decoding usage of each process

Type	Metric	Type	Unit	Monitoring Level	Description
Memory	cce_gpu_memory_used	Gauge	Byte	GPU cards	Used GPU memory **NOTE** If the add-on version is 2.1.24, 2.7.40, or later, this metric can also be used to obtain the corresponding xGPU metric (**xgpu_memory_used**). You can use the **gpu_index** label of the metric to obtain the xGPU metric. For example, **gpu_index=0|0** indicates GPU 0 and xGPU 0.
	cce_gpu_memory_total	Gauge	Byte	GPU cards	Total GPU memory **NOTE** If the add-on version is 2.1.24, 2.7.40, or later, this metric can also be used to obtain the corresponding xGPU metric (**xgpu_memory_total**). You can use the **gpu_index** label of the metric to obtain the xGPU metric. For example, **gpu_index=0|0** indicates GPU 0 and xGPU 0.
	cce_gpu_memory_free	Gauge	Byte	GPU cards	Idle GPU memory
	cce_gpu_bar1_memory_used	Gauge	Byte	GPU cards	Used GPU BAR1 memory
	cce_gpu_bar1_memory_total	Gauge	Byte	GPU cards	Total GPU BAR1 memory
Frequency	cce_gpu_clock	Gauge	MHz	GPU cards	GPU clock frequency
	cce_gpu_memory_clock	Gauge	MHz	GPU cards	The speed at which the GPU memory operates
	cce_gpu_graphics_clock	Gauge	MHz	GPU cards	GPU frequency
	cce_gpu_video_clock	Gauge	MHz	GPU cards	GPU video processor frequency

Type	Metric	Type	Unit	Monitoring Level	Description
Physical status	cce_gpu_temperature	Gauge	°C	GPU cards	GPU temperature
	cce_gpu_power_usage	Gauge	Milli watt	GPU cards	GPU power
	cce_gpu_total_energy_consumption	Gauge	Millij oule	GPU cards	Total GPU energy consumption
Bandwidth	cce_gpu_pcie_link_bandwidth	Gauge	bit	GPU cards	GPU PCIe bandwidth
	cce_gpu_nvlink_bandwidth	Gauge	Gbit/s	GPU cards	GPU NVLink bandwidth
	cce_gpu_pcie_throughput_rx	Gauge	KB/s	GPU cards	GPU PCIe RX bandwidth
	cce_gpu_pcie_throughput_tx	Gauge	KB/s	GPU cards	GPU PCIe TX bandwidth
	cce_gpu_nvlink_utilization_counter_rx	Gauge	KB/s	GPU cards	GPU NVLink RX bandwidth
	cce_gpu_nvlink_utilization_counter_tx	Gauge	KB/s	GPU cards	GPU NVLink TX bandwidth
Memory isolation page	cce_gpu_retired_pages_sbe	Gauge	N/A	GPU cards	Number of isolated GPU memory pages with single-bit errors
	cce_gpu_retired_pages_dbe	Gauge	N/A	GPU cards	Number of isolated GPU memory pages with dual-bit errors

Table 6-5 xGPU monitoring metrics

Metric	Type	Unit	Monitoring Level	Description
xgpu_memory_total	Gauge	Byte	GPU processes	Total xGPU memory

Metric	Type	Unit	Monitoring Level	Description
xgpu_memory_used	Gauge	Byte	GPU processes	Used xGPU memory
xgpu_core_percentage_total	Gauge	%	GPU processes	Total xGPU cores
xgpu_core_percentage_used	Gauge	%	GPU processes	Used xGPU cores
gpu_schedule_policy	Gauge	N/A	GPU cards	xGPU scheduling policy. Options: • **0**: xGPU memory is isolated and cores are shared. • **1**: Both xGPU memory and cores are isolated. • **2**: default mode, indicating that the current card is not used by any xGPU device for allocation.
xgpu_device_health	Gauge	N/A	GPU cards	xGPU device health. Options: • **0**: The xGPU device is healthy. • **1**: The xGPU device is unhealthy.

GPU Metrics Provided by DCGM

Table 6-6 Utilization

Metric	Type	Unit	Description
DCGM_FI_DEV_GPU_UTIL	Gauge	%	GPU utilization. It specifies the time during which one or more kernel functions are active in a period (1s or 1/6s, which varies with the GPU models). This metric displays only the GPUs used by kernel functions, but does not display the specific usage.
DCGM_FI_DEV_MEM_COPY_UTIL	Gauge	%	GPU memory bandwidth utilization of a measured object For example, the maximum memory bandwidth of NVIDIA GPU V100 is 900 GB/s. If the current memory bandwidth is 450 GB/s, the memory bandwidth usage is 50%.

Metric	Type	Unit	Description
DCGM_FI_DEV_ENC_UTIL	Gauge	%	GPU encoder utilization of a measured object
DCGM_FI_DEV_DEC_UTIL	Gauge	%	GPU decoder utilization of a measured object

Table 6-7 Memory

Metric	Type	Unit	Description
DCGM_FI_DEV_FB_FREE	Gauge	MB	Number of remaining GPU memory
DCGM_FI_DEV_FB_USED	Gauge	MB	Number of used GPU memory The value is the same as the value of **Memory-Usage** in the nvidia-smi command.

Table 6-8 Profiling

Metric	Type	Unit	Description
DCGM_FI_PROF_GR_ENGINE_ACTIVE	Gauge	%	Percentage of the time when the graphic or compute engine is in the active state within a period. This is an average value of all graphic or compute engines. An active graphic or compute engine indicates that the graphic or compute context is associated with a thread and the graphic or compute context is busy.

Metric	Type	Unit	Description
DCGM_FI_PROF_SM_ACTIVE	Gauge	%	Fraction of the time during which at least one thread bundle is active on an SM within a period. This is an average value of all SMs and is insensitive to the number of threads in each block. A thread bundle is active after being scheduled and allocated with resources. The thread bundle may be in the computing state or a non-computing state (for example, waiting for a memory request). If the value is less than 0.5, GPUs are not efficiently used. The value should be greater than 0.8. For example, a GPU has N SMs: ● A kernel function uses N thread blocks to run on all SMs in a period. In this case, the value is 1 (100%). ● A kernel function runs N/5 thread blocks in a period. In this case, the value is 0.2. ● A kernel function uses N thread blocks and runs only 1/5 of cycles in a period. In this case, the value is 0.2.
DCGM_FI_PROF_SM_OCCUPANCY	Gauge	%	Ratio of the number of thread bundles that reside on the SM to the maximum number of thread bundles that can reside on the SM within a period. This is an average value of all SMs within a period. A high value does not mean a high GPU usage. Only when the GPU memory bandwidth is limited, a high value of workloads (**DCGM_FI_PROF_DRAM_ACTIVE**) indicates more efficient GPU usage.

Metric	Type	Unit	Description
DCGM_FI_PROF_PIPE_TENSOR_ACTIVE	Gauge	%	Fraction of cycles during which the tensor (HMMA/IMMA) pipe is active. This is an average value within a period, not an instantaneous value. A higher value indicates a higher utilization of tensor cores. Value 1 (100%) indicates that a tensor instruction is sent every instruction cycle in the entire period (one instruction is completed in two cycles). If the value is 0.2 (20%), the possible causes are as follows: ● During the entire period, 20% of the SM tensor cores run at 100% utilization. ● During the entire period, all SM tensor cores run at 20% utilization. ● During 1/5 of the entire period, all SM tensor cores run at 100% utilization. ● Other combinations
DCGM_FI_PROF_PIPE_FP64_ACTIVE	Gauge	%	Fraction of cycles during which the FP64 (double precision) pipe is active. This is an average value within a period, not an instantaneous value. A larger value indicates a higher usage of FP64 cores. Value 1 (100%) indicates that the FP64 instruction is executed every four cycles (for example, Volta cards) in a period. If the value is 0.2 (20%), the possible causes are as follows: ● During the entire period, 20% of the SM FP64 cores run at 100% utilization. ● During the entire period, all SM FP64 cores run at 20% utilization. ● During 1/5 of the entire period, all SM FP64 cores run at 100% utilization. ● Other combinations

Metric	Type	Unit	Description
DCGM_FI_PROF_PIPE_FP32_ACTIVE	Gauge	%	Fraction of cycles during which the fused multiply-add (FMA) pipe is active. Multiply-add applies to FP32 (single precision) and integers. This is an average value within a period, not an instantaneous value. A larger value indicates a higher usage of FP32 cores. Value 1 (100%) indicates that the FP32 instruction is executed every two cycles (for example, Volta cards) in a period. If the value is 0.2 (20%), the possible causes are as follows: ● During the entire period, 20% of the SM FP32 cores run at 100% utilization. ● During the entire period, all SM FP32 cores run at 20% utilization. ● During 1/5 of the entire period, all SM FP32 cores run at 100% utilization. ● Other combinations
DCGM_FI_PROF_PIPE_FP16_ACTIVE	Gauge	%	Fraction of cycles during which the FP16 (half-precision) pipe is active. This is an average value within a period, not an instantaneous value. A larger value indicates a higher usage of FP16 cores. Value 1 (100%) indicates that the FP16 instruction is executed every two cycles (for example, Volta cards) in a period. If the value is 0.2 (20%), the possible causes are as follows: ● During the entire period, 20% of the SM FP16 cores run at 100% utilization. ● During the entire period, all SM FP16 cores run at 20% utilization. ● During 1/5 of the entire period, all SM FP16 cores run at 100% utilization. ● Other combinations

Metric	Type	Unit	Description
DCGM_FI_PROF_DRAM_ACTIVE	Gauge	%	Fraction of cycles during which **Memory BW Utilization** sends data to or receives from device memory. This is an average value within a period, not an instantaneous value. A higher value indicates a higher utilization of device memory. Value 1 (100%) indicates that a **DRAM** instruction is executed in every cycle throughout the entire time period (although a peak value of around 0.8 (80%) is the maximum achievable). If the value is set to 0.2 (20%), it means that 20% of the cycles involve reading from or writing to the device memory within the given time period.
DCGM_FI_PROF_PCIE_TX_BYTES DCGM_FI_PROF_PCIE_RX_BYTES	Counter	Byte/s	Rate of data transmitted or received over the PCIe bus, including the protocol header and data payload. This is an average value within a period, not an instantaneous value. The rate is averaged over the period. For example, if 1 GB of data is transmitted within 1 second, the transmission rate is 1 GB/s regardless of whether the data is transmitted at a constant rate or burst. Theoretically, the maximum PCIe Gen3 bandwidth is 985 MB/s per channel.
DCGM_FI_PROF_NVLINK_RX_BYTES DCGM_FI_PROF_NVLINK_TX_BYTES	Counter	Byte/s	Rate at which data is transmitted or received through NVLink, excluding the protocol header. This is an average value within a period, not an instantaneous value. The rate is averaged over the period. For example, if 1 GB of data is transmitted within 1 second, the transmission rate is 1 GB/s regardless of whether the data is transmitted at a constant rate or burst. Theoretically, the maximum NVLink Gen2 bandwidth is 25 GB/s per link in each direction.

Table 6-9 Frequency (clock)

Metric	Type	Unit	Description
DCGM_FI_DEV_SM_CLOCK	Gauge	MHz	SM clock for the device

Metric	Type	Unit	Description
DCGM_FI_DEV_MEM_CLOCK	Gauge	MHz	Memory clock for the device
DCGM_FI_DEV_APP_SM_CLOCK	Gauge	MHz	SM application clocks
DCGM_FI_DEV_APP_MEM_CLOCK	Gauge	MHz	Memory application clocks
DCGM_FI_DEV_CLOCK_THROTTLE_REASONS	Gauge	MHz	The reason why the clock is throttled

Table 6-10 XID errors and violations

Metric	Type	Unit	Description
DCGM_FI_DEV_XID_ERRORS	Gauge	N/A	The last XID error that occurs in a period of time
DCGM_FI_DEV_POWER_VIOLATION	Counter	µs	A violation caused by the power limit. The value is the time when the violation occurs.
DCGM_FI_DEV_THERMAL_VIOLATION	Counter	µs	A violation caused by the thermal limit. The value is the time when the violation occurs.
DCGM_FI_DEV_SYNC_BOOST_VIOLATION	Counter	µs	A violation caused by the synchronous boost limit. The value is the time when the violation occurs.
DCGM_FI_DEV_BOARD_LIMIT_VIOLATION	Counter	µs	A violation caused by the board limit. The value is the time when the violation occurs.
DCGM_FI_DEV_LOW_UTIL_VIOLATION	Counter	µs	A violation caused by the low utilisation limit. The value is the time when the violation occurs.
DCGM_FI_DEV_RELIABILITY_VIOLATION	Counter	µs	A violation caused by the reliability limit. The value is the time when the violation occurs.

Table 6-11 BAR1

Metric	Type	Unit	Description
DCGM_FI_DEV_BAR1_USED	Gauge	MB	The used BAR1

Metric	Type	Unit	Description
DCGM_FI_DEV_BAR1_FREE	Gauge	MB	The remaining BAR1

Table 6-12 Temperature and power

Metric	Type	Unit	Description
DCGM_FI_DEV_MEMORY_TEMP	Gauge	°C	Memory temperature
DCGM_FI_DEV_GPU_TEMP	Gauge	°C	GPU temperature
DCGM_FI_DEV_POWER_USAGE	Gauge	Watt	GPU power
DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION	Counter	Millijoule	Energy consumed since a driver was loaded

Table 6-13 Retired pages

Metric	Type	Unit	Description
DCGM_FI_DEV_RETIRED_SBE	Gauge	N/A	Number of retired pages due to single bit errors
DCGM_FI_DEV_RETIRED_DBE	Gauge	N/A	Number of retired pages due to double bit errors

For details about more DCGM metrics, see **Field Identifiers**.

6.4 NPU Scheduling

You can use NPUs in CCE containers.

Prerequisites

- An NPU node has been created. For details, see **Creating a Node**.
- The huawei-npu has been installed. For details, see **CCE AI Suite (Ascend NPU)**.

Using NPUs

Create a workload and request NPUs. You can specify the number of NPUs as follows:


```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```

Specify the number of NPUs to be requested in **huawei.com/ascend-310**.

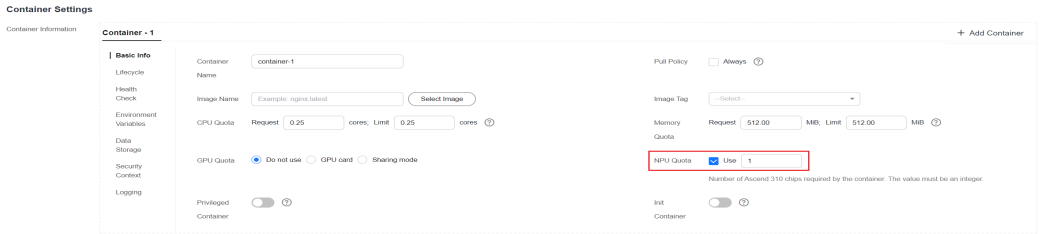
NOTE

When you use **huawei.com/ascend-310** to specify the number of NPUs, the values of requests and limits must be the same.

After **huawei.com/ascend-310** is specified, workloads will be scheduled only to nodes with NPUs. If NPUs are insufficient, a Kubernetes event similar to "0/2 nodes are available: 2 Insufficient huawei.com/ascend-310." will be reported.

To use NPUs on the CCE console, select the NPU quota and specify the number of NPUs to be used when creating a workload.

Figure 6-15 Using NPUs



NPU Node Labels

CCE will label NPU-enabled nodes that are ready to use.

```
$ kubectl get node -L accelerator/huawei-npu
NAME          STATUS   ROLES    AGE     VERSION                                 HUAWEI-NPU
10.100.2.59   Ready    <none>   2m18s   v1.19.10-r0-CCE21.11.1.B006-21.11.1.B006   ascend-310
```

When using NPUs, you can enable the affinity between pods and nodes based on labels so that the pods can be scheduled to the correct nodes.


```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: npu-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: npu-test
  template:
    metadata:
      labels:
        app: npu-test
    spec:
      nodeSelector:
        accelerator/huawei-npu: ascend-310
      containers:
        - name: container-0
          image: nginx:perl
          resources:
            limits:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
            requests:
              cpu: 250m
              huawei.com/ascend-310: '1'
              memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```

6.5 Volcano Scheduling

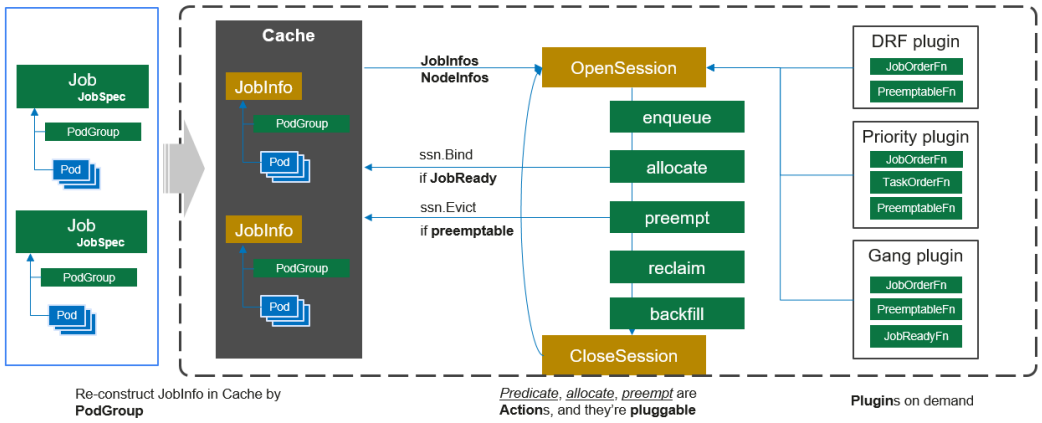
6.5.1 Overview

Volcano is a batch processing platform that runs on Kubernetes for machine learning, deep learning, bioinformatics, genomics, and other big data applications. It provides general-purpose, high-performance computing capabilities, such as job scheduling, heterogeneous chip management, and job running management.

Volcano Scheduler

Volcano Scheduler is a pod scheduling component, which consists of a series of actions and plugins. Actions should be executed in every step. Plugins provide the action algorithm details in different scenarios. Volcano Scheduler features high scalability. You can specify actions and plugins as needed.

Figure 6-16 Volcano Scheduler workflow



The working process of Volcano Scheduler is as follows:

1. Identify and cache the job submitted by the client.
2. Start a periodical session. A scheduling cycle begins.
3. Send jobs that are not scheduled the to-be-scheduled queue of the session.
4. Traverse all jobs to be scheduled and perform actions such as enqueue, allocate, preempt, reclaim, and backfill in the configured sequence to find the most appropriate node for each job. Bind the job to the node. The specific algorithm logic executed in **action** depends on the implementation of each function in the registered plugin.
5. Close the session.

Custom Volcano Resources

- A pod group is a custom Volcano resource type. It is a group of pods with strong association and is mainly used in batch scheduling, for example, ps and worker tasks in TensorFlow.
- A queue contains a group of PodGroups. It is also the basis for the PodGroups to obtain cluster resources.
- Volcano Job (vcjob for short) is a custom job resource type. Different from Kubernetes Jobs, vcjob supports specified scheduler, the minimum number of running pods, tasks, lifecycle management, specified queues, and priority-based scheduling. Volcano Job is more suitable for high-performance computing scenarios such as machine learning, big data, and scientific computing.
- Application scaling priority policy: After the Volcano-backed application scaling priority policy is enabled, the **BalancerPolicyTemplate** and **Balancer** CRD resources will be added to the cluster. **BalancerPolicyTemplate** is used to establish the priority policy, while **Balancer** is employed to specify the scaling priority's scope. A **Balancer** CR corresponds to a **BalancerPolicyTemplate** CR. They work together to determine which priority policies are applied to specific workloads. For details, see **Application Scaling Priority Policies**.

6.5.2 Scheduling Workloads

Volcano is a Kubernetes-based batch processing platform with high-performance general computing capabilities like task scheduling engine, heterogeneous chip

management, and task running management. It provides end users with computing frameworks from multiple domains such as AI, big data, gene, and rendering. It also offers job scheduling, job management, and queue management for computing applications.

Kubernetes typically uses its default scheduler to schedule workloads. To use Volcano, specify Volcano for your workloads. For details about the Kubernetes scheduler, see **Specify schedulers for pods**.

Notes and Constraints

If you schedule a lot of workloads, Volcano will generate a significant number of logs. To prevent the node hosting Volcano from running out of disk space, use Volcano with LTS.

Using Volcano

When using Volcano to schedule workloads, you only need to configure **schedulerName** in the **spec** field of the pod and set the parameter to **volcano**. The following is an example:

1. Use YAML to create a queue.

```
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: q1
spec:
  reclaimable: true
  weight: 1
```

2. Set **schedulerName** in the **spec** field of the pod to **volcano**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        # Submit the job to the q1 queue.
        scheduling.volcano.sh/queue-name: "q1"
        volcano.sh/preemptable: "true"
      labels:
        app: nginx
    spec:
      # Specify Volcano as the scheduler.
      schedulerName: volcano
      containers:
      - name: nginx
        image: nginx
        imagePullPolicy: IfNotPresent
        resources:
          limits:
            cpu: 1
            memory: 100Mi
          requests:
            cpu: 1
            memory: 100Mi
```



```
    ports:
    - containerPort: 80
```

Additionally, Volcano supports the workload queues and preemption, which can be implemented through pod annotations. The following table lists the supported annotations.

Table 6-14 Pod annotations supported by Volcano

Pod Annotations	Description
scheduling.volcano.sh/queue-name: "<queue-name>"	Specifies the queue to which the workload belongs. *<queue-name>* indicates the queue name.
volcano.sh/preemptable: "true"	Indicates whether a job can be preempted. If this function is enabled, the job can be preempted. Options: • **true**: Preemption is enabled. This option is enabled by default. • **false**: Preemption is disabled.

You can obtain pod details to check whether the pod is scheduled by Volcano to the allocated queue.

1. Run the following command to check the pod details and obtain the **scheduling.k8s.io/group-name** value:
   ```
   kubectl describe pod <pod_name>
   ```
   The **scheduling.k8s.io/group-name** value of the pod is displayed.
   ```
   Name:             npu-pod
   Namespace:        default
   Priority:         0
   Service Account:  default
   Node:             <none>
   Labels:           <none>
   Annotations:      scheduling.k8s.io/group-name: podgroup-92d58e7e-02ee-471c-9e50-4ed810aaebb1
   Status:           Pending
   IP:
   ```
2. Check whether the pod is scheduled by Volcano to the allocated queue.
   ```
   kubectl describe pg <group_name>
   ```
   Command output:
   ```
   Spec:
     Min Member:  1
     Min Resources:
       Cpu:     100m
       Memory:  100Mi
     Queue:     q1
   Status:
     Conditions:
       Last Transition Time:  2023-05-30T01:54:43Z
       Reason:                tasks in gang are ready to be scheduled
       Status:                True
       Transition ID:         70be1d7d-3532-41e0-8324-c7644026b38f
       Type:                  Scheduled
     Phase:                   Running
   Events:
     Type    Reason     Age              From     Message
   ```



```
----    ------    ----          ----    -------
Normal  Scheduled  0s (x3 over 2s)  volcano  pod group is ready
```

6.5.3 Resource Usage-based Scheduling

6.5.3.1 Bin Packing

Bin packing is an optimization algorithm that aims to properly allocate resources to each job and get the jobs done using the minimum amount of resources. After bin packing is enabled for cluster workloads, the scheduler preferentially schedules pods to nodes with high resource allocation. This reduces resource fragments on each node and improves cluster resource utilization.

Prerequisites

- A cluster of v1.19 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **Volcano Scheduler**.

Features

Bin packing aims to fill as many existing nodes as possible (try not to allocate blank nodes). In the concrete implementation, the bin packing algorithm scores the nodes that can be delivered, and a higher score indicates a higher resource utilization rate of nodes. Bin packing intends to centrally schedule application workloads onto some nodes in a cluster, which facilitates auto scaling of cluster nodes.

The bin packing add-on works with other scheduling add-ons of the scheduler to score nodes. You can customize the overall weight of the add-on and the weight of each resource to modify the influence in the node score. When using bin packing to calculate node scores, the scheduler considers extended resources such as CPUs, memory, and GPUs requested by pods, and calculates the scores based on the weights configured for each resource.

How It Works

A node is scored based on the overall weight of the bin packing add-on and the weight of each resource. Each type of resource requested by the to-be-scheduled pods is scored. Take CPUs as an example, the CPU score is calculated using the following formula:

CPU.weight x (Requested + Used)/Allocatable

A larger CPU weight leads to a higher score. A higher resource usage of a node leads to a higher node score. The same rule applies to memory and GPU resources. The parameters in the formula for scoring a resource are as follows:

- **CPU.weight**: customized CPU weight
- **Requested**: CPU resources requested by the pods to be scheduled
- **Used**: CPU resources that have been used on the current node
- **Allocatable**: total CPU resources available on the current node

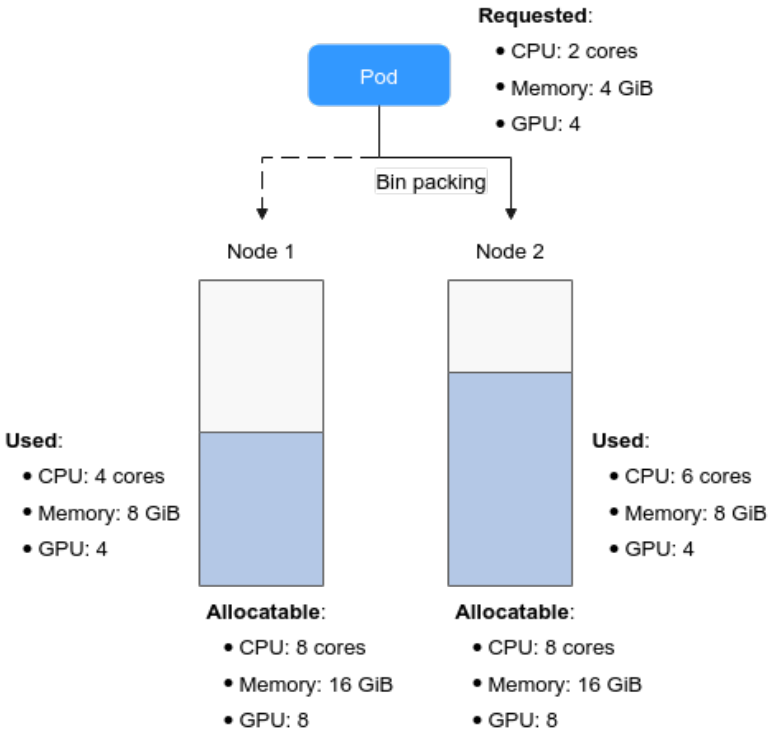
The bin packing add-on calculates the score of a node using the following formula:

Binpack.weight x (CPU.score + Memory.score + GPU.score)/(CPU.weight + Memory.weight + GPU.weight) x 100

A larger bin packing weight leads to a higher score. A larger resource weight leads to a greater influence in the node score. The parameters in the formula for scoring a node are as follows:

- **Binpack.weight**: bin packing weight
- **CPU.score**: calculated CPU score; **CPU.weight**: customized CPU weight
- **Memory.score**: calculated memory score; **Memory.weight**: customized memory weight
- **GPU.score**: calculated GPU score; **GPU.weight**: customized GPU weight

Figure 6-17 Bin packing example



In this figure, there are two nodes in the cluster. When pods need to be scheduled, the bin packing policy scores the two nodes separately.

For example, **CPU.weight** is set to **1**, **Memory.weight** is set to **1**, **GPU.weight** is set to **2**, and **binpack.weight** is set to **5** in a cluster.

1. Scoring for node 1

   The score of each resource type is calculated using the following formulas:

   - CPU: CPU.weight x (Requested + Used)/Allocatable = 1 x (2 + 4)/8 = 0.75
   - Memory: Memory.weight x (Requested + Used)/Allocatable = 1 x (4 + 8)/16 = 0.75

- GPU: GPU.weight x (Requested + Used)/Allocatable = 2 x (4 + 4)/8 = 2

The total score of each node is calculated using the following formula: Binpack.weight x (CPU.score + Memory.score + GPU.score)/(CPU.weight + Memory.weight + GPU.weight) x 100

If **binpack.weight** is set to **5**, the score of node 1 is calculated as follows: 5 x (0.75 + 0.75 + 2)/(1 + 1 + 2) x 100 = 437.5

2. Scoring for node 2
   - CPU: CPU.weight x (Requested + Used)/Allocatable = 1 x (2 + 6)/8 = 1
   - Memory: Memory.weight x (Requested + Used)/Allocatable = 1 x (4 + 8)/16 = 0.75
   - GPU: GPU.weight x (Requested + Used)/Allocatable = 2 x (4 + 4)/8 = 2

   Score of node 2: 5 x (1 + 0.75 + 2)/(1 + 1 + 2) x 100 = 468.75

The calculation results show that the score of node 2 is greater than that of node 1. According to the bin packing policy, new pods will be preferentially scheduled to node 2.

Configuring Bin Packing

After Volcano is installed, bin packing takes effect by default. If the default configuration cannot meet your requirements, you can customize the weight of bin packing and the weight of each resource on the **Scheduling** page. To do so, perform the following operations:

Step 1 Log in to the CCE console.

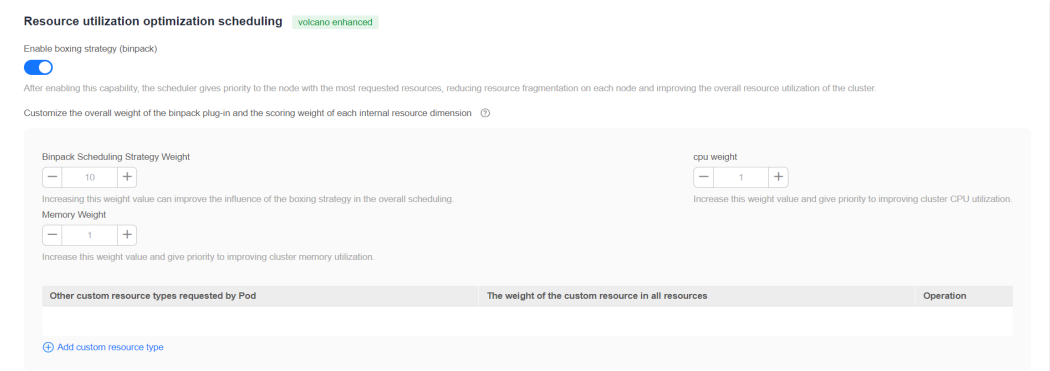
Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

Step 3 In the **Resource utilization optimization scheduling** configuration, enable bin packing.

Table 6-15 Bin packing weights

Item	Description	Default Value
Binpack Scheduling Strategy Weight	A larger value indicates a higher weight of the bin packing policy in overall scheduling.	10
CPU Weight	A larger value indicates a higher cluster CPU usage.	1
Memory Weight	A larger value indicates a higher cluster memory usage.	1
Custom Resource Type	Other custom resource types requested by pods, for example, **nvidia.com/gpu**. A larger value indicates a higher usage of the specified cluster resource.	None

Figure 6-18 Resource utilization optimization scheduling



Step 4 Click **Confirm**.

----End

6.5.3.2 Descheduling

Scheduling in a cluster is the process of binding pending pods to nodes, and is performed by a component called kube-scheduler or Volcano Scheduler. The scheduler uses a series of algorithms to compute the optimal node for running pods. However, Kubernetes clusters are dynamic and their state changes over time. For example, if a node needs to be maintained, all pods on the node will be evicted to other nodes. After the maintenance is complete, the evicted pods will not automatically return back to the node because descheduling will not be triggered once a pod is bound to a node. Due to these changes, the load of a cluster may be unbalanced after the cluster runs for a period of time.

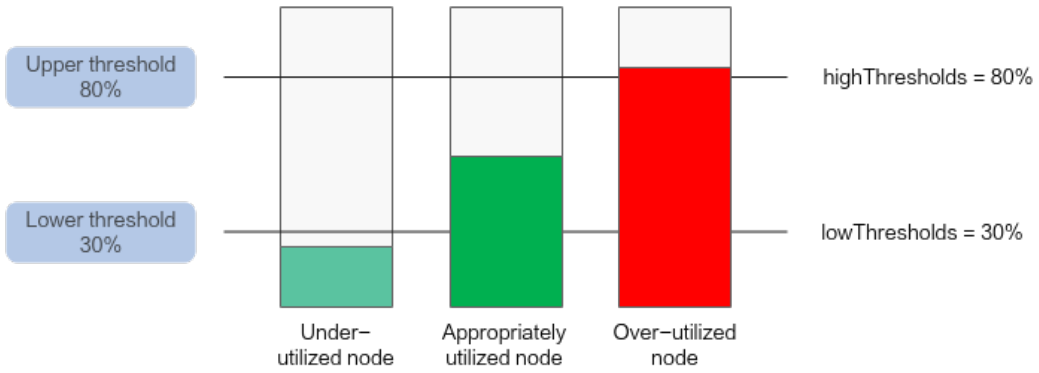
CCE has resolved this issue by using Volcano Scheduler to evict pods that do not comply with the configured policy so that pods can be rescheduled. In this way, the cluster load is balanced and resource fragmentation is minimized.

Features

Load-aware Descheduling

During Kubernetes cluster management, over-utilized nodes are due to high CPU or memory usage, which affects the stable running of pods on these nodes and increases the probability of node faults. To dynamically balance the resource usage between nodes in a cluster, a cluster resource view is required based on node monitoring metrics. During cluster management, real-time monitoring can be used to detect issues such as high resource usage on a node, node faults, and excessive number of pods on a node so that the system can take measures promptly, for example, by migrating some pods from an over-utilized node to under-utilized nodes.

Figure 6-19 Load-aware descheduling



When using this add-on, ensure the **highThresholds** value is greater than the **lowThresholds** value. Otherwise, the descheduler cannot work.

- **Appropriately utilized node**: a node whose resource usage is greater than or equal to 30% and less than or equal to 80%. The resource usage of appropriately utilized nodes is within the expected range.
- **Over-utilized node**: a node whose resource usage is higher than 80%. Some pods will be evicted from over-utilized nodes to reduce its resource usage to be less than or equal to 80%. The descheduler will schedule the evicted pods to under-utilized nodes.
- **Under-utilized node**: a node whose resource usage is lower than 30%.

HighNodeUtilization

This policy finds nodes that are under-utilized and evicts pods from the nodes in the hope that these pods will be scheduled compactly into fewer nodes. This policy must be used with the bin packing policy of Volcano Scheduler or the MostAllocated policy of the kube-scheduler scheduler. Thresholds can be configured for CPU and memory.

Prerequisites

- A cluster of v1.19.16 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see **Volcano Scheduler**.

Notes and Constraints

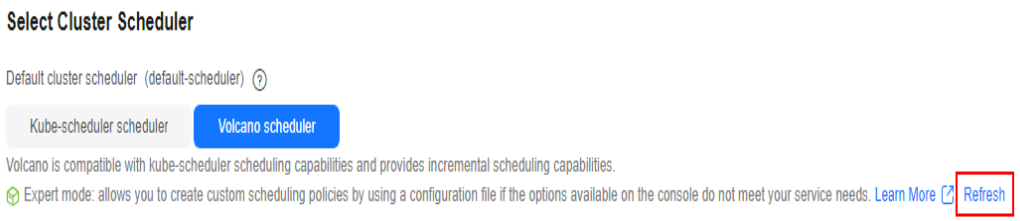
- Pods need to be rescheduled using a scheduler, and no scheduler can label pods or nodes. Therefore, an evicted pod might be rescheduled to the original node.
- Descheduling does not support anti-affinity between pods. An evicted pod is in anti-affinity relationship with other running pods. Therefore, the scheduler may still schedule the pod back to the node where the pod was evicted from.
- When configuring load-aware descheduling, you are required to enable load-aware scheduling on Volcano Scheduler. When configuring HighNodeUtilization, you are required to enable bin packing on Volcano Scheduler.

Configuring a Load-aware Descheduling Policy

When configuring a load-aware descheduling policy, do as follows to enable load-aware scheduling on Volcano Scheduler:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab on the right side of the page. Then, enable load-aware scheduling.

Step 2 On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.



Step 3 Configure a load-aware descheduling policy. The following shows a configuration example in JSON format for Volcano v1.11.21 or later:

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "enablePreemptable": false,
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          },
          {
            "name": "usage",
            "enablePredicate": true,
            "arguments": {
              "usage.weight": 5,
              "cpu.weight": 1,
              "memory.weight": 1,
              "thresholds": {
                "cpu": 80,
                "mem": 80
```



```
          }
         }
        }
       ]
     },
     {
       "plugins": [
        {
          "name": "cce-gpu-topology-predicate"
        },
        {
          "name": "cce-gpu-topology-priority"
        },
        {
          "name": "cce-gpu"
        }
       ]
     },
     {
       "plugins": [
        {
          "name": "nodelocalvolume"
        },
        {
          "name": "nodeemptydirvolume"
        },
        {
          "name": "nodeCSIscheduling"
        },
        {
          "name": "networkresource"
        }
       ]
     }
   ]
},
"deschedulerPolicy": {
  "profiles": [
    {
      "name": "ProfileName",
      "pluginConfig": [
        {
          "args": {
            "ignorePvcPods": true,
            "nodeFit": true,
            "priorityThreshold": {
              "value": 100
            }
          },
          "name": "DefaultEvictor"
        },
        {
          "args": {
            "evictableNamespaces": {
              "exclude": ["kube-system"]
            },
            "metrics": {
              "type": "prometheus_adaptor"
            },
            "targetThresholds": {
              "cpu": 80,
              "memory": 85
            },
            "thresholds": {
              "cpu": 30,
              "memory": 30
            }
          },
          "name": "LoadAware"
```



```
        }
       ],
       "plugins": {
        "balance": {
          "enabled": ["LoadAware"]
        }
       }
      }
     ]
    },
    "descheduler_enable": "true",
    "deschedulingInterval": "10m"
}
```

Table 6-16 Key parameters of a cluster descheduling policy

Parameter	Description
descheduler_enable	Whether to enable a cluster descheduling policy. • **true**: The cluster descheduling policy is enabled. • **false**: The cluster descheduling policy is disabled.
deschedulingInterval	Descheduling period.
deschedulerPolicy	Cluster descheduling policy. For details, see **Table 6-17**.

Table 6-17 deschedulerPolicy parameters

Parameter	Description
profiles.[].plugins.balance.enable.[]	Descheduling policy for a cluster. **LoadAware**: a load-aware descheduling policy is used.
profiles.[].pluginConfig.[].name	Configuration of a load-aware descheduling policy. Options: • **DefaultEvictor**: default eviction policy • **LoadAware**: a load-aware descheduling policy

Parameter	Description

profiles.
[].pluginConfig.
[].args
	Descheduling policy configuration of a cluster.
	• Configurations for the **DefaultEvictor** policy:
	– **ignorePvcPods**: whether PVC pods should be ignored or evicted. Value **true** indicates that the pods are ignored, and value **false** indicates that the pods are evicted. This configuration does not differentiate PVC types (local PVs, SFS, or EVS).
	– **nodeFit**: whether to consider the existing scheduling configurations such as node affinity and taint on the node during descheduling. Value **true** indicates that the existing scheduling configurations will be considered, and value **false** indicates that those will be ignored.
	– **priorityThreshold**: priority setting. If the priority of a pod is greater than or equal to the value of this parameter, the pod will not be evicted. Example:
	`{`
	`"value": 100`
	`}`
	• Configurations for the **LoadAware** policy:
	– **evictableNamespaces**: namespaces where the eviction policy takes effect. The default value is the namespaces other than kube-system. Example:
	`{`
	`"exclude": ["kube-system"]`
	`}`
	– **metrics**: how monitoring data is obtained. Either the Custom Metrics API (prometheus_adaptor) or Prometheus can be used.
	For Volcano 1.11.17 and later versions, use Custom Metrics API to obtain monitoring data. The following is an example:
	`{`
	`"type": "prometheus_adaptor"`
	`}`
	For Volcano 1.11.5 to 1.11.16, use Prometheus to obtain monitoring data. You need to enter the IP address of the Prometheus server. The following is an example:
	`{`
	`"address": "http://10.247.119.103:9090",`
	`"type": "prometheus"`
	`}`
	– **targetThresholds**: threshold for evicting pods from a node. When the CPU or memory usage of a node is greater than the threshold, the pods on the node will be evicted. Example:
	`{`
	`"cpu": 60,`

Parameter	Description

	"memory": 65
	}
	– **thresholds**: threshold for a node to run pods. When the CPU or memory usage of a node is less than the threshold, the node allows evicted pods to run. Example:
	{
	"cpu": 30,
	"memory": 30
	}

Step 4 Click **OK**.

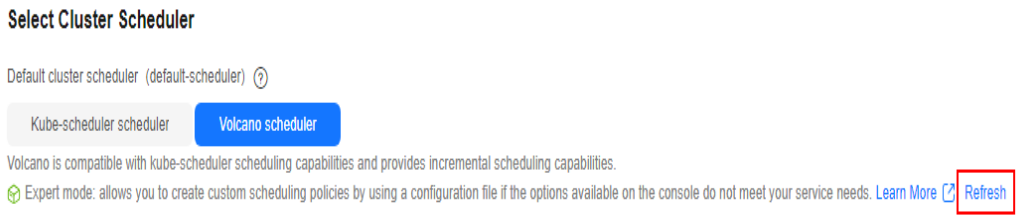
----End

Configuring a HighNodeUtilization Policy

When configuring a HighNodeUtilization policy, do as follows to enable the bin packing policy on Volcano Scheduler:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab on the right side of the page. Then, enable bin packing. For details, see **Bin Packing**.

Step 2 On the **Scheduling** tab page, select **Volcano scheduler**, find the expert mode, and click **Refresh**.



Step 3 Configure a resource defragmentation policy. The following shows a configuration example in JSON format:

```
{
  "colocation_enable": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "enablePreemptable": false,
            "name": "gang"
          },
          {
            "name": "conformance"
          },
          {
            "arguments": {
```



```
          "binpack.weight": 5
         },
         "name": "binpack"
       }
     ]
   },
   {
     "plugins": [
       {
         "enablePreemptable": false,
         "name": "drf"
       },
       {
         "name": "predicates"
       },
       {
         "name": "nodeorder"
       }
     ]
   },
   {
     "plugins": [
       {
         "name": "cce-gpu-topology-predicate"
       },
       {
         "name": "cce-gpu-topology-priority"
       },
       {
         "name": "cce-gpu"
       }
     ]
   },
   {
     "plugins": [
       {
         "name": "nodelocalvolume"
       },
       {
         "name": "nodeemptydirvolume"
       },
       {
         "name": "nodeCSIscheduling"
       },
       {
         "name": "networkresource"
       }
     ]
   }
 ]
},
"deschedulerPolicy": {
  "profiles": [
    {
      "name": "ProfileName",
      "pluginConfig": [
        {
          "args": {
            "ignorePvcPods": true,
            "nodeFit": true,
            "priorityThreshold": {
              "value": 100
            }
          },
          "name": "DefaultEvictor"
        },
        {
          "args": {
            "evictableNamespaces": {
```



```
            "exclude": ["kube-system"]
           },
           "thresholds": {
            "cpu": 25,
            "memory": 25
           }
          },
          "name": "HighNodeUtilization"
         }
        ],
        "plugins": {
         "balance": {
          "enabled": ["HighNodeUtilization"]
         }
        }
       }
      ]
    },
    "descheduler_enable": "true",
    "deschedulingInterval": "10m"
}
```

Table 6-18 Key parameters of a cluster descheduling policy

Parameter	Description
descheduler_enable	Whether to enable a cluster descheduling policy. • **true**: The cluster descheduling policy is enabled. • **false**: The cluster descheduling policy is disabled.
deschedulingInterval	Descheduling period.
deschedulerPolicy	Cluster descheduling policy. For details, see **Table 6-19**.

Table 6-19 deschedulerPolicy parameters

Parameter	Description
profiles.[].plugins.balance.enable.[]	Descheduling policy for a cluster. **HighNodeUtilization**: the policy for minimizing CPU and memory fragments is used.
profiles.[].pluginConfig.[].name	Configuration of a load-aware descheduling policy. Options: • **DefaultEvictor**: default eviction policy • **HighNodeUtilization**: policy for minimizing CPU and memory fragments

Parameter	Description

profiles.
[].pluginConfig.
[].args
	Descheduling policy configuration of a cluster.
	• Configurations for the **DefaultEvictor** policy:
	– **ignorePvcPods**: whether PVC pods should be ignored or evicted. Value **true** indicates that the pods are ignored, and value **false** indicates that the pods are evicted. This configuration does not differentiate PVC types (local PVs, SFS, or EVS).
	– **nodeFit**: whether to consider the existing scheduling configurations such as node affinity and taint on the node during descheduling. Value **true** indicates that the existing scheduling configurations will be considered, and value **false** indicates that those will be ignored.
	– **priorityThreshold**: priority setting. If the priority of a pod is greater than or equal to the value of this parameter, the pod will not be evicted. Example:
	`{`
	`"value": 100`
	`}`
	• Configurations for the **HighNodeUtilization** policy:
	– **evictableNamespaces**: namespaces where the eviction policy takes effect. The default value is the namespaces other than kube-system. Example:
	`{`
	`"exclude": ["kube-system"]`
	`}`
	– **thresholds**: threshold for evicting pods from a node. When the CPU or memory usage of a node is less than the threshold, the pods on the node will be evicted. Example:
	`{`
	`"cpu": 25,`
	`"memory": 25`
	`}`

Step 4 Click **OK**.

----End

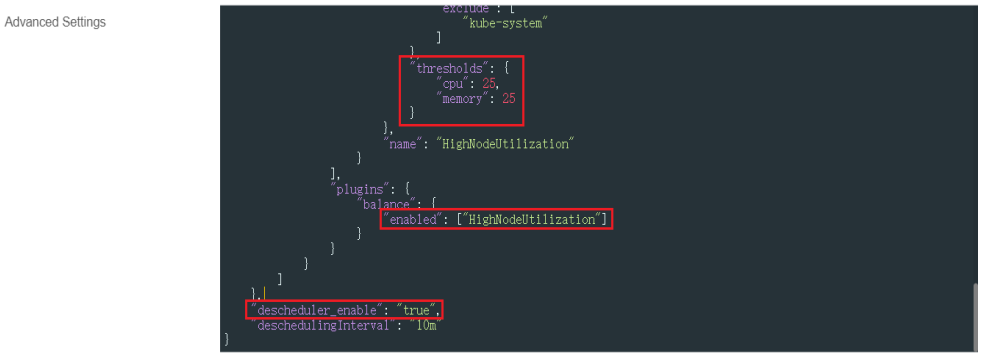
Use Cases

HighNodeUtilization

1. Check the nodes in a cluster. It is found that some nodes are under-utilized.

192.168.44.152 (Private)	1 / 40	0.51% 0.51%	0% 0%
192.168.54.65 (Private)	6 / 40	26.53% 26.53%	33.93% 33.93%

2. Edit the Volcano parameters to enable the descheduler and set the CPU and memory usage thresholds to **25**. When the CPU and memory usage of a node is less than 25%, pods on the node will be evicted.

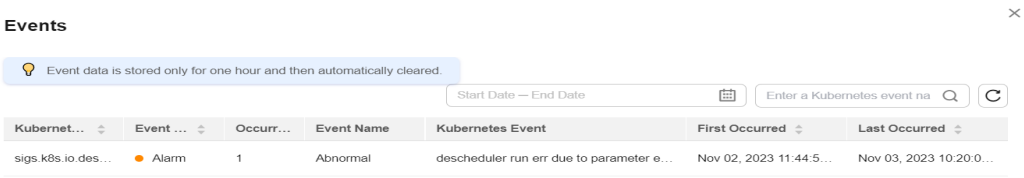


3. After the policy takes effect, pods on the node with IP address 192.168.44.152 will be migrated to the node with IP address 192.168.54.65 for minimized resource fragments.

192.168.44.152 (Private)	0 / 40	0% 0%	0% 0%
192.168.54.65 (Private)	7 / 40	27.04% 27.04%	33.93% 33.93%

Common Issues

If an input parameter is incorrect, for example, the entered value is beyond the accepted value range or in an incorrect format, an event will be generated. In this case, modify the parameter setting as prompted.



6.5.3.3 Node Pool Affinity

In scenarios such as node pool replacement and rolling node upgrade, an old resource pool needs to be replaced with a new one. To prevent the node pool replacement from affecting services, enable soft affinity to schedule service pods to the new node pool. Soft affinity scheduling tries to schedule newly created pods or rescheduled pods to the new node pool. If the pods cannot be scheduled to the new node pool, for example, due to insufficient resources, the pods can also be scheduled to the old node pool. Since a node pool replacement should not affect

services, the node affinity configuration is not declared in service workloads. Use soft affinity in cluster scheduling to schedule pods to new node pools when a pool replacement is triggered.

Volcano aims to soft schedule service pods to specified nodes when node soft affinity is not configured on service workloads.

Scheduling Priority

Soft affinity scheduling of a node pool is implemented based on labels in the node pool. Each node in the node pool is scored to select the optimal one for pod scheduling.

The rule is to schedule pods to nodes with specified labels as far as possible.

The scoring formula is as follows:

Node score = Weight x MaxNodeScore x haveLabel

Parameters:

- **Weight**: weight of the soft affinity add-on in the node pool.
- **MaxNodeScore**: maximum score (100) of a node.
- **haveLabel**: whether the labels configured in the add-on are available on a node. If yes, the value is **1**. If no, the value is **0**.

Prerequisites

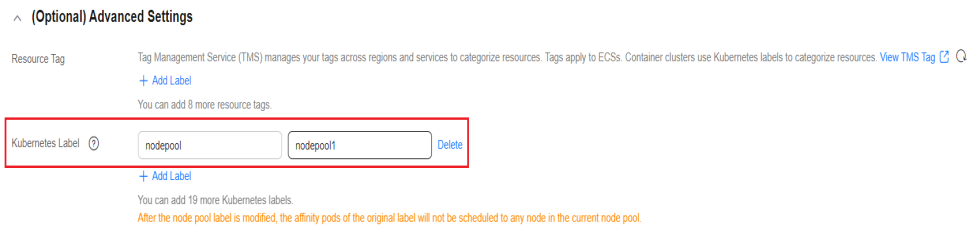
- A cluster of v1.19.16 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see **Volcano Scheduler**.

Configuring Soft Affinity Scheduling for Volcano Node Pools

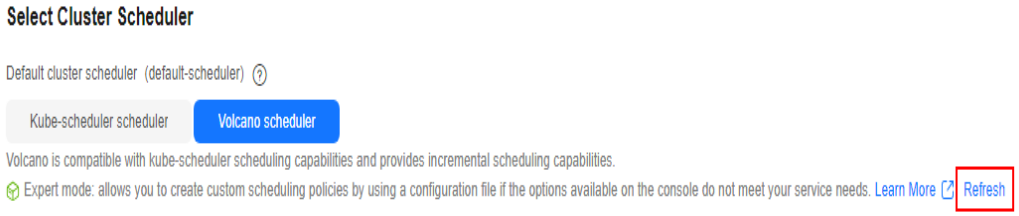
Step 1 Configure labels for affinity scheduling in the node pool.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.
3. Click **Update** of the target node pool. On the page that is displayed, configure labels in the **Kubernetes Label** area.

   The following is an example.



Step 2 In the navigation pane, choose **Settings** and click the **Scheduling** tab. Select **Volcano scheduler**, find the expert mode, and click **Refresh**.



Step 3 Configure Volcano scheduler parameters. The following shows a configuration example in JSON format:

```
...
  "default_scheduler_conf": {
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          },
          {
            // Enable node pool affinity scheduling.
            "name": "nodepoolaffinity",
            // Configure the affinity scheduling weight and labels of the node pool.
            "arguments": {
              "nodepoolaffinity.weight": 10000,
              "nodepoolaffinity.label": "nodepool1=nodepool1"
            }
          }
        ]
      },
      {
        "plugins": [
          {
```



```
                "name": "nodelocalvolume"
            },
            {
                "name": "nodeemptydirvolume"
            },
            {
                "name": "nodeCSIscheduling"
            },
            {
                "name": "networkresource"
            }
        ]
    }
  ]
},
...
```

Step 4 Click **OK**.

----End

6.5.3.4 Load-aware Scheduling

Volcano Scheduler offers CPU and memory load-aware scheduling for pods and preferentially schedules pods to the node with the lightest load to balance node loads. This prevents an application or node failure due to heavy loads on a single node.

Prerequisites

- A cluster of v1.21 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on of v1.11.14 or later has been installed. For details, see **Volcano Scheduler**.
- The Cloud Native Cluster Monitoring add-on (kube-prometheus-stack) has been installed and it runs in local data storage mode. For details, see **Cloud Native Cluster Monitoring**. If you use an on-premises Prometheus monitoring system, Prometheus 2.35.0 or later is recommended. For details, see **Using Self-Built Prometheus to Configure Load-aware Scheduling**.
- Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Features

The native Kubernetes scheduler schedules resources only based on requested resources. However, the actual resource usage of a pod differs greatly from the requested or limited value of the requested resources, which is the cause of cluster load imbalance.

1. The actual resource usage of certain nodes in a cluster is far lower than the resource allocation rate, but no more pods are scheduled onto the nodes, leading to resource waste.
2. The scheduler failed to detect that some nodes in the cluster are overloaded, which could significantly affect the stability of the service.

Volcano resolves the preceding issues based on actual loads. If there are plenty of resources, pods are preferentially scheduled to nodes with the lightest load to balance the load on each node in the cluster.

The status, workload traffic, and requests of a cluster change dynamically, and the resource usage of nodes changes in real time. To prevent extreme load imbalance in a cluster after pod scheduling, Volcano provides load-aware hotspot descheduling for the optimal load balancing of cluster nodes. For details about hotspot descheduling, see **Descheduling**.

How It Works

Load-aware scheduling is implemented using both Volcano and the CCE cloud native monitoring add-on (kube-prometheus-stack). After load-aware scheduling is enabled, metrics such as CPU and memory loads are defined by following Prometheus adapter rules. Then, the kube-prometheus-stack add-on collects and saves the actual CPU and memory loads of each node based on the defined metric rules. Volcano scores and sorts nodes based on the metric values provided by the kube-prometheus-stack add-on and preferentially schedules pods to the node with the lightest load.

Load-aware scheduling scores each node using the weighted average of the CPU and memory metrics as well as the load-aware scheduling policy and preferentially selects the node with the highest score for scheduling. You can create custom weights for the CPU, memory, and load-aware scheduling policy on the **Scheduling** tab by choosing **Settings** in the navigation pane of the target cluster.

The formula for scoring a node is as follows: Weight of the load-aware scheduling policy × [(1 - CPU usage) x CPU weight + (1 - Memory usage) × Memory weight]/(CPU weight + Memory weight)

- CPU usage: average CPU usage of all nodes in the target cluster in the last 10 minutes (The collection frequency can be modified in the Prometheus adapter rule.)
- Memory usage: average memory usage of all nodes in the target cluster in the last 10 minutes

Using Cloud Native Cluster Monitoring

Step 1 After installing the Cloud Native Cluster Monitoring add-on, you need to enable Metrics API to provide container resource metrics such as CPU usage and memory usage.

NOTE

Resource metrics can be provided by Metrics API only when local data storage is enabled for the Cloud Native Cluster Monitoring add-on.

Check whether Metrics API has been enabled for the cluster. If it is enabled, you can skip this step.

```
kubectl get APIServices | grep v1beta1.metrics.k8s.io
```

If any command output is displayed, Metrics API is enabled. Skip this step and go to the next step to add metric collection rules.

If no Metrics API is found, you can manually create an APIService object to enable it.

1. Create a file named **metrics-apiservice.yaml**. Example file content:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

2. Create an APIService object.

```
kubectl create -f metrics-apiservice.yaml
```

3. Check whether Metrics API is enabled for the cluster.

```
kubectl get APIServices | grep v1beta1.metrics.k8s.io
```

> **NOTE**
>
> After Metrics API is enabled, if you need to uninstall the Cloud Native Cluster Monitoring add-on, run the following kubectl and delete the APIService object. Otherwise, the residual APIService resources will cause the Kubernetes Metrics Server add-on to fail to be installed.
>
> ```
> kubectl delete APIService v1beta1.metrics.k8s.io
> ```

Step 2 Add collection rules for custom metrics.

1. Modify the **user-adapter-config** configuration item.

```
kubectl edit configmap user-adapter-config -n monitoring
```

2. Add collection rules to the **rules** field.

   In the following example rules, the rules in red are new ones and those in black are existing ones:

```
...
data:
  config.yaml: >
    rules:
    - seriesQuery: '{__name__=~"node_cpu_seconds_total"}'
      resources:
        overrides:
          instance:
            resource: node
      name:
        matches: node_cpu_seconds_total
        as: node_cpu_usage_avg
      metricsQuery: avg_over_time((1 - avg (irate(<<.Series>>{mode="idle"}[5m])) by (instance))
[10m:30s])
    - seriesQuery: '{__name__=~"node_memory_MemTotal_bytes"}'
      resources:
        overrides:
          instance:
            resource: node
      name:
        matches: node_memory_MemTotal_bytes
        as: node_memory_usage_avg
      metricsQuery: avg_over_time(((1-node_memory_MemAvailable_bytes/<<.Series>>))[10m:30s])
    resourceRules:
      cpu:
        containerQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>,container!
="",pod!=""}[1m])) by (<<.GroupBy>>)
```



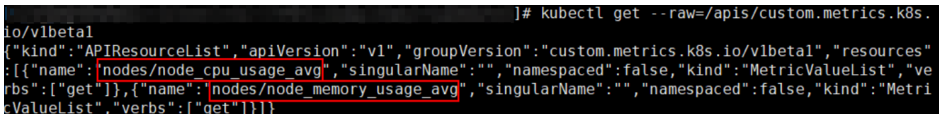
```
      nodeQuery: sum(rate(container_cpu_usage_seconds_total{<<.LabelMatchers>>, id='/'}[1m])) by
(<<.GroupBy>>)
      resources:
        overrides:
          instance:
            resource: node
          namespace:
            resource: namespace
          pod:
            resource: pod
      containerLabel: container
    memory:
      containerQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>,container!
="",pod!=""}) by (<<.GroupBy>>)
      nodeQuery: sum(container_memory_working_set_bytes{<<.LabelMatchers>>,id='/'}) by
(<<.GroupBy>>)
      resources:
        overrides:
          instance:
            resource: node
          namespace:
            resource: namespace
          pod:
            resource: pod
      containerLabel: container
    window: 1m
...
```

- **Rules for collecting the average CPU usage**
  - **node_cpu_usage_avg**: average CPU usage of nodes. The name of this metric cannot be changed.
  - **metricsQuery: avg_over_time((1 - avg (irate(<<.Series>>{mode="idle"}[5m])) by (instance))[10m:30s])**: statement for obtaining nodes' average CPU usage.

    metricsQuery indicates to obtain the average CPU usage of all nodes in the target cluster in the last 10 minutes. To change the period, for example, to the last 5 or 30 minutes, change **10m** in red to **5m** or **30m**.
- **Rules for collecting the average memory usage**
  - **node_memory_usage_avg**: average memory usage of nodes. The name of this metric cannot be changed.
  - **metricsQuery: avg_over_time(((1- node_memory_MemAvailable_bytes/<<.Series>>))[10m:30s])**: statement for obtaining nodes' average memory usage.

    metricsQuery indicates to obtain the average memory usage of all nodes in the target cluster in the last 10 minutes. To change the period, for example, to the last 5 or 30 minutes, change **10m** in red to **5m** or **30m**.

3. Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.
   ```
   kubectl rollout restart deployment custom-metrics-apiserver -n monitoring
   ```
4. Verify that the custom rules are configured successfully.
   a. Run the following command. If the custom metric information is returned, the metric collection configuration on Prometheus is successful.
      ```
      kubectl get --raw=/apis/custom.metrics.k8s.io/v1beta1
      ```

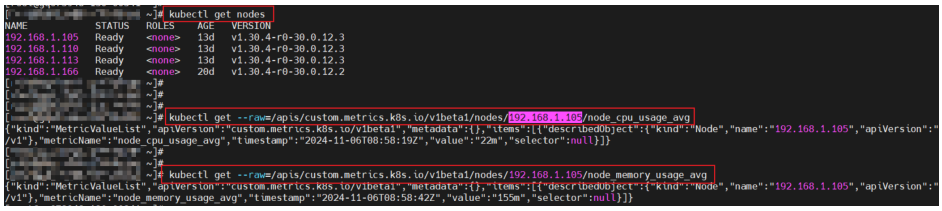
b. Query information about nodes in the cluster.

```
kubectl get nodes
```

Run the following command on any node. Replace *xxxx* with the obtained value of **node_name**. If you want to query the resource information of all nodes, replace *xxxx* with an asterisk (*).

```
kubectl get --raw=/apis/custom.metrics.k8s.io/v1beta1/nodes/xxxx/node_cpu_usage_avg
```

Information similar to the following is displayed.



Step 3 Enable load-aware scheduling.

After Volcano is installed, you can enable or disable load-aware scheduling on the **Scheduling** page by choose **Settings** in the navigation pane. This function is disabled by default.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.
3. In the **Resource utilization optimization scheduling** area, modify the load-aware scheduling settings.

   NOTE

   For optimal load-aware scheduling, disable bin packing because this policy preferentially schedules pods to the node with the maximal resources allocated based on pods' requested resources. This affects load-aware scheduling to some extent. For details about the combination of multiple policies, see **Configuration Cases for Resource Usage-based Scheduling**.

Parameter	Description	Default Value
Load-Aware Scheduling Policy Weight	A larger value indicates a higher weight of the load-aware policy in overall scheduling.	5
CPU Weight	A larger value indicates CPU resources will be preferentially balanced.	1
Memory Weight	A larger value indicates memory resources will be preferentially balanced.	1

Parameter	Description	Default Value
Actual load threshold effective mode	– Soft constraint: When the actual CPU or memory load of a node reaches the threshold, new tasks will be preferentially allocated to underutilized nodes, but this node can still be scheduled. – Hard constraint: When the actual CPU or memory load of a node reaches the threshold, no new tasks can be scheduled to this node.	Hard constraint
Actual CPU Load Threshold	When a node's CPU usage goes beyond this threshold, workloads will be scheduled based on how the load threshold takes effect. New workloads will be preferentially or forcibly scheduled to other nodes. Existing workloads on the nodes are not affected.	80
Actual Memory Load Threshold	When a node's memory usage goes beyond this threshold, workloads will be scheduled based on how the load threshold takes effect. New workloads will be preferentially or forcibly scheduled to other nodes. Existing workloads on the nodes are not affected.	80

----End

Using Self-Built Prometheus to Configure Load-aware Scheduling

Step 1 Install prometheus-adapter in the cluster.

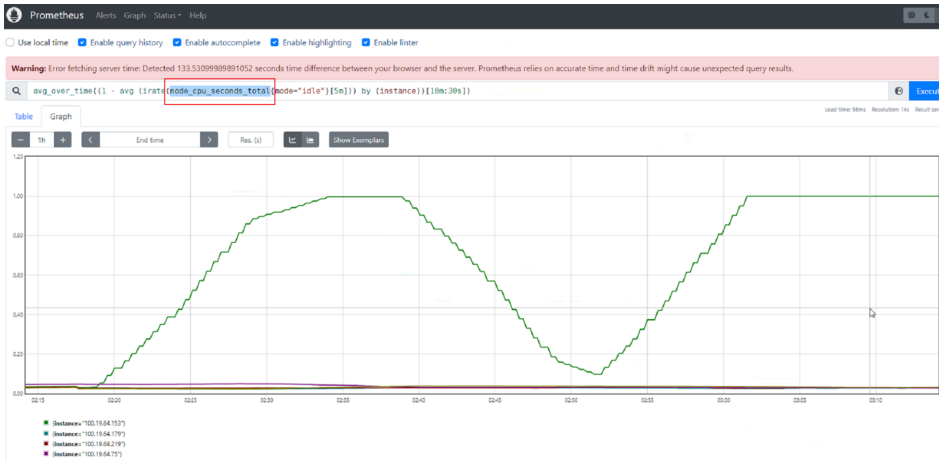
1. Run the following commands to install prometheus-adapter:
   ```
   git clone https://github.com/kubernetes-sigs/prometheus-adapter.git
   cd prometheus-adapter/deploy/manifests
   kubectl apply -f .
   ```
2. Modify the configuration for prometheus-adapter to connect to prometheus-server.
   ```
   kubectl edit deployment prometheus-adapter -n monitoring
   ```
   Change the value of **prometheus-url** as follows:
   – Change HTTPS to HTTP.
   – Change the default domain name to the IP address and port of Prometheus Service. You can run the **kubectl get service -n monitoring** command to query the IP address and port.

   ```
   ...
       containers:
         - name: prometheus-adapter
           image: registry.k8s.io/prometheus-adapter/prometheus-adapter:v0.12.0
           args:
             - --cert-dir=/var/run/serving-cert
             - --config=/etc/adapter/config.yaml
             - --prometheus-url=http://10.21.72.124:9090/
             - --secure-port=6443
   ```



```
        - --tls-cipher-
suites=TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY130
5,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_
ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_E
CDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_EC
DSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_A
ES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_
RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA
...
```

3. Verify that the native **node_cpu_seconds_total** and **node_memory_MemAvailable_bytes** metrics can be obtained on Prometheus.



Step 2 Enable Custom Metrics API to provide container resource metrics.

Check whether Custom Metrics API has been enabled for the cluster. If it is enabled, you can skip this step.

```
kubectl get APIServices | grep v1beta1.custom.metrics.k8s.io
```

If any command output is displayed, Custom Metrics API has been enabled. If no command output is displayed, perform the following operations to register the APIService object. (Note that the Service name and namespace must be consistent with those in the actual installation environment.) The following uses the default values of kube-prometheus.

1. Create a file named **custom-metrics-apiservice.yaml**. Example file content:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app.kubernetes.io/component: metrics
    app.kubernetes.io/name: prometheus-adapter
    app.kubernetes.io/part-of: prometheus-adapter
  name: v1beta1.custom.metrics.k8s.io
spec:
  group: custom.metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: prometheus-adapter
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

2. Create an APIService object.

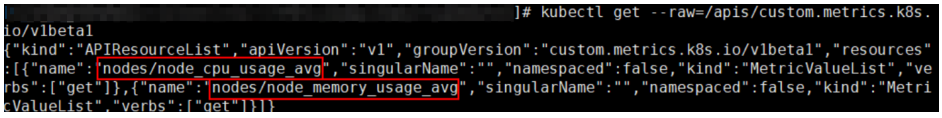
```
kubectl create -f custom-metrics-apiservice.yaml
```

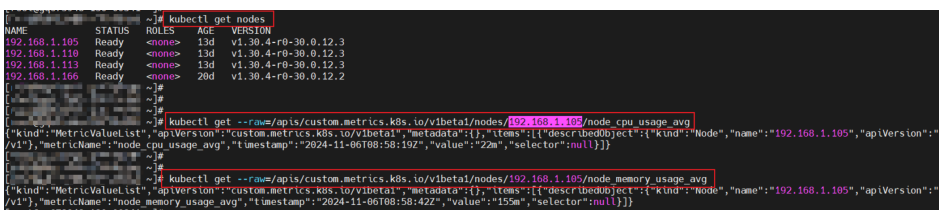

3. Check whether Custom Metrics API is enabled for the cluster.
   ```
   kubectl get APIServices | grep v1beta1.custom.metrics.k8s.io
   ```

Step 3 Add collection rules for custom metrics.

1. Modify the **adapter-config** configuration item.
   ```
   kubectl edit configmap adapter-config -n monitoring
   ```
2. Add collection rules to the **rules** field.

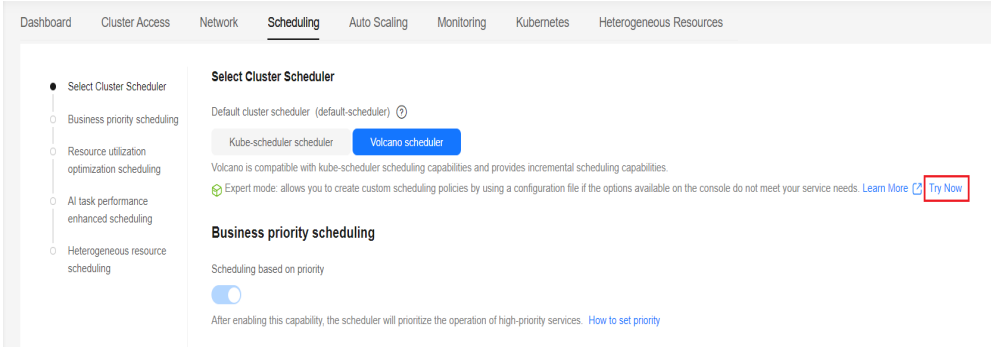
   Example collection rules:
   ```
   ...
   data:
     config.yaml: >
       rules:
       - seriesQuery: '{__name__=~"node_cpu_seconds_total"}'
         resources:
           overrides:
             instance:
               resource: node
         name:
           matches: node_cpu_seconds_total
           as: node_cpu_usage_avg
         metricsQuery: avg_over_time((1 - avg (irate(<<.Series>>{mode="idle"}[5m])) by (instance))
   [10m:30s])
       - seriesQuery: '{__name__=~"node_memory_MemTotal_bytes"}'
         resources:
           overrides:
             instance:
               resource: node
         name:
           matches: node_memory_MemTotal_bytes
           as: node_memory_usage_avg
         metricsQuery: avg_over_time(((1-node_memory_MemAvailable_bytes/<<.Series>>))[10m:30s])
   ...
   ```
3. Redeploy the prometheus-adapter workload in the **monitoring** namespace.
   ```
   kubectl rollout restart deployment prometheus-adapter -n monitoring
   ```
4. Verify that the custom rules are configured successfully.

   a. Run the following command. If the custom metric information is returned, the metric collection configuration on Prometheus is successful.
      ```
      kubectl get --raw=/apis/custom.metrics.k8s.io/v1beta1
      ```
      

   b. Query information about nodes in the cluster.
      ```
      kubectl get nodes
      ```
      Run the following command on any node. Replace *xxxx* with the obtained value of **node_name**. If you want to query the resource information of all nodes, replace *xxxx* with an asterisk (*).
      ```
      kubectl get --raw=/apis/custom.metrics.k8s.io/v1beta1/nodes/xxxx/node_cpu_usage_avg
      ```
      Information similar to the following is displayed.
      

Step 4 Enable load-aware scheduling.

1. Log in to the CCE console.
2. Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.
3. In **Expert mode**, click **Try Now**.



4. Modify the parameter settings in expert mode and configure the load-aware scheduling policy. The parameters in red are added to enable load-aware scheduling. The details are as follows:

```
admission_kube_api_qps: 200
admissions: /jobs/mutate,/jobs/validate,/podgroups/mutate,/pods/validate,/pods/mutate,/queues/
mutate,/queues/validate,/eas/pods/mutate,/eas/pods/validate,/npu/jobs/validate,/resource/validate,/
resource/mutate,/workloadbalancer/balancer/validate,/workloadbalancer/balancerpolicytemplate/
validate
annotations: {}
colocation_enable: "true"
controller_kube_api_qps: 200
default_scheduler_conf:
  actions: allocate, backfill, preempt
  metrics:
    interval: 30s
    type: prometheus_adaptor
  tiers:
    - plugins:
        - name: priority
        - enableJobStarving: false
          enablePreemptable: false
          name: gang
        - name: conformance
        - name: oversubscription
    - plugins:
        - enablePreemptable: false
          name: drf
        - name: predicates
        - name: nodeorder
        - arguments:
            cpu.weight: 1
            memory.weight: 1
            thresholds:
              cpu: 80
              mem: 80
          usage.weight: 5
          enablePredicate: true
          name: usage
    - plugins:
        - name: cce-gpu-topology-predicate
        - name: cce-gpu-topology-priority
        - name: xgpu
    - plugins:
        - name: nodelocalvolume
        - name: nodeemptydirvolume
        - name: nodeCSIscheduling
        - name: networkresource
deschedulerPolicy:
```



```
profiles:
  - name: ProfileName
    pluginConfig:
      - args:
          nodeFit: true
        name: DefaultEvictor
      - args:
          evictableNamespaces:
            exclude:
              - kube-system
          thresholds:
            cpu: 20
            memory: 20
        name: HighNodeUtilization
      - args:
          evictableNamespaces:
            exclude:
              - kube-system
          metrics:
            type: prometheus_adaptor
          nodeFit: true
          targetThresholds:
            cpu: 80
            memory: 85
          thresholds:
            cpu: 30
            memory: 30
        name: LoadAware
    plugins:
      balance:
        enabled: null
descheduler_enable: "false"
deschedulingInterval: 10m
enable_workload_balancer: false
oversubscription_method: nodeResource
oversubscription_profile_period: 300
oversubscription_ratio: 60
recommendation_enable: ""
scheduler_kube_api_qps: 200
update_pod_status_qps: 50
workload_balancer_score_annotation_key: ""
workload_balancer_third_party_types: ""
```

The key parameters for cluster load-aware scheduling are as follows:

- **usage.weight**: indicates the weight of the load-aware scheduling policy. A larger value indicates a higher weight of the load-aware policy in overall scheduling.
- **cpu.weight**: indicates the CPU weight. A larger value indicates CPU resources will be preferentially balanced.
- **memory.weight**: indicates the memory weight. A larger value indicates memory resources will be preferentially balanced.
- **thresholds**:
  - **cpu**: indicates the actual CPU load threshold.
  - **mem**: indicates the actual memory load threshold.
- **enablePredicate**: indicates the actual effective mode of the load threshold.
  - If the value is **true**, hard constraints are enabled. When the actual CPU or memory load of a node reaches the threshold, no new tasks can be scheduled to this node.

- If the value is **false**, soft constraints are enabled. When the actual CPU or memory load of a node reaches the threshold, new tasks will be preferentially allocated to underutilized nodes, but this node can still be scheduled.

----End

6.5.3.5 Configuration Cases for Resource Usage-based Scheduling

Overview

Volcano scheduling involves node filtering and scoring, which is used to filter the nodes meeting scheduling conditions and score the filtered nodes to find the one with the highest score for scheduling. Volcano provides multiple scheduling policies for node scoring. The weight of each scheduling policy can be adjusted based on service scenarios to enhance or reduce the impact of the policy on node scoring.

Scheduling Policies for Node Scoring

The following table lists the scheduling policies supported by Volcano for node scoring.

Scheduling Policy	Parameter	Description	Reference
Bin packing	binpack.weight	After this function is enabled, the parameter defaults to **10**.	**Bin Packing**
kube-scheduler node sorting (nodeorder)	nodeaffinity.weight	Pods are scheduled based on node affinity. The parameter defaults to **2**.	By default, this function is enabled.
	podaffinity.weight	Pods are scheduled based on pod affinity. The parameter defaults to **2**.	
	leastrequested.weight	Pods are scheduled to the node with the least requested resources. The parameter defaults to **1**.	
	balancedresource.weight	Pods are scheduled to the node with balanced resource allocation. The parameter defaults to **1**.	
	mostrequested.weight	Pods are scheduled to the node with the most requested resources. The parameter defaults to **0**.	

Scheduling Policy	Parameter	Description	Reference
	tainttoleration.weight	Pods are scheduled to the node with a high taint tolerance. The parameter defaults to **3**.	
	imagelocality.weight	Pods are scheduled to the node where the required images exist. The parameter defaults to **1**.	
	selectorspread.weight	Pods are evenly scheduled to different nodes. The parameter defaults to **0**.	
	podtopologyspread.weight	Pods are scheduled based on the pod topology. The parameter defaults to **2**.	
NUMA affinity scheduling (numa-aware)	weight	After this function is enabled, the parameter defaults to **1**.	**NUMA Affinity Scheduling**
Load-aware scheduling (usage)	weight	After this function is enabled, the parameter defaults to **5**.	**Load-aware Scheduling**
Node pool affinity scheduling (nodepoolaffinity)	nodepoolaffinity.weight	After this function is enabled, the parameter defaults to **10000**.	**Node Pool Affinity**

How Can I Improve Cluster Resource Utilization by Reducing Node Resource Fragments?

There are both heavy- and low-resource jobs running in a cluster. It is hoped that the low-resource job preferentially uses resource fragments on each node so that idle nodes can be allocated to the high-resource job. This prevents job scheduling failures caused by insufficient node resources.

To resolve the preceding issue, enable bin packing and use the default policy weight **10**. For details, see **Bin Packing**.

Recommended configurations:

- To preferentially reduce CPU fragments in the cluster, increase the CPU weight to **5** and retain the memory weight to **1** in the bin packing policy.

- To preferentially reduce memory fragments in the cluster, increase the memory weight to **5** and retain the CPU weight to **1** in the bin packing policy.
- To preferentially reduce GPU fragments in the cluster, customize the GPU resource type, set the GPU weight to **10**, and retain both the CPU weight and memory weight to **1** in the bin packing policy.

How Can I Balance the Actual CPU and Memory Loads on Nodes?

When a workload is running, the CPU and memory resources used may differ greatly from what was initially requested. To avoid any issues caused by overloading a single node, it is hoped that the scheduler preferentially schedules pods to the nodes with lighter loads based on nodes' CPU and memory usage in the cluster. This balances loads between nodes and ensures the stability of both applications and nodes.

Configuration case 1

1. Enable load-aware scheduling and use the default policy weight **5**. For details, see **Load-aware Scheduling**.
2. Disable bin packing. For details, see **Bin Packing**.

Recommended configurations:

- To preferentially balance the CPU load of each node, increase the CPU weight of the policy to **5** and retain the memory weight to **1**.
- To preferentially balance the memory load of each node, increase the memory weight of the policy to **5** and retain the CPU weight to **1**.
- To use both the CPU and memory usage and the CPU and memory thresholds, do as follows:
  - Hard constraints:
    - After the CPU usage of a node exceeds its CPU threshold, do not schedule new loads to the node.
    - After the memory usage of a node exceeds its memory threshold, do not schedule new loads to the node.
  - Soft constraints:
    - After the CPU usage of a node exceeds its CPU threshold, do not schedule new loads to the node as far as possible.
    - After the memory usage of a node exceeds its memory threshold, do not schedule new loads to the node as far as possible.
  - To balance the load of each node in a cluster while maximizing the cluster resource utilization, enable soft constraints for the CPU and memory thresholds and use the default value **80** for both the CPU and memory thresholds.
  - To ensure workload stability and reduce the CPU and memory usage of heavy-load nodes, enable hard constraints for the CPU and memory thresholds and set the CPU and memory thresholds a value ranging from 60 to 80.

Configuration case 2

The status, workload traffic, and requests of a cluster change dynamically, and the resource usage of nodes changes in real time. Node imbalancing may recur after pod scheduling. Use both load-aware scheduling and descheduling for the optimal load balancing of cluster nodes. For details about hotspot descheduling, see **Descheduling**.

1. Enable load-aware scheduling and use the default policy weight **5**. For details, see **Load-aware Scheduling**.
2. Enable descheduling and configure the load-aware descheduling policy. For details, see **Descheduling**.
3. Disable bin packing. For details, see **Bin Packing**.

Recommended configurations:

- Configure the load-aware descheduling policy as follows:
  - **targetThreshold** for evicting pods from heavy-load nodes: Set the CPU threshold to **75** and memory threshold to **70**.
  - **thresholds** for accepting pods on light-load nodes: Set both the CPU and memory thresholds to **30**.
- Ensure the actual CPU or memory threshold is between the CPU or memory threshold of the heaviest-load node and that of the lightest-load node.
  - Actual CPU threshold: **65**
  - Actual memory threshold: **60**

6.5.4 Priority-based Scheduling

6.5.4.1 Priority-based Scheduling and Preemption

A pod priority indicates the importance of a pod relative to other pods. Volcano supports pod **PriorityClasses** in Kubernetes. After PriorityClasses are configured, the scheduler preferentially schedules high-priority pods. When cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods.

Prerequisites

- A cluster of v1.19 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **Volcano Scheduler**.

Overview

The services running in a cluster are diversified, including core services, non-core services, online services, and offline services. You can configure priorities for different services based on service importance and SLA requirements. For example, configure a high priority for core services and online services so that such services preferentially obtain cluster resources. When cluster resources are used by non-core services and the remaining resources are insufficient for new core services, the scheduler evicts certain pods of non-core services to release the resources for scheduling the pods of the core services.

Table 6-20 lists the priority-based scheduling supported by CCE clusters.

Table 6-20 Priority-based scheduling

Scheduling Type	Description
Priority-based scheduling	The scheduler preferentially guarantees the running of high-priority pods, but will not evict low-priority pods that are running. Priority-based scheduling is enabled by default and cannot be disabled.
Priority-based preemption	When cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods.

Configuring Priority-based Scheduling and Preemption Policies

After Volcano is installed, you can enable or disable priority-based scheduling on the **Scheduling** page.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

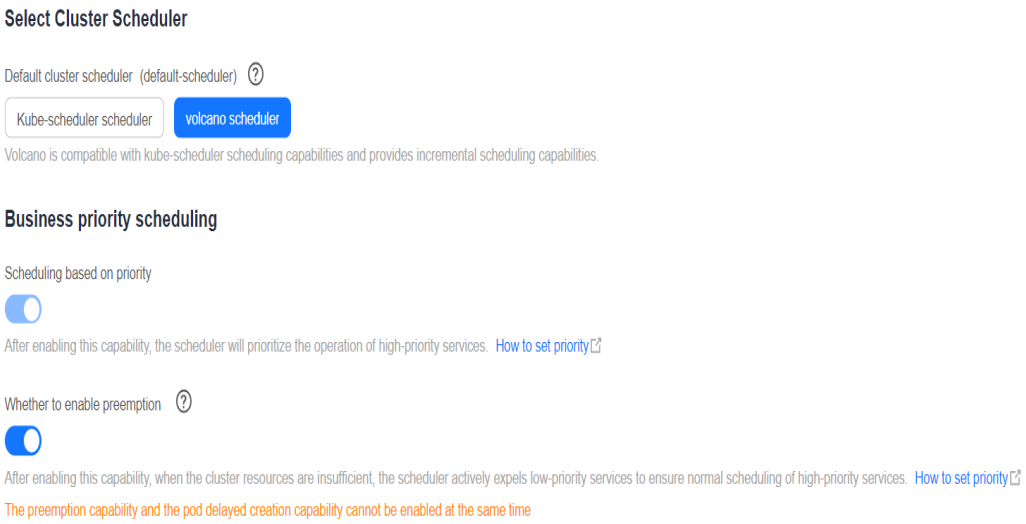
Step 3 In the **Business priority scheduling** area, configure priority-based scheduling.

- **Scheduling based on priority**: The scheduler preferentially guarantees the running of high-priority pods, but will not evict low-priority pods that are running. Priority-based scheduling is enabled by default and cannot be disabled.
- **Whether to enable preemption**: If Volcano Scheduler is used as the default scheduler of the cluster, priority-based preemption is supported. When cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods.

  NOTE

  - After priority-based preemption is enabled, delayed pod creation is not allowed.
  - Priority-based preemption is not allowed on custom ENI/sub-ENI resources or host ports.

Figure 6-20 Priority-based scheduling



Step 4 Click **Confirm**.

Step 5 After the configuration, you can use **PriorityClasses** to schedule the pods of workloads or Volcano jobs based priorities.

1. Create one or more **PriorityClasses**.

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: ""
```

2. Create a workload or Volcano job and specify its PriorityClass name.
   - Workload

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: high-test
  labels:
    app: high-test
spec:
  replicas: 5
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      priorityClassName: high-priority
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 500m
```


- Volcano job

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

----End

Example of Priority-based Scheduling

For example, there are two idle nodes and several workloads with three priorities (high-priority, medium-priority, and low-priority). Run the high-priority workload to exhaust all cluster resources, and issue the medium-priority and low-priority workloads. Then, the two types of workloads are pending due to insufficient resources. When the high-priority workload ends, the pods of the medium-priority workload will be scheduled ahead of the pods of the low-priority workload according to the priority-based scheduling setting.

Step 1 Add three **PriorityClasses** (**high-priority**, **med-priority**, and **low-priority**) in **priority.yaml**.

Example configuration of **priority.yaml**:

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 100
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: med-priority
value: 50
globalDefault: false
description: "This priority class should be used for volcano job only."
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: low-priority
value: 10
globalDefault: false
description: "This priority class should be used for volcano job only."
```

Create PriorityClasses.


```
kubectl apply -f priority.yaml
```

Step 2 Check PriorityClasses.

```
kubectl get PriorityClass
```

Command output:

```
NAME                      VALUE        GLOBAL-DEFAULT   AGE
high-priority             100          false            97s
low-priority              10           false            97s
med-priority              50           false            97s
system-cluster-critical   2000000000   false            6d6h
system-node-critical      2000001000   false            6d6h
```

Step 3 Create a high-priority workload named **high-priority-job** to exhaust all cluster resources.

high-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-high
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: high-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

Run the following command to issue the job:

```
kubectl apply -f high_priority_job.yaml
```

Run the **kubectl get pod** command to check pod statuses:

```
NAME                  READY   STATUS    RESTARTS   AGE
priority-high-test-0  1/1     Running   0          3s
priority-high-test-1  1/1     Running   0          3s
priority-high-test-2  1/1     Running   0          3s
priority-high-test-3  1/1     Running   0          3s
```

The command output shows that all cluster resources have been used up.

Step 4 Create a medium-priority workload **med-priority-job** and a low-priority workload **low-priority-job**.

med-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-medium
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: med-priority
```



```
tasks:
  - replicas: 4
    name: "test"
    template:
      spec:
        containers:
          - image: alpine
            command: ["/bin/sh", "-c", "sleep 1000"]
            imagePullPolicy: IfNotPresent
            name: running
            resources:
              requests:
                cpu: "1"
        restartPolicy: OnFailure
```

low-priority-job.yaml

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: priority-low
spec:
  schedulerName: volcano
  minAvailable: 4
  priorityClassName: low-priority
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

Run the following commands to issue the jobs:

```
kubectl apply -f med_priority_job.yaml
kubectl apply -f low_priority_job.yaml
```

Run the **kubectl get pod** command to check the statuses of the pods for the newly created workloads. The command output shows that the pods are pending due to insufficient resources:

```
NAME                    READY   STATUS    RESTARTS   AGE
priority-high-test-0    1/1     Running   0          3m29s
priority-high-test-1    1/1     Running   0          3m29s
priority-high-test-2    1/1     Running   0          3m29s
priority-high-test-3    1/1     Running   0          3m29s
priority-low-test-0     0/1     Pending   0          2m26s
priority-low-test-1     0/1     Pending   0          2m26s
priority-low-test-2     0/1     Pending   0          2m26s
priority-low-test-3     0/1     Pending   0          2m26s
priority-medium-test-0  0/1     Pending   0          2m36s
priority-medium-test-1  0/1     Pending   0          2m36s
priority-medium-test-2  0/1     Pending   0          2m36s
priority-medium-test-3  0/1     Pending   0          2m36s
```

Step 5 Delete the **high_priority_job** workload to release resources and check whether the pods of the **med-priority-job** workload will be preferentially scheduled.

Run the **kubectl delete -f high_priority_job.yaml** command to release cluster resources and check pod scheduling.


```
NAME                    READY   STATUS    RESTARTS   AGE
priority-low-test-0     0/1     Pending   0          5m18s
priority-low-test-1     0/1     Pending   0          5m18s
priority-low-test-2     0/1     Pending   0          5m18s
priority-low-test-3     0/1     Pending   0          5m18s
priority-medium-test-0  1/1     Running   0          5m28s
priority-medium-test-1  1/1     Running   0          5m28s
priority-medium-test-2  1/1     Running   0          5m28s
priority-medium-test-3  1/1     Running   0          5m28s
```

----End

Example of Priority-based Preemption

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

Step 2 Modify configurations.

1. Select **Volcano scheduler** as the default cluster scheduler.
2. Enable **Scheduling based on priority**.

Step 3 Issue the **high_priority_job** workload in the priority-based scheduling scenario. Then, the scheduler will evict the pods of the **med_priority_job** workload so that the pods of the high-priority workload can be scheduled.

Run the **kubectl apply -f high_priority_job.yaml** command to issue the high-priority workload. Then, check pod statuses.

```
NAME                    READY   STATUS        RESTARTS   AGE
priority-high-test-0    0/1     Pending       0          2s
priority-high-test-1    0/1     Pending       0          2s
priority-high-test-2    0/1     Pending       0          2s
priority-high-test-3    0/1     Pending       0          2s
priority-low-test-0     0/1     Pending       0          14s
priority-low-test-1     0/1     Pending       0          14s
priority-low-test-2     0/1     Pending       0          14s
priority-low-test-3     0/1     Pending       0          14s
priority-medium-test-0  1/1     Terminating   0          21s
priority-medium-test-1  1/1     Terminating   0          21s
priority-medium-test-2  1/1     Terminating   0          21s
priority-medium-test-3  1/1     Terminating   0          21s
```

After the resources used by the **med_priority_job resource** workload are released, the pods of the **high_priority_job** workload can be scheduled.

```
NAME                    READY   STATUS    RESTARTS   AGE
priority-high-test-0    1/1     Running   0          70s
priority-high-test-1    1/1     Running   0          70s
priority-high-test-2    1/1     Running   0          70s
priority-high-test-3    1/1     Running   0          70s
priority-low-test-0     0/1     Pending   0          82s
priority-low-test-1     0/1     Pending   0          82s
priority-low-test-2     0/1     Pending   0          82s
priority-low-test-3     0/1     Pending   0          82s
priority-medium-test-0  0/1     Pending   0          37s
priority-medium-test-1  0/1     Pending   0          36s
priority-medium-test-2  0/1     Pending   0          37s
priority-medium-test-3  0/1     Pending   0          37s
```

When node resources cannot meet the **high_priority_job** requirements, priority-based preemption of volcano-scheduler will be enabled. The pods of **med_priority_job** will be evicted for the deployment of **high_priority_job**. After

new nodes are added using Cluster Autoscaler, volcano-scheduler will schedule **med_priority_job** pods to the new nodes.

According to the preceding test results, enable node scaling if priority-based preemption is enabled so that cluster resources can be allocated on demand to ensure service SLA.

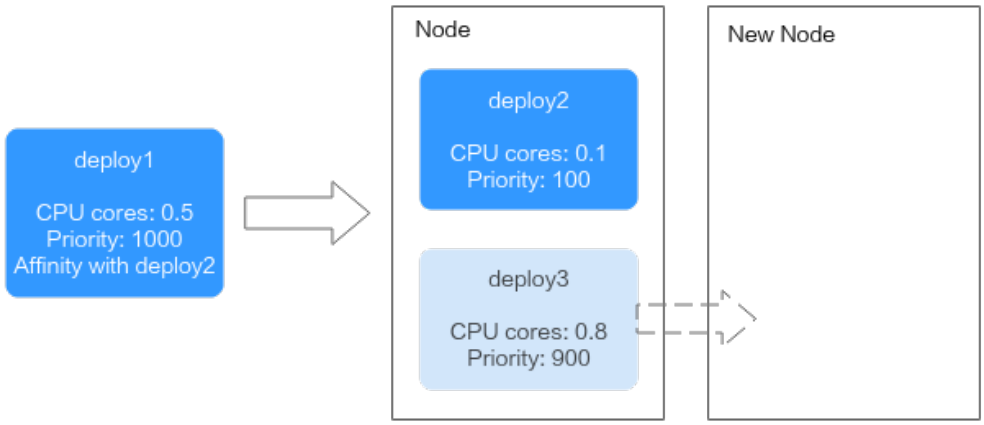
----End

Example of Affinity and Anti-affinity for Priority-based Preemption

Do not configure inter-pod affinity on the pods with lower priorities. If a pod in the pending state is inter-pod affinity with one or more pods with lower priorities on the node, the pod affinity rule cannot be met when preemption is initiated for the pods with lower priorities, and the preemption rule conflicts with the affinity rule. In this case, the scheduler cannot ensure the scheduling of the pending pod. To resolve this issue, configure inter-pod affinity only for pods with the same or higher priority. For details, see **Inter-pod affinity on lower-priority pods**.

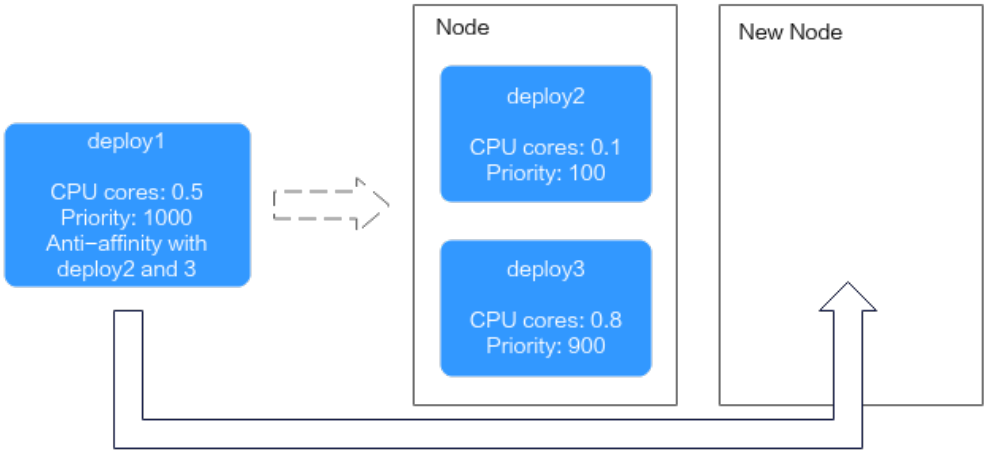
In inter-pod affinity, if priority-based preemption is enabled and deploy1 is affinity with lower-priority deploy2, volcano-scheduler will evict deploy3 and schedule deploy1 to the node to ensure service O&M. The evicted deploy3 will be scheduled to the new node after the new node is ready.

Figure 6-21 Inter-pod affinity on lower-priority pods



In inter-pod anti-affinity, if priority-based preemption is enabled and deploy1 is anti-affinity with deploy2 and deploy3, volcano-scheduler will not evict deploy2 and deploy3 to minimize the impact on other services. Instead, the scheduler will schedule deploy1 to a new node.

Figure 6-22 Inter-pod anti-affinity on lower-priority pods



6.5.5 AI Performance-based Scheduling

6.5.5.1 DRF

Dominant Resource Fairness (DRF) is a scheduling algorithm based on the dominant resource of a container group. DRF scheduling can be used to enhance the service throughput of a cluster, shorten the overall service execution time, and improve service running performance. It is suitable for batch AI training and big data jobs.

Prerequisites

- A cluster of v1.19 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **Volcano Scheduler**.

How It Works

In actual services, limited cluster resources are often allocated to multiple users. Each user has the same rights to obtain resources, but the number of resources they need may be different. It is crucial to fairly allocate resources to each user. A common scheduling algorithm is the max-min fairness share, which allocates resources to meet users' minimum requirements as far as possible and then fairly allocates the remaining resources. The rules are as follows:

1. Resources are allocated in order of increasing demand.
2. No source gets a resource share larger than its demand.
3. Sources with unsatisfied demands get an equal share of the resource.

The max-min fairness algorithm applies to the single resource scenario, where all jobs are requesting the same resources. However, in actual situations, multiple resources are involved. For example, CPU, memory, and GPU resources are requested for allocation. DRF can be used to resolve the preceding issue. DRF can be considered as a general version of the max-min fairness algorithm and

supports fair allocation of multiple types of resources so that the dominant resource of each user meets the max-min fairness requirement.

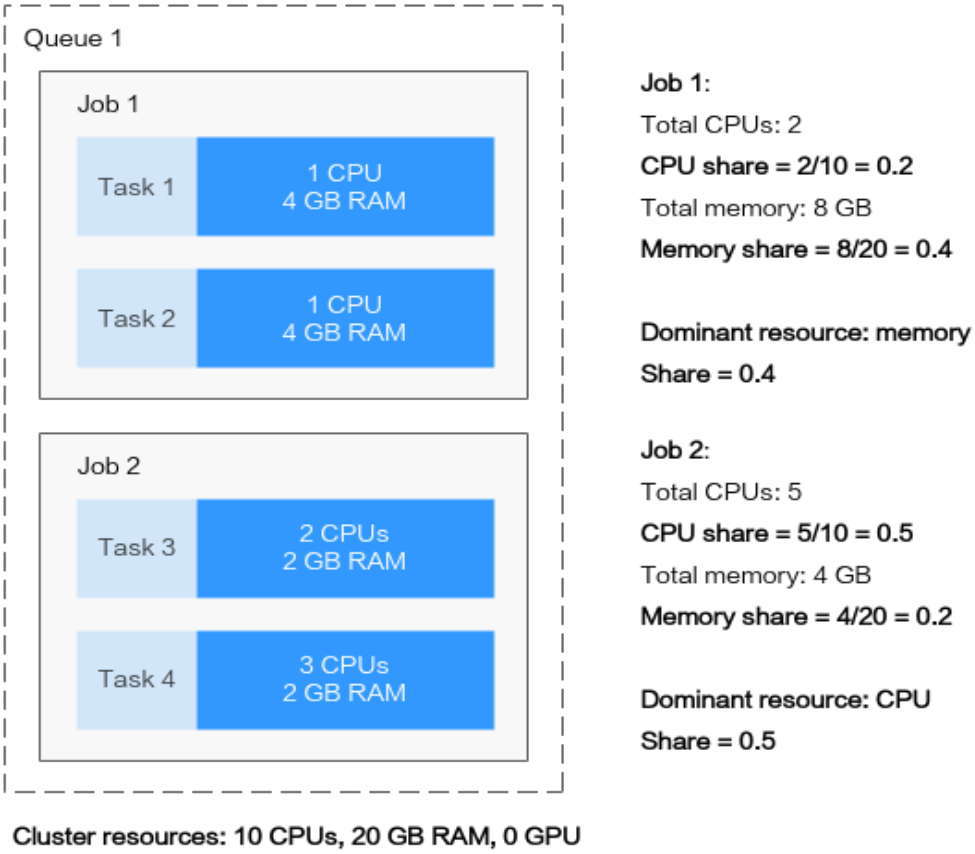
The share value of each job resource is calculated using the following formula:

Share = Total requested resources/Cluster resources

If a job involves multiple resources, the resource with the largest share value is the dominant resource. The share value of the dominant resource will be used in priority-based scheduling.

For example, there are two workloads, job 1 and job 2. The following figure shows the resources requested by the two jobs. After DRF calculation, the dominant resource of job 1 is memory, and its share value is 0.4; the dominant resource of job 2 is CPU, and its share value is 0.5. Since the dominant resource share of job 1 is less than that of job 2, job 1 takes precedence over job 2 in scheduling according to the max-min fairness policy.

Figure 6-23 DRF scheduling



Configuring DRF

After Volcano is installed, you can enable or disable DRF scheduling on the **Scheduling** page. This function is enabled by default.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

Step 3 In the **AI task performance enhanced scheduling** pane, select whether to enable DRF.

This function helps you enhance the service throughput of the cluster and improve service running performance.

Step 4 Click **Confirm**.

----End

6.5.5.2 Gang

Gang scheduling is a scheduling algorithm that schedules correlated processes or threads to run simultaneously on different processors. It meets the scheduling requirements of "All or nothing" in the scheduling process and avoids the waste of cluster resources caused by arbitrary scheduling of pods. Gang is mainly used in scenarios that require multi-process collaboration, such as AI and big data scenarios. Gang scheduling effectively resolves pain points such as resource waiting or deadlocks in distributed training jobs, thereby significantly improving the utilization of cluster resources.

Prerequisites

- A cluster of v1.19 or later is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **Volcano Scheduler**.

How It Works

The Gang scheduling policy is one of the core scheduling algorithms of Volcano. It meets the scheduling requirements of "All or nothing" in the scheduling process and avoids the waste of cluster resources caused by arbitrary scheduling of pods. The Gang scheduler algorithm checks whether the number of scheduled pods in a job meets the minimum requirements for running the job. If yes, all pods in the job will be scheduled. If no, the pods will not be scheduled.

The Gang scheduling algorithm based on container groups is well suitable for scenarios where multi-process collaboration is required. AI scenarios typically involve complex processes. Data ingestion, data analysts, data splitting, trainers, serving, and logging which require a group of containers to work together are suitable for container-based Gang scheduling. Multi-thread parallel computing communication scenarios under MPI computing framework are also suitable for Gang scheduling because master and slave processes need to work together. Containers in a pod group are highly correlated, and there may be resource contention. The overall scheduling allocation can effectively resolve deadlocks. If cluster resources are insufficient, Gang scheduling can significantly improve the utilization of cluster resources.

Configuring Gang

After Volcano is installed, you can enable or disable Gang scheduling on the **Scheduling** page. This function is enabled by default.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

Step 3 In the **AI task performance enhanced scheduling** pane, select whether to enable Gang.

This function helps you enhance the service throughput of the cluster and improve service running performance.

Step 4 Click **Confirm**.

Step 5 After the configuration, use Gang scheduling in workloads or Volcano jobs.

- Create a workload using Gang scheduling.
  a. Create a pod group and specify **minMember** and **minResources** as follows:

```
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
metadata:
  name: pg-test1
spec:
  minMember: 3
  minResources:
    cpu: 3
    memory: 3Gi
```

  - **minMember**: specifies the minimum requirement on the number of pods for running a workload. When the number of pods in the current pod group meets the requirement, these pods can be centrally scheduled.
  - **minResources**: specifies the minimum requirement on resources for running a workload. When the available resources in a cluster meet the requirement, the group of pods can be centrally scheduled.

  b. When creating a workload, use **schedulerName** to specify Volcano Scheduler and **annotation** to specify the pod group in which Volcano Scheduler runs.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: podgroup-test
  labels:
    app: podgroup-test
spec:
  replicas: 6
  selector:
    matchLabels:
      app: podgroup-test
  template:
    metadata:
      annotations:
        scheduling.k8s.io/group-name: pg-test1
      labels:
        app: podgroup-test
    spec:
      schedulerName: volcano
      containers:
      - name: test
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
        resources:
          requests:
            cpu: 500m
```



```
      limits:
        cpu: 500m
```

  - **schedulerName**: Set this parameter to **volcano**, indicating that Volcano will be used to schedule pods for the workload.
  - **scheduling.k8s.io/group-name**: specifies the pod group created in the previous step, for example, **pg-test1**.

- Create a Volcano job using Gang scheduling.

  When creating a Volcano job, you only need to configure **minAvailable** and set **schedulerName** to **volcano**. Volcano Scheduler will automatically create a pod group and manage it. The following shows an example:

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vcjob
spec:
  schedulerName: volcano
  minAvailable: 2
  tasks:
    - replicas: 4
      name: "test"
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                requests:
                  cpu: "1"
          restartPolicy: OnFailure
```

----End

6.5.6 NUMA Affinity Scheduling

In non-uniform memory access (NUMA) architecture, a NUMA node is a fundamental component that includes a processor and local memory. These nodes are physically separate but interconnected through a high-speed bus to form a complete system. To boost system performance, NUMA nodes allow for quicker access to local memory. However, accessing memory across multiple NUMA nodes within a node can cause delays. To enhance memory access efficiency and overall performance, it is crucial to optimize task scheduling and memory allocation.

When working with high-performance computing (HPC), real-time applications, or memory-intensive workloads that require frequent communication between CPUs, accessing nodes across NUMA in a cloud native environment can lead to decreased system performance due to increased latency and overhead. Volcano's NUMA affinity scheduling resolves the issue by scheduling pods to the worker node that requires the least number of cross-NUMA nodes. This reduces data transmission overheads, optimizes resource utilization, and enhances overall system performance.

For more information, see **https://github.com/volcano-sh/volcano/blob/master/docs/design/numa-aware.md**.

Prerequisites

- A CCE standard or Turbo cluster is available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed in the cluster. For details, see **Volcano Scheduler**.

Pod Scheduling Process

After a topology policy is configured for pods, Volcano predicts the nodes that match the policy. For details about how to configure a pod topology policy, see **Example of NUMA Affinity Scheduling**. The scheduling process is as follows:

1. Volcano filters nodes with the same policy based on the topology policy configured for pods. The topology policy provided by Volcano is the same as that provided by the **topology manager**.
2. Among the nodes where the same policy applies, Volcano selects the nodes whose CPU topology meets the policy requirements for scheduling.

Pod Topology Policy	How to Filter Nodes During Pod Scheduling	
	1. Filter nodes that meet the topology policy set for the pod.	**2. Further filter the node whose CPU topology meets the policy.**
none	Nodes with the following topology policies will not be filtered during scheduling: • **none**: unschedulable • **best-effort**: schedulable • **restricted**: schedulable • **single-numa-node**: schedulable	None
best-effort	Nodes with the **best-effort** topology policy will be filtered. • **none**: unschedulable • **best-effort**: schedulable • **restricted**: unschedulable • **single-numa-node**: unschedulable	Best-effort scheduling: Pods are preferentially scheduled to a single NUMA node. If a single NUMA node cannot meet the requested CPU cores, the pods can be scheduled to multiple NUMA nodes.

Pod Topology Policy	How to Filter Nodes During Pod Scheduling	
	1. Filter nodes that meet the topology policy set for the pod.	**2. Further filter the node whose CPU topology meets the policy.**
restricted	Nodes with the **restricted** topology policy will be filtered. • **none**: unschedulable • **best-effort**: unschedulable • **restricted**: schedulable • **single-numa-node**: unschedulable	Restricted scheduling: • If the upper CPU limit of a single NUMA node is greater than or equal to the requested CPU cores, pods can only be scheduled to a single NUMA node. If the remaining CPU cores of a single NUMA node are insufficient, the pods cannot be scheduled. • If the upper CPU limit of a single NUMA node is less than the requested CPU cores, pods can be scheduled to multiple NUMA nodes.
single-numa-node	Nodes with the **single-numa-node** topology policy will be filtered. • **none**: unschedulable • **best-effort**: unschedulable • **restricted**: unschedulable • **single-numa-node**: schedulable	Pods can only be scheduled to a single NUMA node.

For example, two NUMA nodes provide resources, each with a total of 32 CPU cores. The following table lists resource allocation.

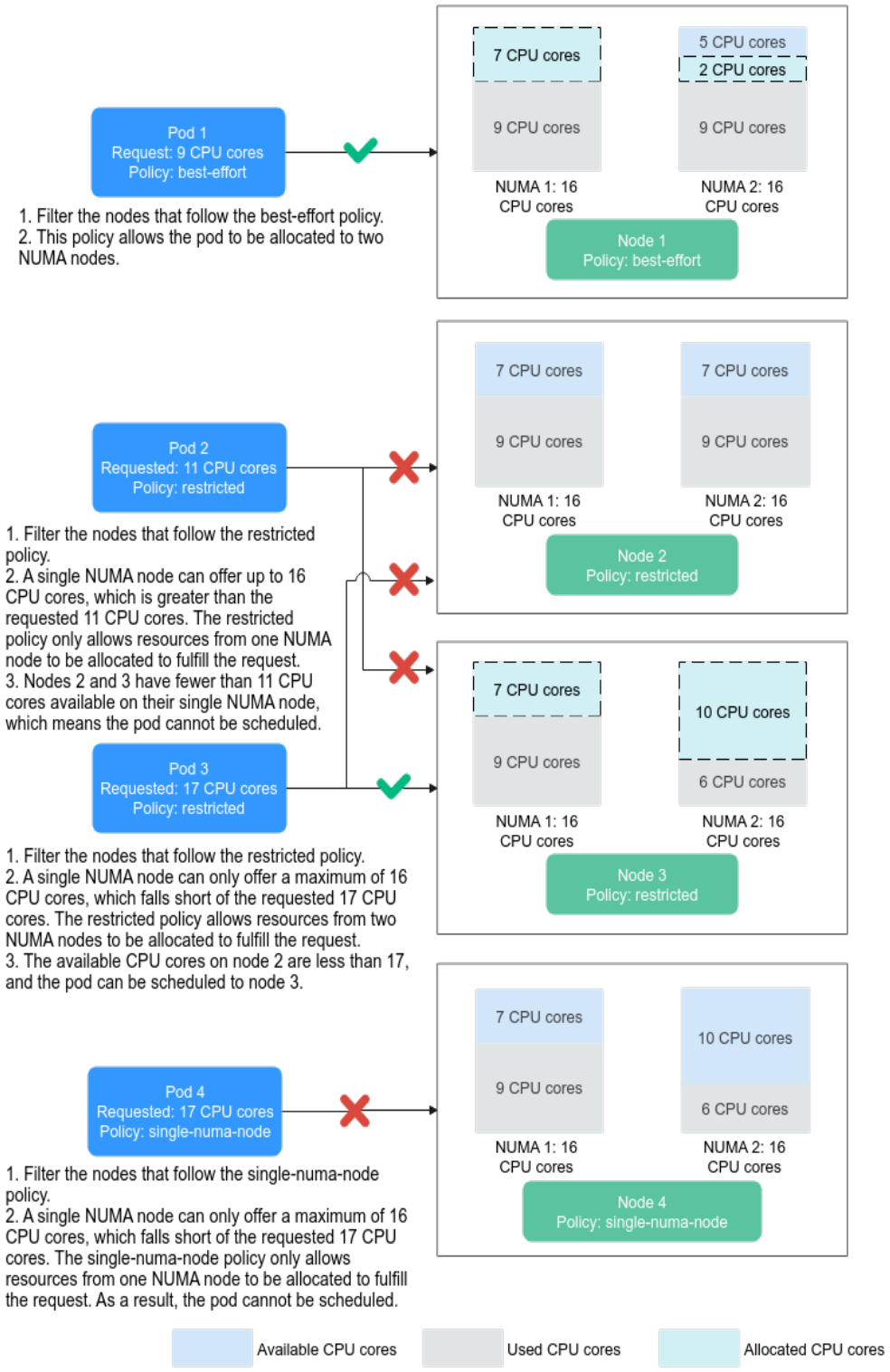
Worker Node	Node Topology Policy	Total CPU Cores on NUMA Node 1		Total CPU Cores on NUMA Node 2	
		Total CPU Cores	Available CPU Cores	Total CPU Cores	Available CPU Cores
Node 1	best-effort	16	7	16	7
Node 2	restricted	16	7	16	7
Node 3	restricted	16	7	16	10

Worker Node	Node Topology Policy	Total CPU Cores on NUMA Node 1		Total CPU Cores on NUMA Node 2	
		Total CPU Cores	Available CPU Cores	Total CPU Cores	Available CPU Cores
Node 4	single-numa-node	16	7	16	10

Figure 6-24 shows the scheduling of a pod after a topology policy is configured.

- When 9 CPU cores are requested by a pod and the **best-effort** topology policy is used, Volcano selects node 1 whose topology policy is also **best-effort**, and this policy allows the pod to be scheduled to multiple NUMA nodes. Therefore, the requested 9 CPU cores will be allocated to two NUMA nodes, and the pod can be scheduled to node 1.
- When 11 CPU cores are requested by a pod and the **restricted** topology policy is used, Volcano selects nodes 2 and 3 whose topology policy is also **restricted**, and each node provides at least 11 CPU cores. However, the remaining CPU cores on node 2 or 3 are less than the requested. Therefore, the pod cannot be scheduled.
- When 17 CPU cores are requested by a pod and the **restricted** topology policy is used, Volcano selects nodes 2 and 3 whose topology policy is also **restricted**, this policy allows the pod to be scheduled to multiple NUMA nodes, and the upper CPU limit of both the nodes is less than 17. Then, the pod can be scheduled to node 3.
- When 17 CPU cores are requested by a pod and the **single-numa-node** topology policy is used, Volcano selects nodes whose topology policy is also **single-numa-node**. However, no node can provide a total of 17 CPU cores. Therefore, the pod cannot be scheduled.

Figure 6-24 Comparison of NUMA scheduling policies



Scheduling Priority

A topology policy aims to schedule pods to the optimal node. In this example, each node is scored to sort out the optimal node.

Principle: Schedule pods to the worker nodes that require the fewest NUMA nodes.

The scoring formula is as follows:

score = weight x (100 - 100 x numaNodeNum/maxNumaNodeNum)

Parameters:

- **weight**: the weight of NUMA Aware Plugin.
- **numaNodeNum**: the number of NUMA nodes required for running the pod on worker nodes.
- **maxNumaNodeNum**: the maximum number of NUMA nodes required for running the pod among all worker nodes.

For example, three nodes meet the CPU topology policy for a pod and the weight of NUMA Aware Plugin is set to **10**.

- Node A: One NUMA node provides the CPU resources required by the pod (numaNodeNum = 1).
- Node B: Two NUMA nodes provide the CPU resources required by the pod (numaNodeNum = 2).
- Node C: Four NUMA nodes provide the CPU resources required by the pod (numaNodeNum = 4).

According to the preceding formula, **maxNumaNodeNum** is **4**.

- score (Node A) = 10 x (100 - 100 x 1/4) = 750
- score (Node B) = 10 x (100 - 100 x 2/4) = 500
- score (Node C) = 10 x (100 - 100 x 4/4) = 0

Therefore, the optimal node is Node A.

Enabling NUMA Affinity Scheduling for Volcano

Step 1 Enable static CPU management in the node pool. For details, see **Enabling CPU Management for a Custom Node Pool**.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. Choose **Nodes** in the navigation pane and click the **Node Pools** tab on the right. Locate the target node pool and choose **More** > **Manage**.
3. On the **Manage Configurations** page, change the **cpu-manager-policy** value to **static** in the **kubelet** area.

4. Click **OK**.

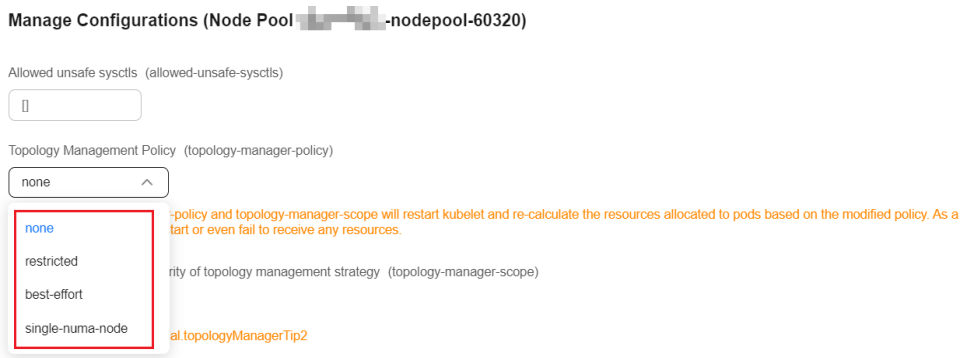
Step 2 Configure a CPU topology policy in the node pool.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Nodes**. On the right of the page, click

the **Node Pools** tab and choose **More** > **Manage** in the **Operation** column of the target node pool.

2. Change the kubelet **Topology Management Policy (topology-manager-policy)** value to the required CPU topology policy.

   Valid topology policies include **none**, **best-effort**, **restricted**, and **single-numa-node**. For details, see **Pod Scheduling Process**.



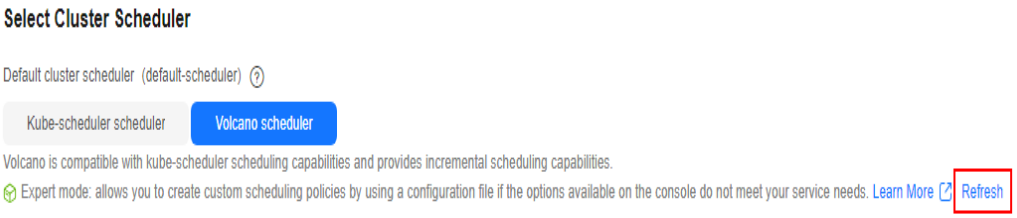
Step 3 Enable the numa-aware add-on and the **resource_exporter** function.

Volcano 1.7.1 or later

1. Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, locate **Volcano Scheduler** on the right, and click **Edit**.
2. In the **Extended Functions** area, enable **NUMA Topology Scheduling** and click **OK**.

Volcano earlier than 1.7.1

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab. Select **Volcano scheduler**, find the expert mode, and click **Refresh**.



2. Enable **resource_exporter_enable** to collect node NUMA information. The following is an example in JSON format:

```
{
  "plugins": {
    "eas_service": {
      "availability_zone_id": "",
      "driver_id": "",
      "enable": "false",
      "endpoint": "",
      "flavor_id": "",
      "network_type": "",
      "network_virtual_subnet_id": "",
      "pool_id": "",
      "project_id": "",
      "secret_name": "eas-service-secret"
    }
```



```
  },
  "resource_exporter_enable": "true"
}
```

After this function is enabled, you can view the NUMA topology information of the current node.

```
kubectl get numatopo
NAME            AGE
node-1          4h8m
node-2          4h8m
node-3          4h8m
```

3. Enable the Volcano numa-aware algorithm add-on.

   kubectl edit cm -n kube-system volcano-scheduler-configmap

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: volcano-scheduler-configmap
  namespace: kube-system
data:
  default-scheduler.conf: |-
    actions: "allocate, backfill, preempt"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: overcommit
      - name: drf
      - name: predicates
      - name: nodeorder
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
        arguments:
          NetworkType: vpc-router
      - name: numa-aware # add it to enable numa-aware plugin
        arguments:
          weight: 10 # the weight of the NUMA Aware Plugin
```

----End

Example of NUMA Affinity Scheduling

The following describes how to choose NUMA nodes for scheduling pods according to pod scheduling policies. For details, see **Pod Scheduling Process**.

- **single-numa-node**: When pods are scheduled, nodes in the node pool with the **single-numa-node** topology management policy will be chosen, and a single NUMA node must provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod cannot be scheduled.
- **restricted**: When pods are scheduled, nodes in the node pool with the **restricted** topology management policy will be chosen, and a set of NUMA nodes on the same node must provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod cannot be scheduled.
- **best-effort**: When pods are scheduled, nodes in the node pool with the **best-effort** topology management policy will be chosen, and a single NUMA node

needs to provide the CPU cores. If none of the nodes in the pool meet these requirements, the pod will be scheduled to the most suitable node.

Step 1 Refer to the following examples for configuration.

1. Example 1: Configure NUMA affinity for a Deployment.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: numa-tset
spec:
  replicas: 1
  selector:
    matchLabels:
      app: numa-tset
  template:
    metadata:
      labels:
        app: numa-tset
      annotations:
        volcano.sh/numa-topology-policy: single-numa-node    # Configure the topology policy.
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          resources:
            requests:
              cpu: 2          # The value must be an integer and must be the same as that in limits.
              memory: 2048Mi
            limits:
              cpu: 2          # The value must be an integer and must be the same as that in requests.
              memory: 2048Mi
      imagePullSecrets:
      - name: default-secret
```

2. Example 2: Create a Volcano job and enable NUMA affinity for it.

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: vj-test
spec:
  schedulerName: volcano
  minAvailable: 1
  tasks:
    - replicas: 1
      name: "test"
      topologyPolicy: best-effort   # set the topology policy for task
      template:
        spec:
          containers:
            - image: alpine
              command: ["/bin/sh", "-c", "sleep 1000"]
              imagePullPolicy: IfNotPresent
              name: running
              resources:
                limits:
                  cpu: 20
                  memory: "100Mi"
          restartPolicy: OnFailure
```

Step 2 Analyze NUMA scheduling.

The following table shows example NUMA nodes.

Worker Node	Topology Manager Policy	Allocatable CPU Cores on NUMA Node 0	Allocatable CPU Cores on NUMA Node 1
Node 1	single-numa-node	16	16
Node 2	best-effort	16	16
Node 3	best-effort	20	20

In the preceding examples,

- In example 1, 2 CPU cores are requested by a pod, and the **single-numa-node** topology policy is used. Therefore, the pod will be scheduled to node 1 with the same policy.
- In example 2, 20 CPU cores are requested by a pod, and the **best-effort** topology policy is used. The pod will be scheduled to node 3 because it can allocate all the requested 20 CPU cores onto one NUMA node, while node 2 can do so on two NUMA nodes.

----End

Checking NUMA Node Usage

Run the **lscpu** command to check the CPU usage of the current node.

```
# Check the CPU usage of the current node.
lscpu
...
CPU(s):              32
NUMA node(s):        2
NUMA node0 CPU(s):   0-15
NUMA node1 CPU(s):   16-31
```

Then, check the NUMA node usage.

```
# Check the CPU allocation of the current node.
cat /var/lib/kubelet/cpu_manager_state
{"policyName":"static","defaultCpuSet":"0,10-15,25-31","entries":{"777870b5-c64f-42f5-9296-688b9dc212ba":{"container-1":"16-24"},"fb15e10a-b6a5-4aaa-8fcd-76c1aa64e6fd":{"container-1":"1-9"}},"checksum":318470969}
```

The preceding example shows that two containers are running on the node. One container uses CPU cores 1 to 9 of NUMA node 0, and the other container uses CPU cores 16 to 24 of NUMA node 1.

Common Issues

Schedule pods failed.

If Volcano is set as the scheduler and only NUMA is enabled without configuring CPU management during pod scheduling, job scheduling may fail. To fix this issue, do as follows:

- Before using NUMA affinity scheduling, make sure that Volcano has been deployed and is running properly.

- When using NUMA affinity scheduling:
  a. Set the CPU management policy (**cpu-manager-policy**) of the node pool to **static**.
  b. Correctly configure the topology management policy (**topology-manager-policy**) in the node pool.
  c. Configure a correct topology policy for pods to filter nodes with the same topology policy in the node pool. For details, see **Example of NUMA Affinity Scheduling**.
  d. Configure the Volcano scheduler to schedule application pods. For details, see **Scheduling Workloads**. Make sure that the CPU requests for all containers within the pods are integers (measured in cores) and that the requests and limits are identical.

6.5.7 Application Scaling Priority Policies

With application scaling priority policies, you can manage resources more efficiently by customizing the scaling order of pods across different node types. If the default scaling priority policy is applied, pods will be scheduled first to yearly/monthly nodes during scale-out, followed by pay-per-use nodes and virtual-kubelet nodes (scaling pods to CCI). During scale-in, pods are deleted sequentially from virtual-kubelet nodes (scaling pods to CCI), pay-per-use nodes, and yearly/monthly nodes.

The application scaling priority policy includes the following two aspects:

- Scale-out: Volcano schedules new pods in a cluster based on preset node priority for scale-out.
- Scale-in: When a workload is specified, Volcano scores the workload based on preset node priority to determine pod deletion sequence during scale-in.

Notes and Constraints

- The cluster version must be 1.23.11 or later, 1.25.6 or later, or 1.27.3 or later.
- The Volcano Scheduler add-on (1.12.1 or later) must be installed in a cluster, and the application scaling priority policy function must be enabled.
- By default, the scaling priority takes effect for Deployments (including **ReplicaSet**). To make the scaling priority take effect on third-party workloads, you can adjust the advanced settings. For details, see **Configuring a Scaling Priority Policy for a Third-Party Workload**.
- To use the scale-out scheduling priority policies, you need to set **spec.schedulerName** of a workload to **volcano** or set the default cluster scheduler to **volcano**. The application scaling priority policy function applies not to workloads with no resource limit and requested resources configured.
- If the default priority policy is used, Volcano Scheduler schedules workloads based on the priorities of yearly/monthly nodes, pay-per-use nodes, and virtual-kubelet nodes (scaling pods to CCI). However, the priorities cannot be fully implemented, because Volcano Scheduler takes scheduling results into account from multiple dimensions rather than just one.
- Volcano Scheduler must balance scheduling performance with scheduling results. When there are a large number of schedulable nodes in a cluster, it selects only some of them for scheduling to ensure scheduling performance

and will not find the best global scheduling solution. For details, see **Scheduler Performance Tuning**. This behavior conflicts with the scaling priority policies. But you can make Volcano Scheduler select all nodes for scheduling by **adjusting the proportion of nodes that can be scheduled by Volcano Scheduler**.

Overview

After the application scaling priority policy is enabled, the **Balancer** and **BalancerPolicyTemplate** CRDs are added to a cluster, and the default scaling priority policy is created. For details, see **Applying the Default Application Scaling Priority Policy**. Volcano Scheduler obtains the priority of each node based on the **BalancerPolicyTemplate** CR to control the pod scheduling priority during application scale-out. In addition, it configures the priority during application scale-in based on both **Balancer** and **BalancerPolicyTemplate** CRs.

- The **BalancerPolicyTemplate** CRDs are used to define priority policies. For example, in the default scaling priority policy, the **BalancerPolicyTemplate** CR assigns the highest priority to yearly/monthly nodes, followed by pay-per-use nodes, and the lowest priority to virtual-kubelet nodes (scaling pods to CCI) by default.

  The **BalancerPolicyTemplate** CRs cannot be updated.
- The **Balancer** CRDs are used to declare the application scope of scaling priorities. When creating a **Balancer** CR, you can specify a workload in a namespace, a specific Deployment, or a specific ReplicaSet as the application scope.

A **Balancer** CR corresponds to a **BalancerPolicyTemplate** CR. They work together to determine which priority policies are applied to specific workloads.

In Volcano Scheduler's default scaling priority policy, the **BalancerPolicyTemplate** CR classifies yearly/monthly nodes, pay-per-use nodes, and virtual-kubelet nodes (scaling pods to CCI) into different priorities. Volcano Scheduler takes these priorities into account during scale-out and preferentially schedules new pods to the yearly/monthly nodes with higher priorities.

Volcano Scheduler applies annotations to pods within the application scope specified by the **Balancer** CR based on the priorities set by the **BalancerPolicyTemplate** CR. It may add the following annotations to a pod that meets the conditions:

- **openvessel.io/workload-balancer-score**: indicates a pod's score, which is higher if the pod is on a high-priority node.
- **autoscaling.volcano.sh/dominated-by-balancer**: specifies the **Balancer** CR that controls the current pod. Pods with low scores are preferentially scaled in.

NOTE

If the existing pods already have the community supported **controller.kubernetes.io/pod-deletion-cost** annotation added, scale-in will be performed based on the priority defined by this annotation. If two pods have the same value for this annotation, the **openvessel.io/workload-balancer-score** annotation will be used to determine which pod to scale-in.

You can configure the **workload_balancer_score_annotation_key** parameter in advanced settings to specify the annotation key for storing pod scores. For details, see **Configuring a Scaling Priority Policy for a Third-Party Workload**.

Configuring an Application Scaling Priority Policy

Step 1 Install Volcano Scheduler in a cluster and enable the application scaling priority policy. The default scaling priority policy will be created in the cluster.

1. Obtain a default **Balancer** CR.

```
# kubectl get balancer default-balancer -oyaml

apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: default-balancerpolicytemplate
  targets:
  - namespaceSelector:
      matchExpressions:
        - key: kubernetes.io/metadata.name
          operator: Exists
  weight: 10
```

2. Obtain a default **BalancerPolicyTemplate** CR.

```
# kubectl get balancerpolicytemplate default-balancerpolicytemplate -oyaml

apiVersion: autoscaling.volcano.sh/v1alpha1
kind: BalancerPolicyTemplate
metadata:
  name: default-balancerpolicytemplate
spec:
  policy:
    policyName: Priority
    priorities:
      priorityGroups:
      - priority: 10
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - post-paid
      - priority: 100
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - pre-paid
      - priority: 1
        requirements:
        - key: kubernetes.io/role
          operator: In
          values:
          - virtual-kubelet
          - bursting
```

For details about the parameters, see **Applying the Default Application Scaling Priority Policy**.

Step 2 Deploy a workload and set the number of pods to 1.

Pods of the current workload are preferentially scheduled to yearly/monthly nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: balancer-test
  namespace: default
  labels:
    virtual-kubelet.io/burst-to-cci: 'auto'  #If the resources of a cluster are not enough, pods in this cluster
```



```
can be deployed on CCI.
spec:
  replicas: 1
  selector:
    matchLabels:
      app: balancer-test
  template:
    metadata:
      labels:
        app: balancer-test
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: IfNotPresent
        name: container-1
        resources:
          limits:
            cpu: 250m
            memory: 512Mi
          requests:
            cpu: 250m
            memory: 512Mi
      schedulerName: volcano
```

Step 3 Increase the number of workload pods to 5.

Pods of the current workload are preferentially scheduled to yearly/monthly nodes. If there are not enough yearly/monthly nodes, these pods will be preferentially scheduled to pay-per-use nodes. If there are not enough pay-per-use nodes, these pods will be scheduled to virtual-kubelet nodes (scaling pods to CCI).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: balancer-test
  namespace: default
  labels:
    virtual-kubelet.io/burst-to-cci: 'auto'  #If the resources of a cluster are not enough, pods in this cluster
can be deployed on CCI.
spec:
  replicas: 5
  selector:
    matchLabels:
      app: balancer-test
  template:
    metadata:
      labels:
        app: balancer-test
    spec:
      containers:
      - image: nginx:latest
        imagePullPolicy: IfNotPresent
        name: container-1
        resources:
          limits:
            cpu: 250m
            memory: 512Mi
          requests:
            cpu: 250m
            memory: 512Mi
      schedulerName: volcano
```

Step 4 View the scores of pods.

1. Pods on a yearly/monthly node:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     annotations:
   ```



```
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer  #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score: "100" #Priority of the current yearly/monthly node, which also indicates the pods' score
...
  nodeName: 192.168.20.100 #A yearly/monthly node
```

2. Pods on a pay-per-use node:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer  #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score "10"  #Priority of the current pay-per-use node, which also indicates the pods' score
    ...
  nodeName: 192.168.20.196 #A pay-per-use node
```

3. Pods on a virtual-kubelet node (scaling pods to CCI):

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    autoscaling.volcano.sh/dominated-by-balancer: default-balancer  #The Balancer CR named default-balancer controls the scaling priority of the current pods.
    openvessel.io/workload-balancer-score: "1"  #Priority of the current virtual-kubelet node, which also indicates the pods' score
    ...
  nodeName: virtual-kubelet #A virtual-kubelet node
```

Step 5 Gradually reduce the number of the workload pods.

Pods on virtual-kubelet nodes (scaling pods to CCI) are deleted first, followed by pods on pay-per-use nodes and those on yearly/monthly nodes.

----End

Applying the Default Application Scaling Priority Policy

When the default application scaling priority policy is used, the following default CRs are present in a cluster:

- A **Balancer** CR:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: default-balancerpolicytemplate
  targets:
  - namespaceSelector:
      matchExpressions:
        - key: kubernetes.io/metadata.name
          operator: Exists
  weight: 10
```

Table 6-21 Key parameters of a Balancer CR

Field	Description	Type	Remarks
metadata.name	Name	String	This field is mandatory.

Field	Description	Type	Remarks
spec.balancerPolicyTemplateName	Name of the priority policy	String	This field is mandatory. The value is the name of the corresponding **BalancerPolicyTemplate** CR in the cluster.

spec.targets	Application scope of the priority policy	Slice	This field is mandatory. Example:
			• Applying to applications in the **default** namespace:
			`spec:`
			`  targets:`
			`  - namespaceSelector:`
			`  matchLabels:`
			`  kubernetes.io/metadata.name: default`
			• Applying to applications in multiple namespaces like **default**, **other**, and **another**:
			`spec:`
			`  targets:`
			`  - namespaceSelector:`
			`  matchExpressions:`
			`  - key: kubernetes.io/metadata.name`
			`  operator: In`
			`  values:`
			`  - default`
			`  - other`
			`  - another`
			• Applying to applications in all namespaces:
			`spec:`
			`  targets:`
			`  - namespaceSelector:`
			`  matchExpressions:`
			`  - key: kubernetes.io/metadata.name`
			`  operator: Exists`
			• Only applying to Deployments which are named in the format of *xxx-xxx-xxx*:
			`spec:`
			`  targets:`
			`  - objectSelectors:`
			`  - name: xxx-xxx-xxx`
			`  kind: Deployment`
			`  apiVersion: apps/v1`
			• Only applying to Deployments which are named in the format of *xxx-xxx-xxx* and are in the **default** namespace:
			`spec:`
			`  targets:`
			`  - namespaceSelector:`
			`  matchLabels:`
			`  kubernetes.io/metadata.name: default`
			`  objectSelectors:`
			`  - name: xxx-xxx-xxx`
			`  kind: Deployment`
			`  apiVersion: apps/v1`

Field	Description	Type	Remarks
spec.weight	Weight of the priority policy	int32	This field is mandatory. When there are multiple **Balancer** CRs in a cluster, an application may fall within the scope of more than one of them. In such cases, the **Balancer** CR with the highest weight will be applied.

- A **BalancerPolicyTemplate** CR:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: BalancerPolicyTemplate
metadata:
  name: default-balancerpolicytemplate
spec:
  policy:
    policyName: Priority
    priorities:
      priorityGroups:
      - priority: 10
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - post-paid
      - priority: 100
        requirements:
        - key: node.cce.io/billing-mode
          operator: In
          values:
          - pre-paid
      - priority: 1
        requirements:
        - key: kubernetes.io/role
          operator: In
          values:
          - virtual-kubelet
          - bursting
```

Table 6-22 Key parameters of a BalancerPolicyTemplate CR

Field	Description	Type	Remarks
metadata.name	Name	String	This field is mandatory.
spec.policy	Content of the priority policy	Struct	This field is mandatory.
spec.policy.policyname	Name of the priority policy	String	This field is mandatory. Only the priority policy named **Priority** is supported.

Field	Description	Type	Remarks
spec.policy.priorities.priorityGroups	Specific priority defined in the priority policy	Slice	This field is mandatory. Example: • Setting the priority of a yearly/monthly node to **100**: priorityGroups: - priority: 100 requirements: - key: node.cce.io/billing-mode operator: In values: - pre-paid • Setting the priority of a pay-per-use node to **10**: priorityGroups: - priority: 10 requirements: - key: node.cce.io/billing-mode operator: In values: - post-paid • Setting the priority of a virtual-kubelet or bursting node to **1**: priorityGroups: - priority: 1 requirements: - key: kubernetes.io/role operator: In values: - virtual-kubelet - bursting

Customizing an Application Scaling Priority Policy

The **BalancerPolicyTemplate** CRDs are used to define priority policies. If you need to customize an application scaling priority policy, you need to modify the **BalancerPolicyTemplate** CR.

NOTE

If there are multiple **BalancerPolicyTemplate** CRs in a cluster, they will all affect the scaling result. Therefore, if the default scaling priority policy is not in use, run the following command to delete it:

```
kubectl delete balancerpolicytemplate default-balancerpolicytemplate
```

Assume that **during scale-out, a workload is preferentially scheduled to a node running HCE 2.0 and then to a node running Euler. During scale-in, Volcano Scheduler first deletes the workload pods on the node running Euler and then deletes the pods on the node running HCE 2.0.**

Step 1 Write a new **BalancerPolicyTemplate** CR.

```
vim new-balancerpolicytemplate.yaml
```

The content is as follows:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: BalancerPolicyTemplate
metadata:
```



```
  name: new-balancerpolicytemplate
spec:
  policy:
    policyName: Priority
    priorities:
      priorityGroups:
      - priority: 10    # Set the priority of the node running EulerOS to 10.
        requirements:
        - key: os.name  # Label of the Node OS
          operator: In
          values:
          - EulerOS_2.0_SP9x86_64  # The minor version number of the OS may be involved. You can add the
minor version number as needed.
      - priority: 100   # Set the priority of the node running HCE 2.0 to 100.
        requirements:
        - key: os.name  # Label of the Node OS
          operator: In
          values:
          - Huawei_Cloud_EulerOS_2.0_x86_64
```

Step 2 Create a **BalancerPolicyTemplate** CR.

```
kubectl create -f new-balancerpolicytemplate.yaml
```

Step 3 Modify **default-balancer**. You can also create a **Balancer** CR as needed.

```
kubectl edit balancer default-balancer
```

The modified content is as follows:

```
apiVersion: autoscaling.volcano.sh/v1alpha1
kind: Balancer
metadata:
  name: default-balancer
spec:
  balancerPolicyTemplateName: new-balancerpolicytemplate
  targets:
  - namespaceSelector:
      matchExpressions:
        - key: kubernetes.io/metadata.name
          operator: Exists
  weight: 10
```

Step 4 Check whether the value of **openvessel.io/workload-balancer-score** in each pod meets the expectation.

The value of **openvessel.io/workload-balancer-score** in each pod on the node running EulerOS is set to **10**. The value of **openvessel.io/workload-balancer-score** in each pod on the node running HCE 2.0 is set to **100**.

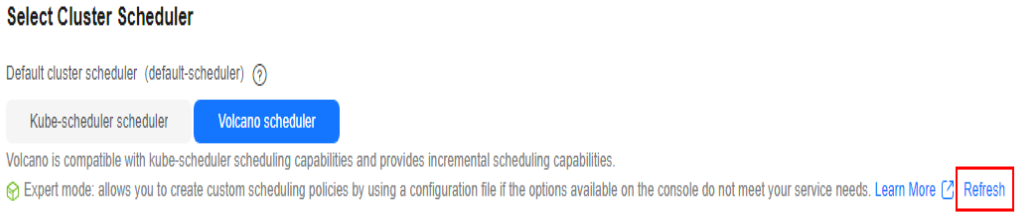
----End

Configuring a Scaling Priority Policy for a Third-Party Workload

For a workload that is not a Deployment but is managed by CRDs, you can configure the scaling priority policies for the workload in the **Advanced Settings** area, so that Volcano can support the scaling priority policies of the workload.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Settings** and click the **Scheduling** tab. Select **Volcano scheduler**, find the expert mode, and click **Refresh**.



Step 3 In the navigation pane, choose **Add-ons**, locate **Volcano Scheduler**, click **Install** or **Edit**, and adjust the configuration parameters in the **Parameters** area.

Step 4 Specify the type of the third-party workload to be supported. The following is an example in JSON format:

```
{
   "default_scheduler_conf": {
...
   },
   "workload_balancer_score_annotation_key": "controller.kubernetes.io/pod-deletion-cost",
   "workload_balancer_third_party_types": "apps.kruise.io/v1alpha1/clonesets,apps.kruise.io/v1beta1/
statefulsets"
}
```

- **workload_balancer_score_annotation_key**: specifies the score annotation key of a pod. **openvessel.io/workload-balancer-score** or **controller.kubernetes.io/pod-deletion-cost** is supported. Setting this parameter to other values will cause volcano to exit abnormally.
- **workload_balancer_third_party_types**: The value is a character string consisting of the group, version, and kind of a third-party workload, and CRDs are separated by commas (,).

  The value represented the workload kind needs to be in a plural form, for example, **apps.kruise.io/v1alpha1/clonesets,apps.kruise.io/v1beta1/statefulsets**. If it is in a non-plural form, for example, **apps.kruise.io/v1alpha1/cloneset**, the corresponding CRD cannot be monitored.

  If the format is incorrect, volcano will exit abnormally. If the specified CRD is not present in the cluster, the application scaling priority policy cannot work properly.

If the CRD is configured to scale in based on priority, the controller overseeing it can detect the pod score annotation while scaling and adjust the sequence accordingly.

----End

Appendix: Adjusting the Proportion of Nodes That Can Be Scheduled by Volcano Scheduler

Write a volcano-scheduler resource object.

```
kubectl edit deploy volcano-scheduler -nkube-system
```

The content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: volcano-scheduler
    app.kubernetes.io/managed-by: Helm
    release: cceaddon-volcano
```



```
  name: volcano-scheduler
  namespace: kube-system
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: volcano-scheduler
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: volcano-scheduler
        release: cceaddon-volcano
    spec:
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: app
                  operator: In
                  values:
                  - volcano-scheduler
              topologyKey: topology.kubernetes.io/zone
            weight: 100
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - volcano-scheduler
            topologyKey: kubernetes.io/hostname
      containers:
      - command:
        - /bin/sh
        - -c
        - /volcano-scheduler --leader-elect=true --lock-object-namespace=kube-system
          --feature-
gates=CSIMigrationFlexVolumeFuxi=true,CSIMigrationFlexVolumeFuxiComplete=true,MultiGPUScheduling=tr
ue
          --kube-api-qps=200 --alsologtostderr --listen-address=$(MY_POD_IP):8080
          --enable-healthz=true --healthz-address=$(MY_POD_IP):11251 --enable-metrics=true --percentage-
nodes-to-find=100
          --scheduler-conf=/volcano.scheduler/default-scheduler.conf -v=3 1>>/var/log/volcano/volcano-
scheduler.log
```

--percentage-nodes-to-find=100 specifies that Volcano Scheduler can find all nodes in a cluster during scheduling selection.

6.6 Cloud Native Hybrid Deployment

6.6.1 Overview

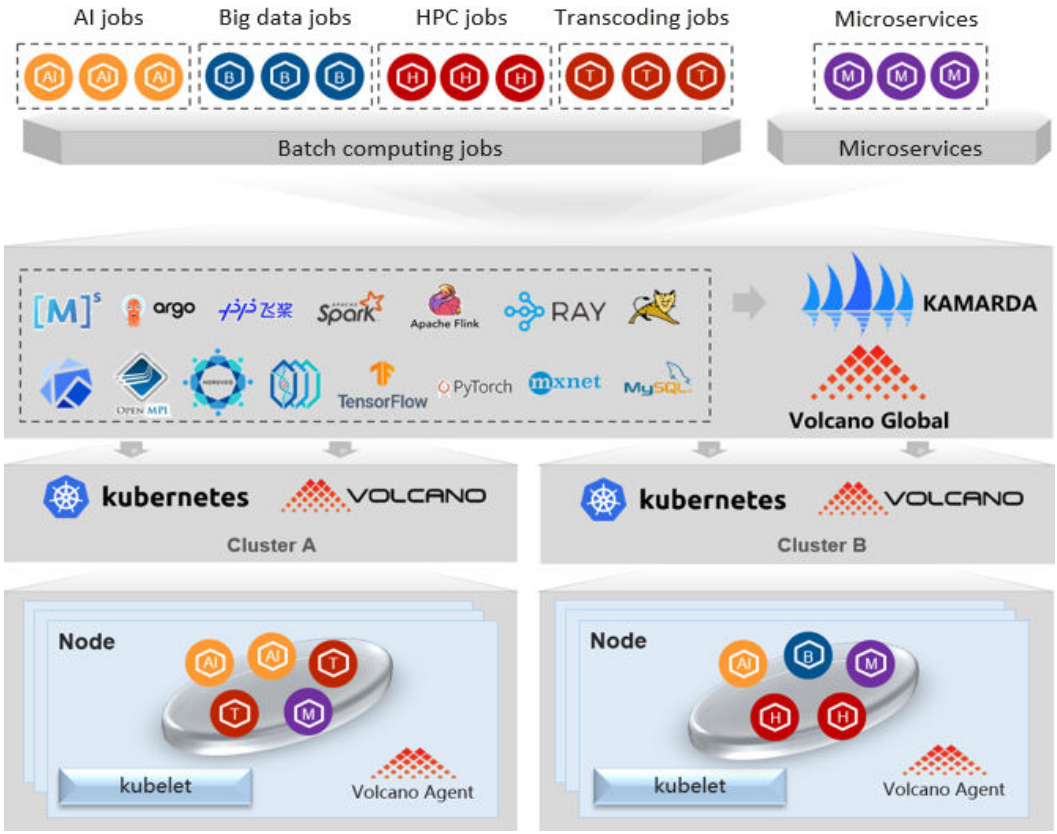
With the rapid development of cloud native technologies, many applications are going cloud native. From 2021 to 2022, the total number of cloud native applications in Kubernetes clusters has increased by more than 30% year-on-year.

Kubernetes is becoming the platform for running almost anything — a virtual "operating system" for cloud native applications in the cloud era. However, further research shows that the CPU usage of most user nodes in Kubernetes clusters is less than 15%. According to a survey conducted on a range of clients, the primary causes of low resource utilization may be summed up as follows, if interference elements like idle resources and package activities are taken out:

1. Nodes are deployed in different clusters. They cannot share compute resources with each other, resulting in an increase in resource fragments.
2. The node specifications are not ideal for applications that undergo frequent changes. At first, the node specifications match the application requirements, resulting in a high resource allocation rate. However, as the applications evolve, their resource demands change, causing a significant difference in the ratio of requested resources to node specifications. This leads to a decrease in the allocation rate of node resources and an increase in compute resource fragmentation.
3. There are a large number of reserved resources. Online services experience daily peaks and troughs. To ensure service performance and stability, users apply for resources based on peak usage, which may result in many idle resources in the cluster during certain times.
4. Online and offline services are deployed in separate Kubernetes clusters, and resources cannot be shared between them at different time. This means that during off-peak hours for online services, the resources cannot be used by offline services.

These are typical instances of the difficulties encountered when creating cloud native applications. Various deployment solutions are needed for different service architectures during cloud native progress. Applications with varying architectures evolve at different paces, so development teams must balance service performance with service quality. How can these complex scenarios be made simpler so that customers can gradually increase resource usage and reduce costs?

CCE has created a cloud native hybrid deployment solution based on the Volcano and Kubernetes ecosystems, which helps users improve resource utilization, reduce costs, and increase efficiency. This solution is the result of years of exploration and practice in hybrid deployment.



As shown in this picture, hybrid deployment involves more than just merging small clusters into a larger one and deploying multiple services within that cluster. It requires the proper deployment of user applications and ensuring that the necessary resources are available to support their operation. This picture highlights the core designs in the cloud native hybrid deployment solution: unified scheduling of resources across all domains and hierarchical resource management.

Unified Scheduling of Resources Across All Domains and Hierarchical Resource Management

Unified scheduling of resources across all domains

Unified scheduling of resources across all domains ensures smooth scheduling of cross-cloud, cross-cluster resources in distributed cloud environments. This also enables unified scheduling of both online and offline services.

- Volcano uses static analysis to gather static features of applications, such as CPU, memory, storage, and GPU requirements, as well as affinity between applications, regions, and cloud platforms.
- Volcano then connects with the monitoring system to gather dynamic data from various cloud platforms, clusters, and running applications. This data is analyzed and used to learn about service fluctuations (daily, weekly, or monthly) and service types (CPU-sensitive, L3 cache-sensitive, or memory-sensitive).
- Finally, Volcano schedules applications to suitable environments using its flexible and customizable scheduling policies like prediction-based intelligent scheduling policy, service-based bin packing, rescheduling, and running status-based resource oversubscription.

Volcano manages resources on the distributed cloud platforms in a unified manner and schedules different types of applications to proper locations. This effectively solves the problem of resource fragmentation caused by multiple clusters and node specification mismatch caused by application updates. It frees users from complex resource planning and changes brought about by application iterations.

Hierarchical resource management

Hierarchical resource management ensures that applications can access to the necessary resources in their designated environments after they have been scheduled.

Resources are isolated from multiple dimensions like CPU, L3 cache, memory, network, and storage based on Huawei Cloud EulerOS 2.0. Additionally, resources, mainly in the kernel mode, with some in the user mode, can be quickly preempted in milliseconds or evicted in seconds to ensure the quality of online services.

- Resource isolation measures (such as CPU core binding, NUMA affinity, tidal affinity, and network bandwidth control) ensure the resource-sensitive services to meet their SLOs.
- Resource priority control (such as hierarchical CPU suppression, memory tiering, network priority control, and disk I/O priority control) improves resource allocation and has little or no impact on the SLOs of high-priority services.

Hierarchical resource management provides a basis for deploying online services and hybrid deployment of online and offline services. This resolves problems that a large number of resources are reserved for applications and resources cannot be reused at different times.

Online and Offline Jobs

Jobs can be classified into online jobs and offline jobs based on whether services are always online.

- **Online job**: Such jobs run for a long time, with regular traffic surges, tidal resource requests, and high requirements on SLA, such as advertising and e-commerce services.
- **Offline jobs**: Such jobs run for a short time, have high computing requirements, and can tolerate high latency, such as AI and big data services.

Features

Function	Description	Documentation
Dynamic resource oversubscription	Based on the types of online and offline jobs, Volcano scheduling is used to utilize the resources that are requested but not used in the cluster (the difference between the number of requested resources and the number of used resources) for resource oversubscription and hybrid deployment to improve cluster resource utilization.	**Dynamic Resource Oversubscription**

Function	Description	Documentation
CPU Burst	CPU Burst is an elastic traffic limiting mechanism that allows temporarily exceeding the CPU limit to reduce the long-tail response time of services and improve the quality of latency-sensitive services.	CPU Burst
Egress network bandwidth guarantee	The egress network bandwidth used by online and offline services is balanced to ensure sufficient network bandwidth for online services.	Guaranteed Egress Network Bandwidth

6.6.2 Enabling Cloud Native Hybrid Deployment

Prerequisites

- A CCE standard or Turbo cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.9-r0 or later
  - v1.25: v1.25.4-r0 or later
- Volcano of v1.10.0 or later has been installed in the cluster.

Notes and Constraints

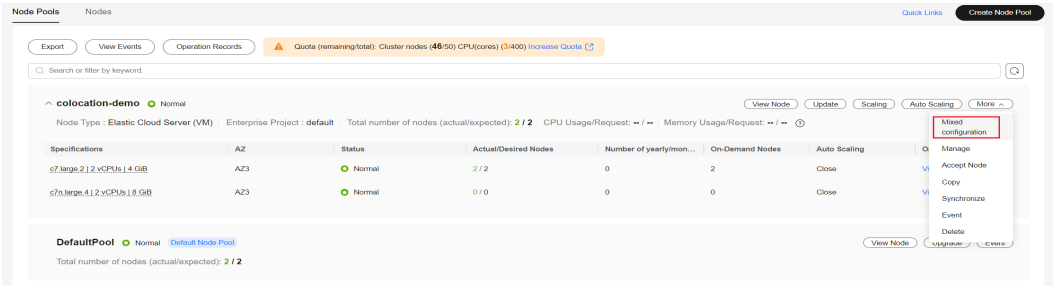
- After cloud native hybrid deployment is enabled, the Volcano scheduler will automatically enable the oversubscription add-on. Ensure the add-on remains enabled during the usage of cloud native hybrid deployment.
- The hybrid deployment agent is deployed in affinity mode as a DaemonSet only on an x86 node that runs Huawei Cloud EulerOS 2.0.
- The default node pool does not allow the modification of hybrid configurations.

Procedure

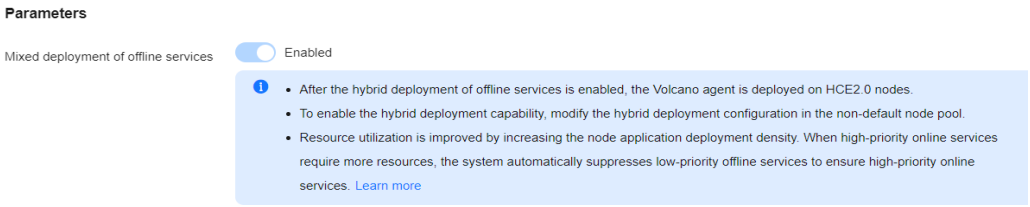
Cloud native hybrid deployment is managed by node pool. You need to enable hybrid deployment in the target node pool and configure it accordingly. By default, all hybrid deployment capabilities are enabled and default parameter values are used in the configuration. You can modify the configuration as needed.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Nodes** in the navigation pane and choose **More** > **Mixed configuration** in the **Operation** column of the target node pool.



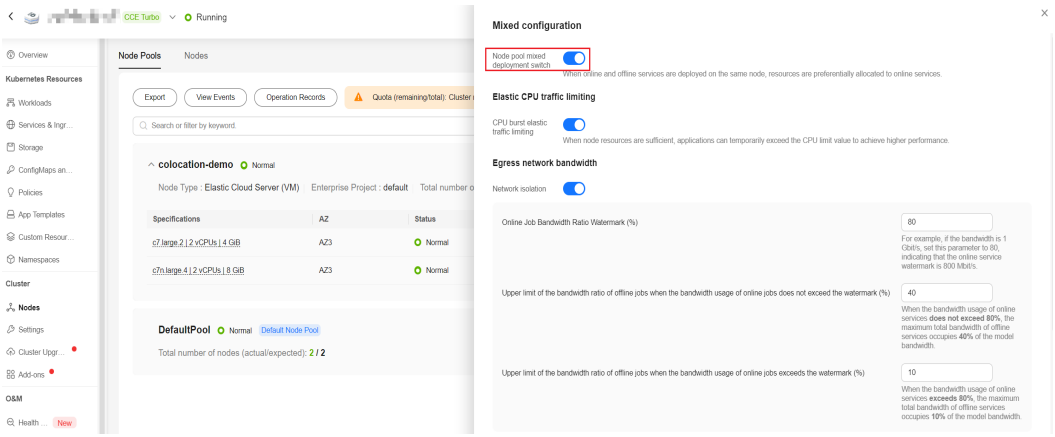
Step 3 If Volcano does not have hybrid deployment of online and offline services enabled, you can enable this function on the configuration page that is displayed and continue with the configuration after installing or updating Volcano.



Step 4 On the **Mixed configuration** page, enable hybrid deployment.

NOTE

When hybrid deployment is enabled, the system will check if you have enabled kubelet oversubscription in hybrid deployment. If so, you will be prompted to migrate this configuration to cloud native hybrid deployment automatically. For details about the migration, see **Switching kubelet Oversubscription to Resource Oversubscription in Cloud Native Hybrid Deployment**.



You can configure the parameters listed in the following table.

Parameter	Default Setting	Description
CPU burst elastic traffic limiting	Enabled	After this function is enabled, the CPU usage of a pod can temporarily exceed the CPU limit when there are enough node resources. This reduces response latency for long-tail services. For details, see **CPU Burst**.

Parameter	Default Setting	Description
Egress network bandwidth	Enabled	In CCE Turbo clusters, networks of online and offline services can be isolated. For details, see **Guaranteed Egress Network Bandwidth**.
Resource Oversubscription	Enabled	Node resources can be oversubscribed by collecting node load information in real time and detecting the resources that have been allocated but not used by the nodes. Select the resource type that requires oversubscription. By default, oversubscription is enabled for both CPU and memory. For details, see **Dynamic Resource Oversubscription**. **NOTE** • If the cluster version does not meet specific requirements, resource oversubscription will not take effect. For details, see **Table 6-23**. • When you modify the resource oversubscription configuration: – You can add oversubscribed resources (such as CPU and memory) at any time. – You can reduce oversubscribed resource types, for example, from CPU and memory to only CPU, only when the resource allocation rate does not exceed 100%.

Step 5 Click **OK**.

----End

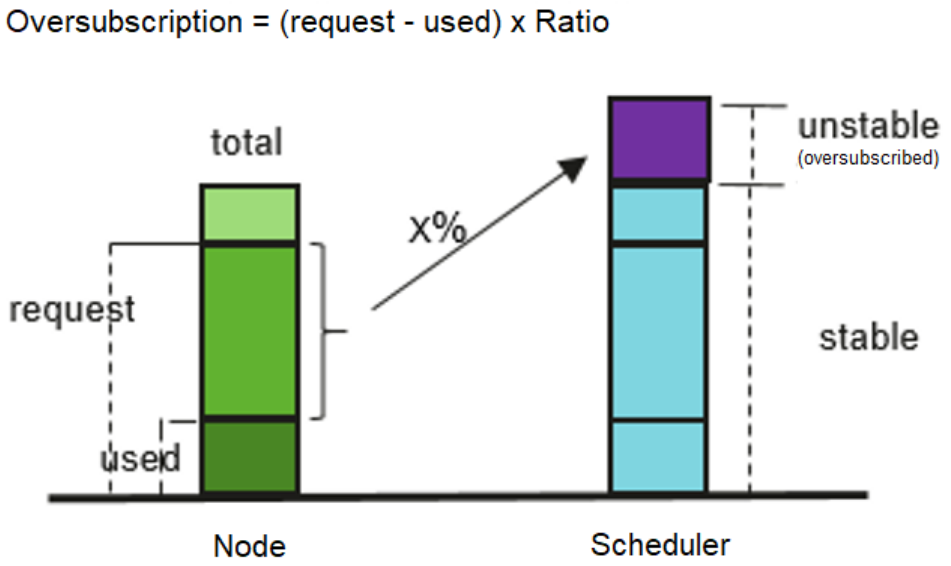
6.6.3 Dynamic Resource Oversubscription

Many services see surges in traffic. To ensure performance and stability, resources are often requested at the maximum needed. However, the surges may ebb very shortly and resources, if not released, are wasted in non-peak hours. Especially for online jobs that request a large quantity of resources to ensure SLA, resource utilization can be as low as it gets.

Resource oversubscription is the process of making use of idle requested resources. Oversubscribed resources are suitable for deploying offline jobs, which focus on throughput but have low SLA requirements and can tolerate certain failures.

Hybrid deployment of online and offline jobs in a cluster can better utilize cluster resources.

Figure 6-25 Resource oversubscription



Features

NOTE

After dynamic resource oversubscription and elastic scaling are enabled in a node pool, oversubscribed resources change rapidly because the resource usage of high-priority applications changes in real time. To prevent frequent node scale-ins and scale-outs, do not consider oversubscribed resources when evaluating node scale-ins.

Hybrid deployment is supported, and CPU and memory resources can be oversubscribed. The key features are as follows:

- Offline jobs preferentially run on oversubscribed nodes.

  If both oversubscribed and non-oversubscribed nodes exist, the former will score higher than the latter and offline jobs are preferentially scheduled to oversubscribed nodes.
- Online jobs can use only non-oversubscribed resources if scheduled to an oversubscribed node.

  Offline jobs can use both oversubscribed and non-oversubscribed resources of an oversubscribed node.
- In the same scheduling period, online jobs take precedence over offline jobs.

  If both online and offline jobs exist, online jobs are scheduled first. When the node resource usage exceeds the upper limit and the node requests exceed 100%, offline jobs will be evicted.
- CPU/Memory resources can be isolation by kernel.

  CPU isolation: Online jobs can quickly preempt CPU resources of offline jobs and suppress the CPU usage of the offline jobs.

  Memory isolation: When system memory resources are used up and OOM Kill is triggered, the kernel evicts offline jobs first.
- kubelet offline jobs obey the following admission rules:

After the pod is scheduled to a node, kubelet starts the pod only when the node resources can meet the pod request (predicateAdmitHandler.Admit). kubelet starts the pod when both of the following conditions are met:

- The total request of pods to be started and online running jobs < allocatable nodes
- The total request of pods to be started and online/offline running job < allocatable nodes+oversubscribed nodes

- Resource oversubscription and hybrid deployment can be configured separately.

  Enabling hybrid deployment of a node pool also enables oversubscription by default. Nodes are then labeled with both **volcano.sh/colocation="true"** and **volcano.sh/oversubscription="true"**. To use hybrid deployment for both online and offline jobs without oversubscription, simply disable oversubscription in hybrid deployment settings. This will remove the **volcano.sh/oversubscription="true"** label.

  The following table lists the features that can be used after hybrid deployment or oversubscription is enabled.

Hybrid Deployment	Oversubscription	Oversubscription Resource	Scenario for Evicting Offline Pods
No	No	No	No
Yes	No	No	The actual resource usage of a node exceeds the upper limit.
No	Yes	Yes	The actual resource usage of a node exceeds the upper limit and the pod requests on the node exceed 100%.
Yes	Yes	Yes	The actual resource usage of a node exceeds the upper limit.

How to Use

Consider the cluster version when determining how to make use of resource oversubscription. For details, see **Table 6-23**.

Oversubscription mode and compatibility mode cannot be used concurrently. Select either of them. If the cluster version does not support cloud native hybrid

deployment, hybrid deployment resource oversubscription and parameter settings do not take effect. To use resource oversubscription, use **kubelet oversubscription**.

Table 6-23 Mapping between cluster versions and resource oversubscription

Cluster Version	Specific Version	Resource Oversubscription	Description
Later than v1.25	None	**Resource Oversubscription in Cloud Native Hybrid Deployment**	None
v1.25	v1.25.4-r0 or later	**Resource Oversubscription in Cloud Native Hybrid Deployment**	None
	Earlier than v.1.25.4-r0 (kubelet oversubscription is used in cluster versions earlier than v.1.25.4-r0. Upgrade the cluster versions to v1.25.4-r0 or later.)	Existing node pools: **kubelet oversubscription** New node pools: **resource oversubscription in cloud native hybrid deployment**	For existing node pools, migrate kubelet oversubscription to resource oversubscription in cloud native hybrid deployment for unified management. For details, see **Switching kubelet Oversubscription to Resource Oversubscription in Cloud Native Hybrid Deployment**.
	Earlier than v1.25.4-r0	**kubelet oversubscription**	kubelet oversubscription cannot be migrated to resource oversubscription in cloud native hybrid deployment.
v1.23	v1.23.9-r0 or later	**Resource Oversubscription in Cloud Native Hybrid Deployment**	None

Cluster Version	Specific Version	Resource Oversubscription	Description
	Earlier than v.1.23.9-r0 (kubelet oversubscription is used in cluster versions earlier than v.1.23.9-r0. Upgrade the cluster versions to v1.23.9-r0 or later.)	Existing node pools: **kubelet oversubscription** New node pools: **resource oversubscription in cloud native hybrid deployment**	For existing node pools, migrate kubelet oversubscription to resource oversubscription in cloud native hybrid deployment for unified management. For details, see **Switching kubelet Oversubscription to Resource Oversubscription in Cloud Native Hybrid Deployment**.
	Earlier than v1.23.9-r0, but later than or equal to v1.23.5-r0	**kubelet oversubscription**	kubelet oversubscription cannot be migrated to resource oversubscription in cloud native hybrid deployment.
v1.21	v1.21.7-r0 or later	**kubelet oversubscription**	None
v1.19	v1.19.16-r4 or later	**kubelet oversubscription**	None

When cloud native hybrid deployment is enabled, resource oversubscription is enabled by default. For details, see **Resource Oversubscription in Cloud Native Hybrid Deployment**. The cloud native hybrid deployment add-on, **volcano-agent**, reports oversubscribed resources and evicts service load from nodes. Its core features include CPU/memory suppression, dynamic resource oversubscription, CPU burst, and hierarchical QoS control of egress.

To maintain compatibility with earlier versions, the function of manually setting kubelet parameters to enable resource oversubscription is still available. For details, see **Compatible kubelet Oversubscription (Not Recommended)**. However, this solution only provides basic functions and has not been updated. It does not support new features like CPU burst and hierarchical QoS control of egress.

Resource Oversubscription in Cloud Native Hybrid Deployment

NOTICE

Specifications

- Cluster version
  - v1.23: v1.23.9-r0 or later
  - v1.25: v1.25.4-r0 or later
- Cluster type: CCE standard or Turbo
- Node OS: Huawei Cloud EulerOS 2.0
- Node type: ECS on x86
- Volcano version: 1.10.0 or later

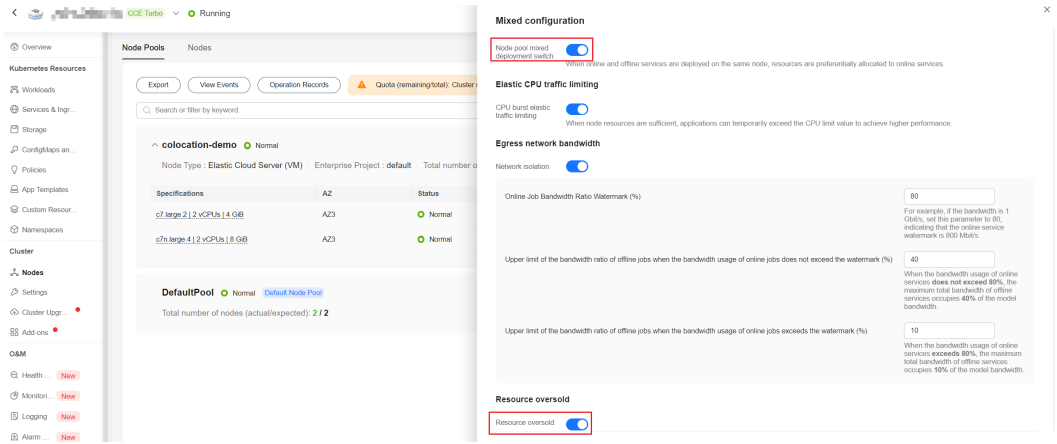
Constraints

- Before enabling oversubscription, ensure that the overcommit add-on is not enabled on Volcano.
- Running pods cannot be converted between online and offline services. To convert services, rebuild pods.
- If you have set **cpu-manager-policy** to statically bind CPU cores on a node, do not assign the QoS class of Guaranteed to offline pods. This is because offline pods may occupy the CPUs of online pods, leading to an online pod startup failure and offline pods failing to start even though they have been successfully scheduled. To prevent this, switch the pods to online pods if CPU core binding is required.
- If **cpu-manager-policy** is set to static CPU core binding on a node, do not bind CPU cores to all online pods. This is because doing so can cause online pods to occupy all available CPU or memory resources, leaving only a small number of oversubscribed resources.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More** > **Mixed configuration**.

Ensure that node pool hybrid deployment and resource oversubscription are enabled. For details, see **Procedure**.



Step 3 (Optional) Adjust resource oversubscription parameters.

Table 6-24 Resource oversubscription parameters

Parameter	Description
High CPU Eviction Threshold (%)	When the CPU usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is **80**, indicating that offline job eviction is triggered when the CPU usage of a node exceeds 80%.
Low CPU Eviction Threshold (%)	When the CPU usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the CPU usage of the node is lower than the lower limit. The default value is **30**, indicating that offline jobs are accepted again when the CPU usage of a node is lower than 30%.
High Memory Eviction Threshold (%)	When the memory usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is **60**, indicating that offline job eviction is triggered when the memory usage of a node exceeds 60%.
Low Memory Eviction Threshold (%)	When the memory usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the memory usage of the node is lower than the lower limit. The default value is **30**, indicating that offline jobs are accepted again when the memory usage of a node is lower than 30%.

Step 4 Check the Volcano configuration.

```
kubectl edit cm volcano-scheduler-configmap -n kube-system
```

Check the oversubscription configuration in **volcano-scheduler-configmap**. Ensure that the add-on configuration does not contain the overcommit add-on. If **- name: overcommit** exists, delete this configuration.

```
...
data:
  volcano-scheduler.conf: |
    actions: "allocate, backfill, preempt"    # Configure a preemption action.
    tiers:
    - plugins:
      - name: gang
        enablePreemptable: false
        enableJobStarving: false
      - name: priority
      - name: conformance
      - name: oversubscription
    - plugins:
      - name: drf
```



```
    - name: predicates
    - name: nodeorder
    - name: binpack
  - plugins:
    - name: cce-gpu-topology-predicate
    - name: cce-gpu-topology-priority
    - name: cce-gpu
...
```

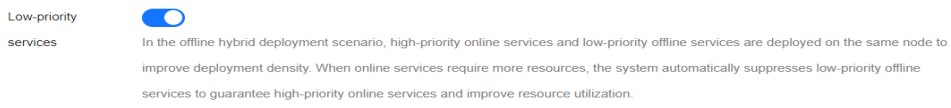
Step 5 Create resources at a high- and low-priorityClass, respectively.

```
cat <<EOF | kubectl apply -f -

apiVersion: scheduling.k8s.io/v1
description: Used for high priority pods
kind: PriorityClass
metadata:
  name: volcano-production
preemptionPolicy: PreemptLowerPriority
value: 999999
---
apiVersion: scheduling.k8s.io/v1
description: Used for low priority pods
kind: PriorityClass
metadata:
  name: volcano-free
preemptionPolicy: PreemptLowerPriority
value: -90000

EOF
```

Step 6 Deploy online and offline jobs.

For both online and offline jobs, set **schedulerName** to **volcano** to enable Volcano.

- If you enable low-priority services when creating an offline job workload, the annotation **volcano.sh/qos-level: "-1"** will be added to it.

  Set **priorityClassName** to **volcano-free**.

  ```
  kind: Deployment
  apiVersion: apps/v1
  spec:
    replicas: 4
    template:
      metadata:
        annotations:
          metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
          volcano.sh/qos-level: "-1" # Offline job annotation
      spec:
        schedulerName: volcano             # Volcano is used.
        priorityClassName: volcano-free        # volcano-free priorityClass
        ...
  ```

- For online jobs, set **priorityClassName** to **volcano-production**.

  ```
  kind: Deployment
  apiVersion: apps/v1
  spec:
    replicas: 4
    template:
      metadata:
        annotations:
          metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
      spec:
  ```



```
    schedulerName: volcano          # Volcano is used.
    priorityClassName: volcano-production   # volcano-production priorityClass
    ...
```

Step 7 Run the following command to check the number of oversubscribed resources and the resource usage:

```
kubectl describe node <nodeIP>
# kubectl describe node 192.168.0.0
Name:              192.168.0.0
Roles:             <none>
Labels:            ...
                   volcano.sh/oversubscription=true
Annotations:       ...
                   volcano.sh/oversubscription-cpu: 2335
                   volcano.sh/oversubscription-memory: 341753856
Allocatable:
  cpu:             3920m
  memory:          6263988Ki
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource         Requests      Limits
  --------         --------      ------
  cpu              4950m (126%)  4950m (126%)
  memory           1712Mi (27%)  1712Mi (27%)
```

----End

Compatible kubelet Oversubscription (Not Recommended)

NOTICE

Specifications

- Cluster version
  - v1.19: v1.19.16-r4 or later
  - v1.21: v1.21.7-r0 or later
  - v1.23: v1.23.5-r0 or later
  - v1.25 or later
- Cluster type: CCE standard or Turbo
- Node OS: EulerOS 2.9 (kernel-4.18.0-147.5.1.6.h729.6.eulerosv2r9.x86_64) or Huawei Cloud EulerOS 2.0
- Node type: ECS
- Volcano version: 1.7.0 or later

Constraints

- Before enabling oversubscription, ensure that the overcommit add-on is not enabled on Volcano.
- Modifying the label of an oversubscribed node does not affect the running pods.
- Running pods cannot be converted between online and offline services. To convert services, rebuild pods.
- If the label **volcano.sh/oversubscription=true** is configured for a node in the cluster, the **oversubscription** configuration must be added to the Volcano add-on. Otherwise, the scheduling of oversold nodes will be abnormal. Ensure that you have correctly configure labels because the scheduler does not check the add-on and node configurations. For details, see **Table 6-25**.
- To disable oversubscription, perform the following operations:
  - Remove the **volcano.sh/oversubscription** label from the oversubscribed node.
  - Set **over-subscription-resource** to **false**.
  - Modify the configmap of Volcano Scheduler named **volcano-scheduler-configmap** and remove the oversubscription add-on.
- If you have set **cpu-manager-policy** to statically bind CPU cores on a node, do not assign the QoS class of Guaranteed to offline pods. This is because offline pods may occupy the CPUs of online pods, leading to an online pod startup failure and offline pods failing to start even though they have been successfully scheduled. To prevent this, switch the pods to online pods if CPU core binding is required.
- If **cpu-manager-policy** is set to static CPU core binding on a node, do not bind CPU cores to all online pods. This is because doing so can cause online pods to occupy all available CPU or memory resources, leaving only a small number of oversubscribed resources.

If the label **volcano.sh/oversubscription=true** is configured for a node in the cluster, the **oversubscription** configuration must be added to the Volcano add-on.

Otherwise, the scheduling of oversold nodes will be abnormal. For details about the related configuration, see **Table 6-25**.

Ensure that you have correctly configure labels because the scheduler does not check the add-on and node configurations.

Table 6-25 Configuring oversubscription labels for scheduling

Oversubscription in Add-on	Oversubscription Label on Node	Scheduling
Yes	Yes	Triggered by oversubscription
Yes	No	Triggered
No	No	Triggered
No	Yes	Not triggered or failed. Avoid this configuration.

Step 1 Use kubectl to access the cluster.

Step 2 Check the Volcano configuration.

```
kubectl edit cm volcano-scheduler-configmap -n kube-system
```

Check the oversubscription configuration in **volcano-scheduler-configmap**. Ensure that the add-on configuration does not contain the overcommit add-on. If **- name: overcommit** exists, delete this configuration.

```
...
data:
  volcano-scheduler.conf: |
    actions: "allocate, backfill, preempt"    # Configure a preemption action.
    tiers:
    - plugins:
      - name: gang
        enablePreemptable: false
        enableJobStarving: false
      - name: priority
      - name: conformance
      - name: oversubscription
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
...
```

Step 3 Enable node oversubscription.

A label can be configured to use oversubscribed resources only after the oversubscription feature is enabled for a node. Related nodes can be created only in a node pool. To enable the oversubscription feature, perform the following steps:

1. Create a node pool.

2. Choose **Manage** in the **Operation** column of the created node pool.
3. On the **Manage Components** page, enable **Node oversubscription feature (over-subscription-resource)** and click **OK**.

Step 4 Set the node oversubscription label.

The **volcano.sh/oversubscription** label needs to be configured for an oversubscribed node. If this label is set for a node and the value is **true**, the node is an oversubscribed node. Otherwise, the node is not an oversubscribed node.

```
kubectl label node 192.168.0.0 volcano.sh/oversubscription=true
```

An oversubscribed node also supports the oversubscription thresholds, as listed in **Table 6-26**. For example:

```
kubectl annotate node 192.168.0.0 volcano.sh/evicting-cpu-high-watermark=70
```

Querying the node information

```
# kubectl describe node 192.168.0.0
Name:              192.168.0.0
Roles:             <none>
Labels:            ...
                   volcano.sh/oversubscription=true
Annotations:       ...
                   volcano.sh/evicting-cpu-high-watermark: 70
```

Table 6-26 Node oversubscription annotations

Parameter	Description
volcano.sh/evicting-cpu-high-watermark	Upper limit for CPU usage. When the CPU usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is **80**, indicating that offline job eviction is triggered when the CPU usage of a node exceeds 80%.
volcano.sh/evicting-cpu-low-watermark	Lower limit for CPU usage. When the CPU usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the CPU usage of the node is lower than the lower limit. The default value is **30**, indicating that offline jobs are accepted again when the CPU usage of a node is lower than 30%.
volcano.sh/evicting-memory-high-watermark	Upper limit for memory usage. When the memory usage of a node exceeds the specified value, offline job eviction is triggered and the node becomes unschedulable. The default value is **60**, indicating that offline job eviction is triggered when the memory usage of a node exceeds 60%.

Parameter	Description
volcano.sh/evicting-memory-low-watermark	Lower limit for memory usage. When the memory usage of a node is higher than the upper limit, offline jobs will be evicted. The node accepts the offline jobs again only when the memory usage of the node is lower than the lower limit. The default value is **30**, indicating that offline jobs are accepted again when the memory usage of a node is lower than 30%.
volcano.sh/oversubscription-types	Oversubscribed resource type. Options: ● **cpu**: oversubscribed CPU ● **memory**: oversubscribed memory ● **cpu,memory**: oversubscribed CPU and memory The default value is **cpu,memory**.

Step 5 Create resources at a high- and low-priorityClass, respectively.

```
cat <<EOF | kubectl apply -f -

apiVersion: scheduling.k8s.io/v1
description: Used for high priority pods
kind: PriorityClass
metadata:
  name: volcano-production
preemptionPolicy: PreemptLowerPriority
value: 999999
---
apiVersion: scheduling.k8s.io/v1
description: Used for low priority pods
kind: PriorityClass
metadata:
  name: volcano-free
preemptionPolicy: PreemptLowerPriority
value: -90000

EOF
```

Step 6 Deploy online and offline jobs and configure priorityClasses for these jobs.

The **volcano.sh/qos-level** annotation needs to be added to distinguish offline jobs. The value is an integer ranging from -7 to 7. If the value is less than 0, the job is an offline job. If the value is greater than or equal to 0, the job is an online job. You do not need to set this annotation for online jobs. For both online and offline jobs, set **schedulerName** to **volcano** to enable Volcano.

NOTE

The priorities between online jobs and between offline jobs are not differentiated, and the value validity is not verified. If the value of **volcano.sh/qos-level** of an offline job is not a negative integer ranging from -7 to 0, the job is processed as an online job.

For an offline job:

```
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
```



```
  metadata:
    annotations:
      metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
      volcano.sh/qos-level: "-1"        # Offline job annotation
  spec:
    schedulerName: volcano              # Volcano is used.
    priorityClassName: volcano-free         # volcano-free priorityClass
    ...
```

For an online job:

```
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 4
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
    spec:
      schedulerName: volcano          # Volcano is used.
      priorityClassName: volcano-production   # volcano-production priorityClass
      ...
```

Step 7 Run the following command to check the number of oversubscribed resources and the resource usage:

kubectl describe node *<nodeIP>*

```
# kubectl describe node 192.168.0.0
Name:              192.168.0.0
Roles:             <none>
Labels:            ...
                   volcano.sh/oversubscription=true
Annotations:       ...
                   volcano.sh/oversubscription-cpu: 2335
                   volcano.sh/oversubscription-memory: 341753856
Allocatable:
  cpu:             3920m
  memory:          6263988Ki
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource         Requests      Limits
  --------         --------      ------
  cpu              4950m (126%)  4950m (126%)
  memory           1712Mi (27%)  1712Mi (27%)
```

In the preceding command, CPU and memory are in the unit of m CPU cores and MiB, respectively.

----End

Deployment Example

The following uses an example to describe how to deploy online and offline jobs in hybrid mode.

Step 1 Configure a cluster with two nodes, one oversubscribed and the other non-oversubscribed.

```
# kubectl get node
NAME          STATUS   ROLES    AGE    VERSION
192.168.0.173   Ready    <none>   4h58m   v1.19.16-r2-CCE22.5.1
192.168.0.3     Ready    <none>   148m    v1.19.16-r2-CCE22.5.1
```

- 192.168.0.173 is an oversubscribed node (with the **volcano.sh/oversubscription=true** label).

- 192.168.0.3 is a non-oversubscribed node (without the **volcano.sh/oversubscription=true** label).

```
# kubectl describe node 192.168.0.173
Name:               192.168.0.173
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    ...
                    volcano.sh/oversubscription=true
```

Step 2 Submit offline job creation requests. If resources are sufficient, all offline jobs will be scheduled to the oversubscribed node.

The offline job template is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: offline
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: offline
  template:
    metadata:
      labels:
        app: offline
      annotations:
        volcano.sh/qos-level: "-1"       # Offline job label
    spec:
      schedulerName: volcano             # Volcano is used.
      priorityClassName: volcano-free         # volcano-free priorityClass
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 500m
              memory: 512Mi
            limits:
              cpu: "1"
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Offline jobs are scheduled to the oversubscribed node.

```
# kubectl get pod -o wide
NAME                       READY   STATUS    RESTARTS  AGE     IP             NODE
offline-69cdd49bf4-pmjp8   1/1     Running   0         5s    192.168.10.178   192.168.0.173
offline-69cdd49bf4-z8kxh   1/1     Running   0         5s    192.168.10.131   192.168.0.173
```

Step 3 Submit online job creation requests. If resources are sufficient, the online jobs will be scheduled to the non-oversubscribed node.

The online job template is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
```



```
    labels:
      app: online
  spec:
    schedulerName: volcano               # Volcano is used.
    priorityClassName: volcano-production         # volcano-production priorityClass
    containers:
      - name: container-1
        image: resource_consumer:latest
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: 1400m
            memory: 512Mi
          limits:
            cpu: "2"
            memory: 512Mi
    imagePullSecrets:
      - name: default-secret
```

Online jobs are scheduled to the non-oversubscribed node.

```
# kubectl get pod -o wide
NAME                READY   STATUS    RESTARTS  AGE     IP             NODE
online-ffb46f656-4mwr6  1/1     Running   0         5s    192.168.10.146   192.168.0.3
online-ffb46f656-dqdv2  1/1     Running   0         5s    192.168.10.67    192.168.0.3
```

Step 4 Improve the resource usage of the oversubscribed node and observe whether offline job eviction is triggered.

Deploy online jobs to the oversubscribed node (192.168.0.173).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: online
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
      labels:
        app: online
    spec:
      affinity:                          # Submit an online job to an oversubscribed node.
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                - 192.168.0.173
      schedulerName: volcano               # Volcano is used.
      priorityClassName: volcano-production         # volcano-production priorityClass
      containers:
        - name: container-1
          image: resource_consumer:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 700m
              memory: 512Mi
            limits:
              cpu: 700m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```


Submit the online or offline jobs to the oversubscribed node (192.168.0.173) at the same time.

```
# kubectl get pod -o wide
NAME                    READY   STATUS    RESTARTS  AGE     IP             NODE
offline-69cdd49bf4-pmjp8  1/1     Running    0      13m   192.168.10.178   192.168.0.173
offline-69cdd49bf4-z8kxh  1/1     Running    0      13m   192.168.10.131   192.168.0.173
online-6f44bb68bd-b8z9p  1/1     Running     0     3m4s   192.168.10.18   192.168.0.173
online-6f44bb68bd-g6xk8  1/1     Running     0     3m12s   192.168.10.69   192.168.0.173
```

Check the oversubscribed node with IP address 192.168.0.173. It is found that resources are oversubscribed, where there are 2343m CPU cores and 3073653200 MiB of memory. Additionally, the CPU allocation rate exceeded 100%.

```
# kubectl describe node 192.168.0.173
Name:              192.168.0.173
Roles:             <none>
Labels:              …
                   volcano.sh/oversubscription=true
Annotations:         …
                   volcano.sh/oversubscription-cpu: 2343
                   volcano.sh/oversubscription-memory: 3073653200
                   …
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource             Requests      Limits
  --------             --------      ------
  cpu                  4750m (121%)  7350m (187%)
  memory                 3760Mi (61%)  4660Mi (76%)
                     …
```

Increase the CPU usage of online jobs on the node. Offline job eviction is triggered.

```
# kubectl get pod -o wide
NAME                    READY   STATUS    RESTARTS  AGE     IP             NODE
offline-69cdd49bf4-bwdm7  1/1     Running   0       11m   192.168.10.208  192.168.0.3
offline-69cdd49bf4-pmjp8   0/1    Evicted    0      26m   <none>         192.168.0.173
offline-69cdd49bf4-qpdss   1/1    Running   0       11m   192.168.10.174  192.168.0.3
offline-69cdd49bf4-z8kxh   0/1    Evicted    0      26m   <none>         192.168.0.173
online-6f44bb68bd-b8z9p   1/1     Running   0       24m   192.168.10.18   192.168.0.173
online-6f44bb68bd-g6xk8   1/1     Running   0       24m   192.168.10.69   192.168.0.173
```

----End

Handling Suggestions

- After kubelet of the oversubscribed node is restarted, the resource view of Volcano Scheduler is not synchronized with that of kubelet. As a result, OutOfCPU occurs in some newly scheduled jobs, which is normal. After a period of time, Volcano Scheduler can properly schedule online and offline jobs.
- After online and offline jobs are submitted, you are not advised to dynamically change the job type (adding or deleting annotation volcano.sh/qos-level: "-1") because the current kernel does not support the change of an offline job to an online job.
- CCE collects the resource usage (CPU/memory) of all pods running on a node based on the status information in the cgroups system. The resource usage may be different from the monitored resource usage, for example, the resource statistics displayed by running the **top** command.
- You can add oversubscribed resources (such as CPU and memory) at any time.

  You can reduce the oversubscribed resource types only when the resource allocation rate does not exceed 100%.

- If an offline job is deployed on a node ahead of an online job and the online job cannot be scheduled due to insufficient resources, configure a higher priorityClass for the online job than that for the offline job.
- If there are only online jobs on a node and the eviction threshold is reached, the offline jobs that are scheduled to the current node will be evicted soon. This is normal.

Switching kubelet Oversubscription to Resource Oversubscription in Cloud Native Hybrid Deployment

If the cluster meets the migration requirements described in **Table 6-23**, perform the following operations to migrate oversubscription:

1. Enabling resource oversubscription in a cloud-native hybrid deployment node pool will automatically disable kubelet oversubscription if it was previously enabled, as the two functions conflict with each other.
2. kubelet oversubscription will be automatically switched to resource oversubscription in cloud native hybrid deployment. During the switchover, kubelet temporarily cancels resource oversubscription reported through node annotations and evicts offline pods until the node's resource allocation rate is less than 100%. This is a normal case. After volcano-agent takes over resource oversubscription, this function will automatically restore.

6.6.4 Resource Oversubscription Based on Pod Profiling

Volcano now supports the oversubscription algorithm based on pod profiling. The algorithm monitors the CPU and memory usage of pods on a node to estimate the node's overall resource usage. This estimation is based on statistical analysis of pod resource usage and is done with a certain level of confidence. To prevent frequent pod evictions caused by service fluctuations due to insufficient resources, the algorithm also takes into account the thresholds and fluctuations in node resource usage and calculates a stable oversubscription amount.

The pod profiling-based algorithm is superior to the real-time CPU and memory usage algorithm by preventing significant oversubscription fluctuations and insufficient coverage of resource bursts. This ensures stable service performance while still allowing for resource oversubscription.

How It Works

Resource oversubscription based on pod profiling is implemented by both the Volcano agent and scheduler. After this function is enabled, the Volcano agent periodically collects the CPU and memory usage data of pods on a node. It then calculates the average, peak, and standard deviation of each pod's CPU and memory usage, and evaluates the node's CPU and memory usage based on the statistical characteristics of the pods.

Formula for calculating oversubscribed resources: **Oversubscribed node resources = (Allocated node resources - Evaluated node resource usage) x Oversubscription ratio**

The oversubscription volume is periodically updated to the node's annotation so that the Volcano scheduler can schedule pods based on the oversubscription volume of each node.

Prerequisites

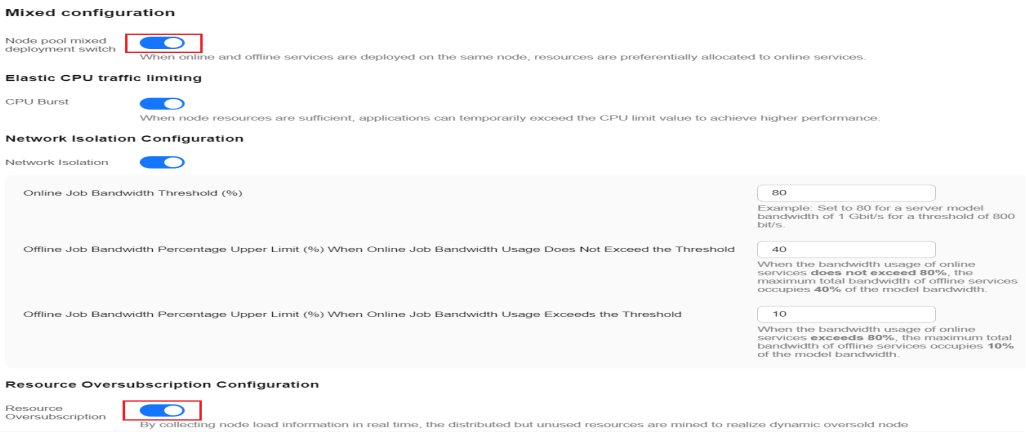
Dynamic resource oversubscription has been enabled. For details, see **Dynamic Resource Oversubscription**.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More** > **Mixed configuration**.

Ensure that node pool hybrid deployment and resource oversubscription are enabled. For details, see **Procedure**.



Step 3 In the navigation pane, choose **Settings**. On the displayed page, click the **Scheduling** tab. In the **Default cluster scheduler** area, select **Volcano scheduler** and enable the expert mode.

Step 4 On the displayed page, configure parameters.

Parameter	Description
overSubscription-Method	Method of calculating oversubscribed resources. The options are **nodeResource** and **podProfile**. **nodeResource** is the default algorithm based on node resource usage, and **podProfile** is the algorithm based on pod profiling.

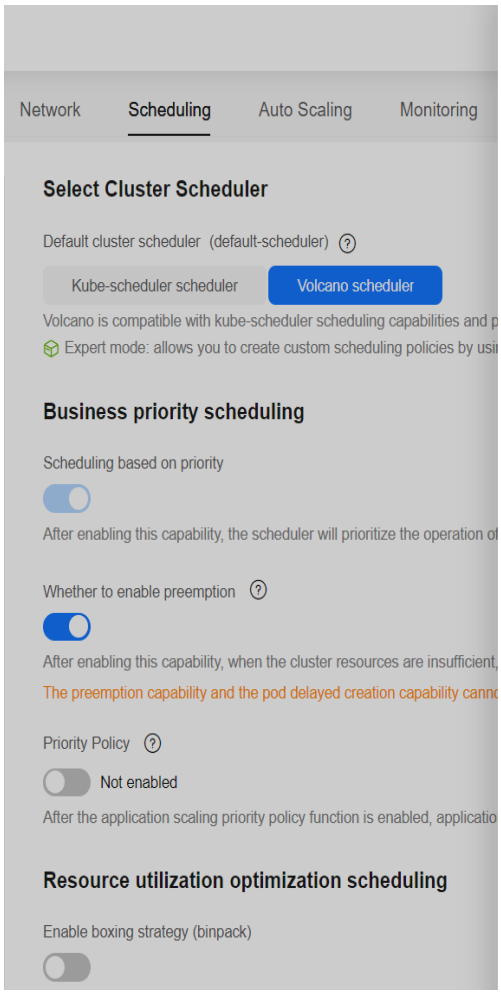
Parameter	Description
profilePeriod	Interval for pod profiling, in seconds. The value ranges from 60 (1 minute) to 2592000 (1 month). If a pod's metrics are not collected for the entire period, the node's resources will be evaluated according to the resources requested by the pod. Therefore, no node resources will be oversubscribed when the pod profiling algorithm is initially enabled until the profiling period ends.

----End

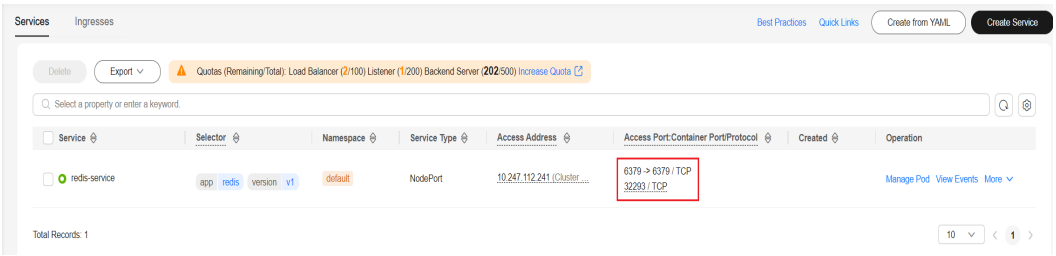
Use Cases

Pod profiling-based resource oversubscription disabled

Step 1 When configuring the expert mode, set the **oversubscription_method** value to **nodeResource**, indicating that the current cluster uses the default algorithm based on node resource usage.



Step 2 On the CCE console, create a Redis workload and associate it with a NodePort Service.



Step 3 Run the following command to increase load on the newly created Redis workload to simulate changes in service load:

```
./redis-benchmark -h <node_ip> -p 32293 -t set,get -n 3000000 -q
sleep 30
./redis-benchmark -h <node_ip> -p 32293 -t set,get -n 2000000 -q
sleep 20
./redis-benchmark -h <node_ip> -p 32293 -t set,get -n 2500000 -q
```

In the script, replace *<node_ip>* with the IP address of the node in the cluster and **32293** with the node port obtained in the previous step.

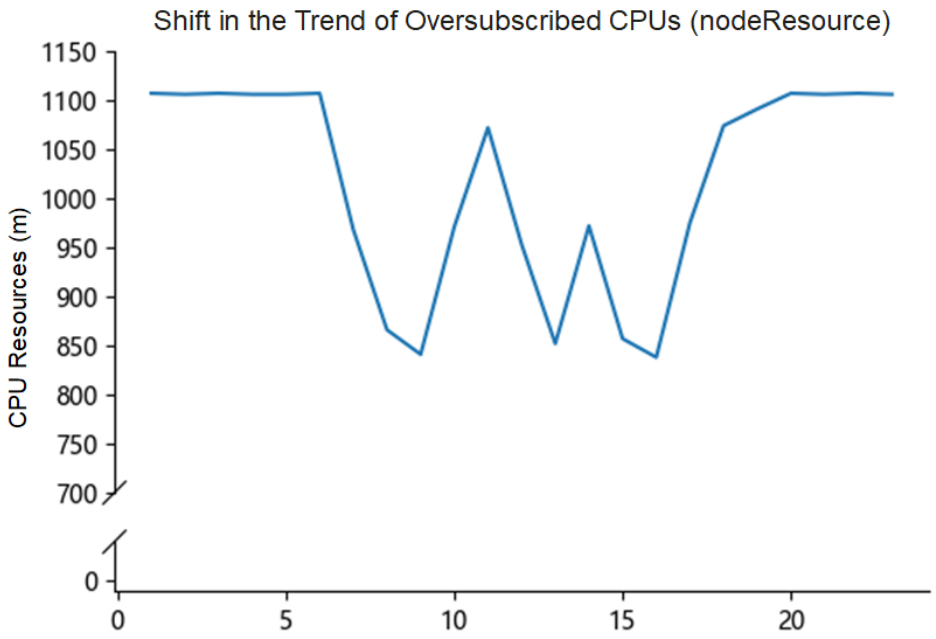
Step 4 Check the current volume of oversubscribed resources on the node and monitor any changes.

```
kubectl describe node 192.168.98.230
```

Command output:

```
Name:               192.168.98.230
Roles:              <none>
Labels:             ...
                    volcano.sh/colocation=true
                    volcano.sh/oversubscription=true
Annotations:        ...
                    volcano.sh/oversubscription-cpu: 1103
                    volcano.sh/oversubscription-memory: 1076471825
                    ...
CreationTimestamp:  Fri, 20 Sep 2024 16:12:33 +0800
...
```

Step 5 Obverse the changes in the oversubscribed resources on the node. The figure below shows the fluctuation of oversubscribed CPU resources. When the Redis workload is halted, the number of oversubscribed CPUs rises. However, it promptly drops back down once the workload resumes. If the number of oversubscribed CPUs rises and new pods are scheduled onto the node, there is a risk of CPU contention occurring when the workload restarts. This may lead to the eviction of some pods.



----End

Pod profiling-based resource oversubscription enabled

Step 1 Go to the expert mode configuration page again, set **oversubscription_method** to **podProfile**, and **oversubscription_profile_period** to **60** for a quick demonstration. However, in practice, choose a proper profiling period based on the service's characteristics to ensure that the complete resource usage period of the service is covered.

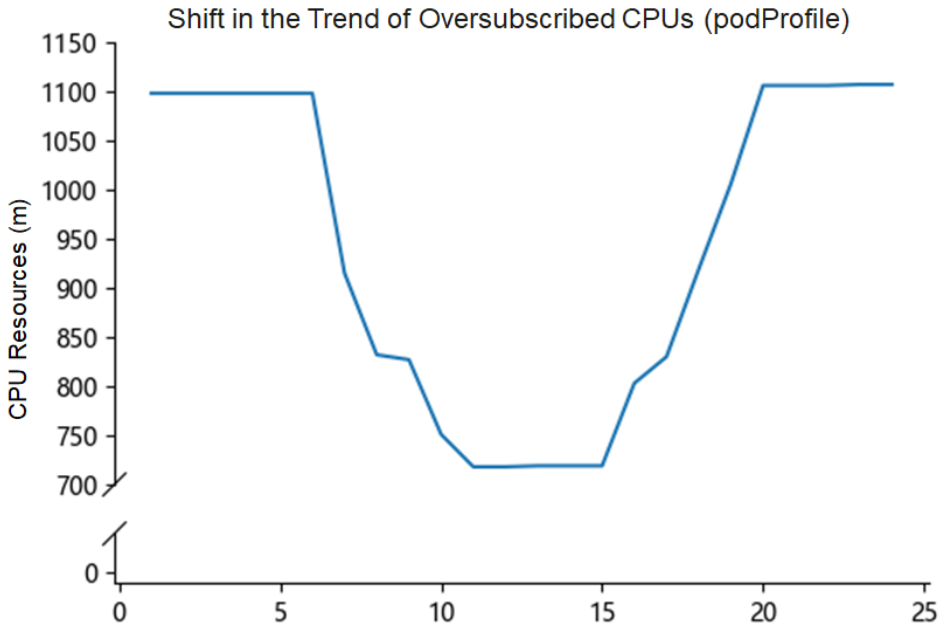
Expert Mode (Volcano Scheduling)

Download Import

Current data

```
    - name: ProfileName
      pluginConfig:
        - args:
            nodeFit: true
          name: DefaultEvictor
        - args:
            evictableNamespaces:
              exclude:
                - kube-system
            thresholds:
              cpu: 20
              memory: 20
          name: HighNodeUtilization
        - args:
            evictableNamespaces:
              exclude:
                - kube-system
            metrics:
              type: prometheus_adaptor
            nodeFit: true
            targetThresholds:
              cpu: 80
              memory: 85
            thresholds:
              cpu: 30
              memory: 30
          name: LoadAware
      plugins:
        balance:
          enabled: null
descheduler_enable: 'false'
deschedulingInterval: 10m
enable_workload_balancer: 'false'
oversubscription_method: podProfile
oversubscription_profile_period: 60
oversubscription_ratio: 60
recommendation_enable: ''
scheduler_kube_api_qps: 200
update_pod_status_qps: 50
workload_balancer_score_annotation_key: ''
workload_balancer_third_party_types: ''
```


Step 2 Wait for about 2 minutes for the Volcano agent to finish configuring the switchover and accumulating profile data. Then, run the command in **Step 3** again and observe the changes in oversubscribed CPU resources.



The figure shows that, throughout the entire script running process, the pod profiling-based algorithm takes into account the variation in pod resource usage and calculates a stable oversubscription amount. This prevents resource contention and pod eviction caused by fluctuations in pod resource usage.

----End

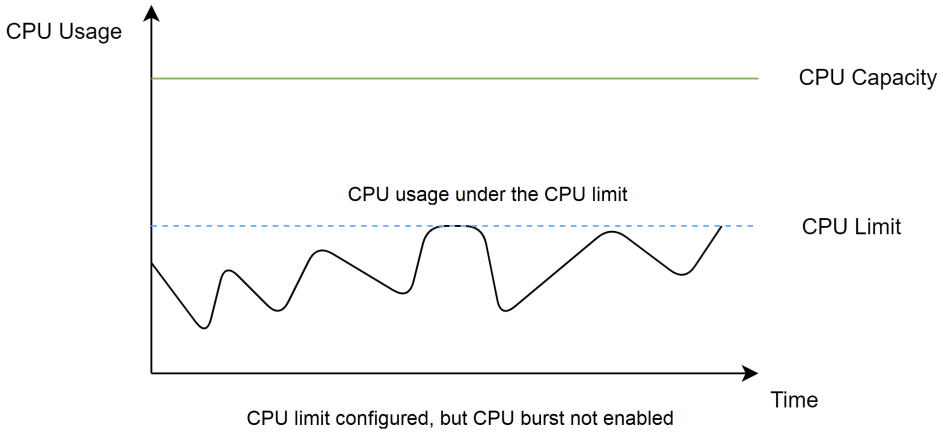
6.6.5 CPU Burst

If a CPU limit is set for the containers in a pod, the CPU usage of the containers cannot exceed the limit. Frequent CPU traffic limiting affects service performance and increases the long-tail response latency, especially for latency-sensitive services.

CPU burst is an elastic traffic limiting mechanism that allows temporarily exceeding the CPU limit to reduce the long-tail response time of services. When the CPU quota for a service in each CPU scheduling period is remaining, the system accumulates the CPU quota. If the CPU limit needs to be exceeded in subsequent scheduling periods, the accumulated CPU quota can be used.

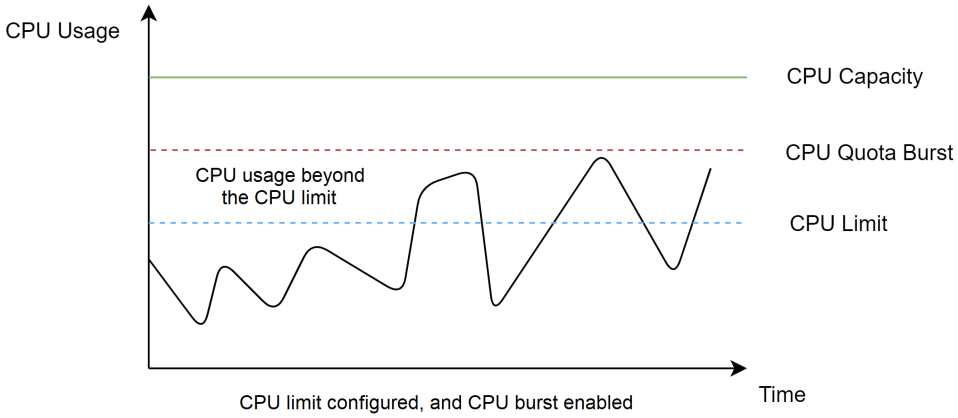
- If CPU burst is not enabled, the CPU quota for a container cannot exceed the limit, and the accumulated burst resources cannot be used.

Figure 6-26 CPU burst not enabled



- After CPU burst is enabled, the CPU quota for a container can exceed the limit to use the accumulated burst resources.

Figure 6-27 CPU burst enabled



Notes and Constraints

- Cluster version: CCE Turbo cluster v1.23.5-r0 or later
- OS version: Huawei Cloud EulerOS 2.0
- Volcano: v1.9.0 or later; hybrid deployment: enabled

Procedure

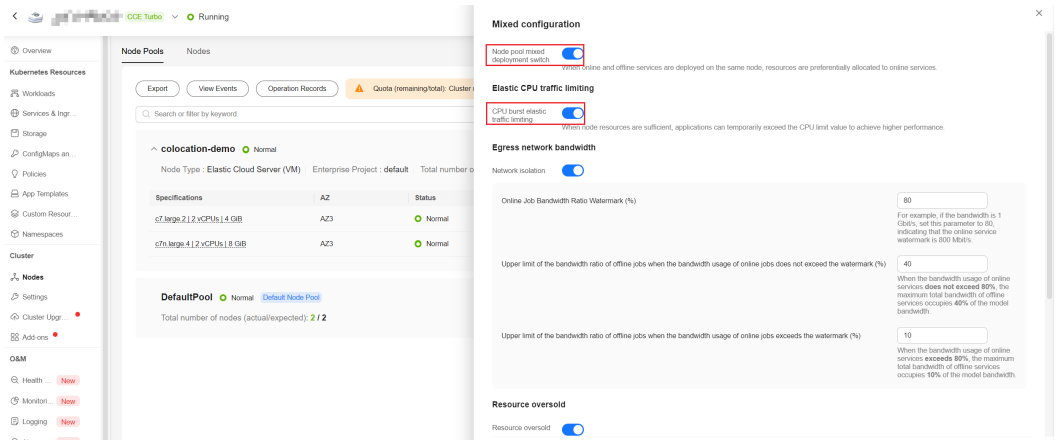
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More** > **Mixed configuration**.

Ensure that node pool hybrid deployment and traffic limit for CPU burst are enabled.

NOTE

After CPU burst is disabled, this function is still enabled on the existing pods where CPU burst has been enabled. Disabling CPU burst takes effect only on new pods.



Step 3 Deploy a workload in a node pool where hybrid deployment has been enabled. Take Nginx as an example. Set **requests** to **2** and **limits** to **4**, and create a Service that can be accessed in the cluster for the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        volcano.sh/enable-quota-burst: "true"
        volcano.sh/quota-burst-time: "200000"
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        resources:
          limits:
            cpu: "4"
          requests:
            cpu: "2"
      imagePullSecrets:
        - name: default-secret
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```


Annotation	Mandatory	Description
volcano.sh/enable-quota-burst	Yes	CPU burst is enabled for the workload.
volcano.sh/quota-burst-time	No	To ensure CPU scheduling stability and reduce contention when multiple containers encounter CPU bursts at the same time, the default **CPU Burst** value is the same as the **CPU Quota** value, indicating that a container can use a maximum of twice the **CPU Limit** value. By default, **CPU Burst** is set for all service containers in a pod. For example, if the **CPU Limit** of a container is 4, the default **CPU Burst** value is 400000 (1 core = 100000), indicating that a maximum of four additional CPU cores can be used after the value of CPU limit is reached.

Step 4 Verify CPU burst.

You can use the wrk tool to increase load of the workload and observe the service latency, traffic limiting, and CPU limit exceeding when CPU Burst is enabled and disabled, respectively.

1. Run the following command to increase load of the pod. *<service_ip>* indicates the service IP address associated with the pod.
   ```
   # Download and install the wrk tool on the node.
   # The Gzip compression module is enabled in the Apache configuration to simulate the computing logic for the server to process requests.
   # Run the following command to increase the load. Note that you need to change the IP address of the target application.
   wrk -H "Accept-Encoding: deflate, gzip" -t 4 -c 28 -d 120  --latency --timeout 2s http://<service_ip>
   ```
2. Obtain the pod ID.
   ```
   kubectl get pod -n <namespace> <pod_name> -o jsonpath='{.metadata.uid}'
   ```
3. You can run the following commands on the node to view the traffic limiting status and CPU limit exceeding status. In the command, *<pod_id>* indicates the pod ID.
   ```
   cat /sys/fs/cgroup/cpu/kubepods/burstable/pod<pod_id>/cpu.stat
   ```
   Information similar to the following is displayed:
   ```
   nr_periods 0  # Number of scheduling periods
   nr_throttled 0  # Traffic limiting times
   throttled_time 0  # Traffic limiting duration (ns)
   nr_bursts 0 # CPU Limit exceeding times
   burst_time 0 # Total Limit exceeding duration
   ```


Table 6-27 Result summary in this example

CPU Burst	P99 latency	nr_throttled Traffic Limiting Times	throttled_time Traffic Limiting Duration	nr_bursts Limit Exceeding Times	bursts_time Total Limit Exceeding Duration
Not enabled	2.96 ms	986	14.3s	0	0
Enabled	456 µs	0	0	469	3.7s

----End

6.6.6 Guaranteed Egress Network Bandwidth

An egress network bandwidth is guaranteed by configuring network priorities for the following scenarios:

- The egress network bandwidth used by online and offline services is balanced to ensure sufficient network bandwidth for online services. When the threshold is reached for online services, the bandwidth usage of offline services will be reduced.
- When online services occupy a small number of network resources, offline services can use more bandwidth. When online services occupy a large number of network resources, the resource usage of offline services will be reduced to ensure that more network bandwidth prioritizes online services.

Notes and Constraints

To use a guaranteed egress network bandwidth, ensure the following requirements are met:

- Only nodes running Huawei Cloud EulerOS 2.0 are supported.
- Only CCE Turbo clusters of v1.23 or later are supported.
- Volcano of v1.9.0 or later must be installed in the cluster, and hybrid deployment must be enabled by setting **colocation_enable** in the **expert mode** to **true**.
- Before enabling, modifying, or disabling a guaranteed egress network bandwidth, ensure Volcano is working properly.
- For pods that have been running on the node before Volcano is installed, manually restart the pods after enabling guaranteed network bandwidth so that the feature can take effect.
- Uninstalling Volcano or disabling hybrid deployment does not affect the guaranteed egress network bandwidth settings already existing on the node. To disable the guaranteed egress network bandwidth, disable network isolation.
- If bandwidth limit is enabled, the protocol stack cache may be stacked. For protocols without backpressure mechanisms, such as UDP, packet loss and ENOBUFS may occur.

- Bandwidth limit increases the risk that offline services cannot obtain bandwidth. Services may even be abnormal due to insufficient bandwidth or pod health check may fail.
- Egress network bandwidth guarantee is not prioritized in the following scenarios:
  - When **network bandwidth limit** is used for hybrid online or offline pods, the priority of network bandwidth limit is higher than that of the current function.
  - When a pod uses the node network (**hostNetwork**), the egress network bandwidth guarantee function does not take effect.

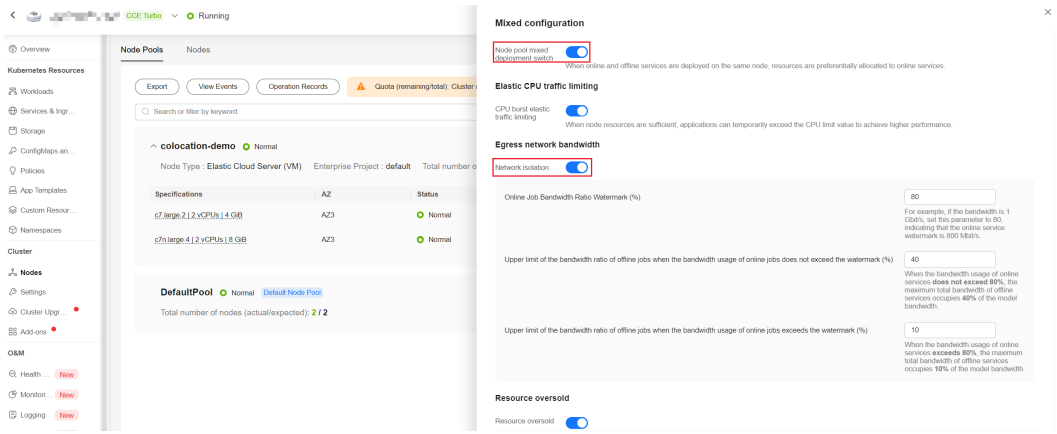
Procedure

The following describes how to enable or disable egress network bandwidth guarantee.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab page, locate the target node pool and choose **More** > **Mixed configuration**.

Ensure that node pool hybrid deployment and network isolation are enabled.



Step 3 (Optional) Modify parameters for egress network bandwidth guarantee.

NOTE

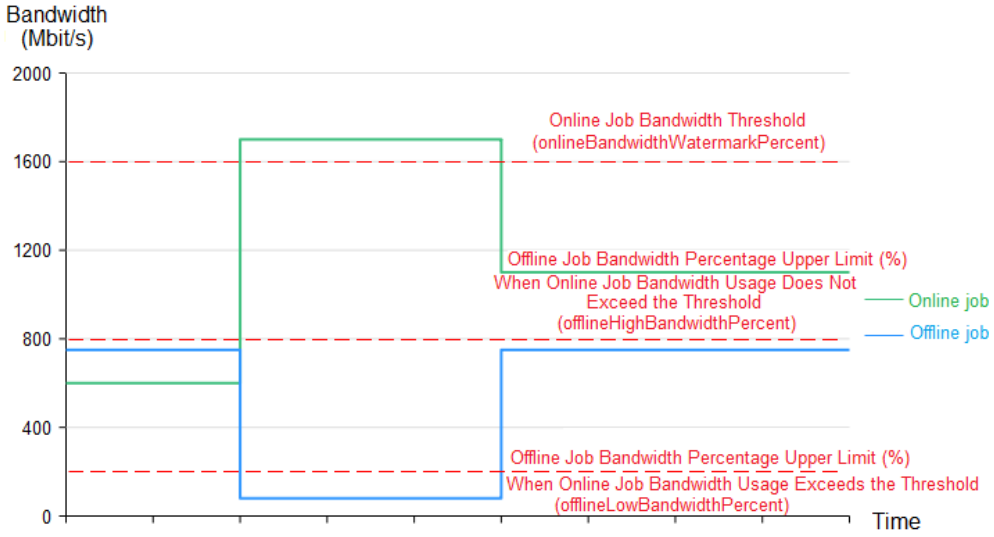
The modified parameters take effect for all nodes running Huawei Cloud EulerOS 2.0 in the cluster.

Table 6-28 Network isolation parameters

Parameter	Parameter	Description	Default Value	Configuration Range
Online Job Bandwidth Threshold (%)	onlineBandwidthWatermarkPercent	Ratio of the total bandwidth threshold of online services to the assured bandwidth of the node type *Total bandwidth threshold of online services = Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100*	80	Value range: 1 to 1000 **NOTE** The actual network bandwidth may be larger than the assured bandwidth but less than the maximum bandwidth. Therefore, the value can be greater than 100.
Offline Job Bandwidth Percentage Upper Limit (%) When Online Job Bandwidth Usage Does Not Exceed the Threshold	offlineHighBandwidthPercent	Ratio of the maximum total bandwidth usage of offline services to the assured bandwidth of the node type when the bandwidth usage of online services does not exceed the threshold. If the total bandwidth usage of online services on the same node does not exceed the value of *Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100*, the total bandwidth usage of offline services on the same node cannot exceed the value of *Assured bandwidth of the node type x offlineHighBandwidthPercent/100*.	40	

Parameter	Parameter	Description	Default Value	Configuration Range
Offline Job Bandwidth Percentage Upper Limit (%) When Online Job Bandwidth Usage Exceeds the Threshold	offlineLowBandwidthPercent	Ratio of the maximum total bandwidth usage of offline services to the assured bandwidth of the node type when the bandwidth usage of online services exceeds the threshold. If the total bandwidth usage of online services on the same node exceeds the value of *Assured bandwidth of the node type x onlineBandwidthWatermarkPercent/100*, the total bandwidth usage of offline services on the same node cannot exceed the value of *Assured bandwidth of the node type x offlineLowBandwidthPercent/100*.	10	

Figure 6-28 Example of egress network bandwidth guarantee



In the preceding figure, when the bandwidth of the online job is lower than the bandwidth baseline, the bandwidth threshold of the offline job is relatively high, indicating that the offline job can use certain bandwidth. When the bandwidth of the online job exceeds the bandwidth baseline, the bandwidth threshold of the

offline job will be lowered accordingly to reduce the bandwidth used by the offline job so that a higher bandwidth can be reserved for the online job.

----End

7 Network

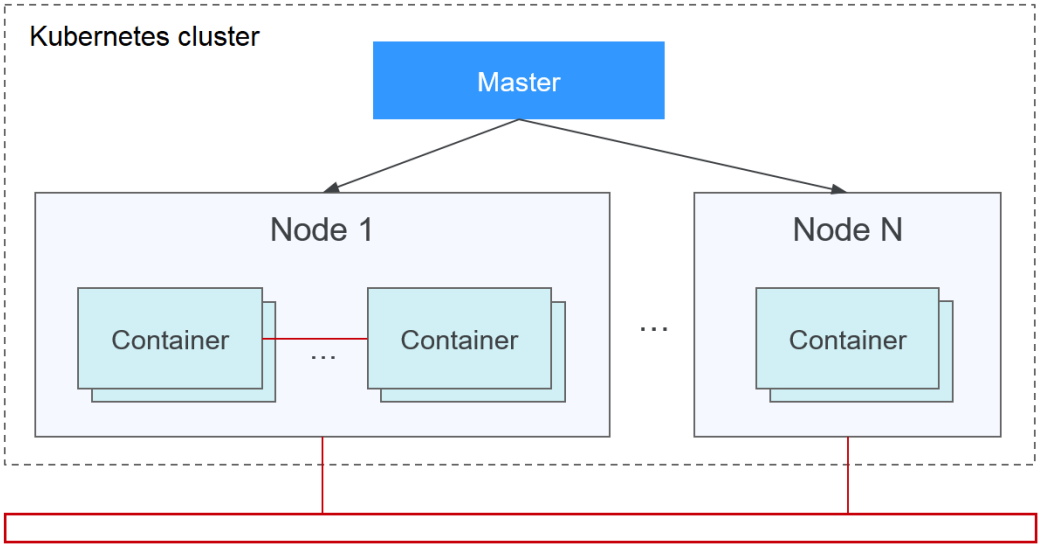
7.1 Overview

You can learn about a cluster network from the following two aspects:

- What is a cluster network like? A cluster consists of multiple nodes, and pods (or containers) are running on the nodes. Nodes and containers need to communicate with each other. For details about the cluster network types and their functions, see **Cluster Network Structure**.
- How is pod access implemented in a cluster? Accessing a pod or container is a process of accessing services of a user. Kubernetes provides **Service** and **Ingress** to address pod access issues. This section summarizes common network access scenarios. You can select the proper scenario based on site requirements. For details about the network access scenarios, see **Access Scenarios**.

Cluster Network Structure

All nodes in the cluster are located in a VPC and use the VPC network. The container network is managed by dedicated network add-ons.



- **Node Network**

  A node network assigns IP addresses to hosts (nodes in the figure above) in a cluster. Select a VPC subnet as the node network of the CCE cluster. The number of available IP addresses in a subnet determines the maximum number of nodes (including master nodes and worker nodes) that can be created in a cluster. This quantity is also affected by the container network. For details, see the container network model.

- **Container Network**

  A container network assigns IP addresses to pods in a cluster. CCE inherits the IP-Per-Pod-Per-Network network model of Kubernetes. That is, each pod has an independent IP address on a network plane and all containers in a pod share the same network namespace. All pods in a cluster exist in a directly connected flat network. They can access each other through their IP addresses without using NAT. Kubernetes only provides a network mechanism for pods, but does not directly configure pod networks. The configuration of pod networks is implemented by specific container network add-ons. The container network add-ons are responsible for configuring networks for pods and managing container IP addresses.

  Currently, CCE supports the following container network models:

  – Container tunnel network: The container tunnel network is constructed on but independent of the node network through tunnel encapsulation. This network model uses VXLAN to encapsulate Ethernet packets into UDP packets and transmits them in tunnels. Open vSwitch serves as the backend virtual switch.

  – VPC network: The VPC network model seamlessly combines VPC routing with the underlying network, making it ideal for high-performance scenarios. However, the maximum number of nodes allowed in a cluster is determined by the VPC route quota. Each node is assigned a CIDR block of a fixed size. The VPC network model outperforms the container tunnel network model in terms of performance because it does not have tunnel encapsulation overhead. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in a cluster can be directly accessed from outside the cluster.

  – Developed by CCE, Cloud Native Network 2.0 deeply integrates Elastic Network Interfaces (ENIs) and Sub Network Interfaces (sub-ENIs) of VPC. Container IP addresses are allocated from the VPC CIDR block. ELB passthrough networking is supported to direct access requests to containers. Security groups and EIPs are bound to deliver high performance.

  The performance, networking scale, and application scenarios of a container network vary according to the container network model. For details about the functions and features of different container network models, see **Overview**.

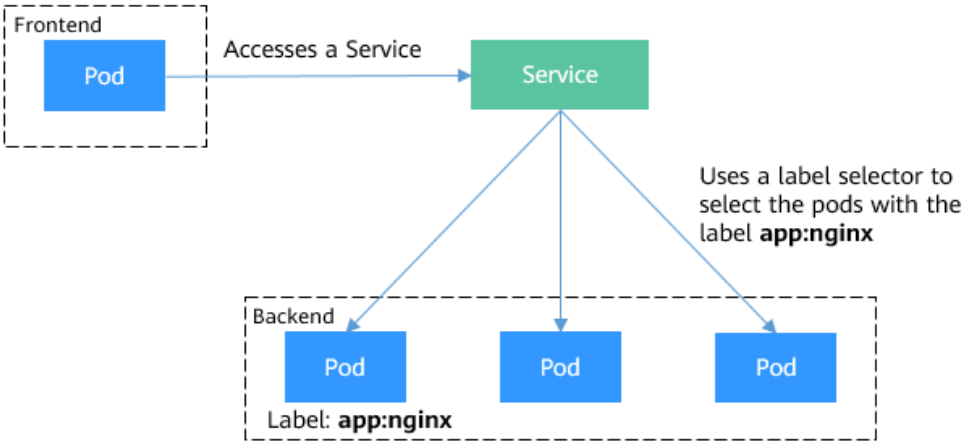
- **Service Network**

  Service is also a Kubernetes object. Each Service has a static IP address. When creating a cluster on CCE, you can specify the Service CIDR block. The Service CIDR block cannot overlap with the node or container CIDR block. The Service CIDR block can be used only within a cluster.

Service

A Service is used for pod access. With a static IP address, a Service forwards access traffic to pods and performs load balancing for these pods.

Figure 7-1 Accessing pods through a Service



You can configure the following types of Services:

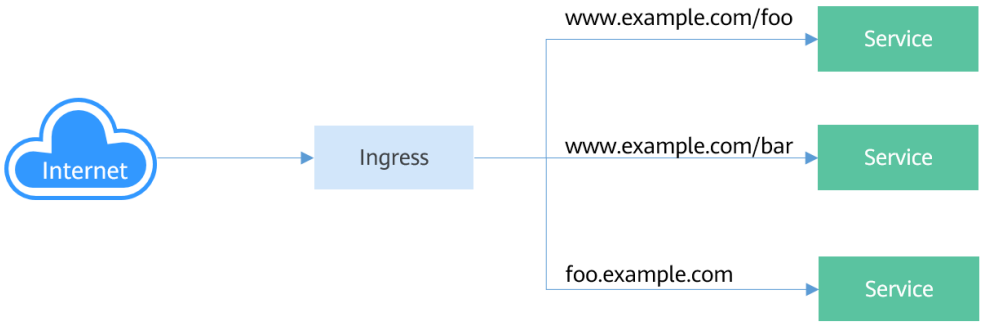
- ClusterIP: used to make the Service only reachable from within a cluster.
- NodePort: used for access from outside a cluster. A NodePort Service is accessed through the port on the node.
- LoadBalancer: used for access from outside a cluster. It is an extension of NodePort, to which a load balancer routes, and external systems only need to access the load balancer.
- DNAT: used for access from outside a cluster. It translates addresses for cluster nodes and allows multiple cluster nodes to share an EIP.

For details about the Service, see **Overview**.

Ingress

Services forward requests using layer-4 TCP and UDP protocols. Ingresses forward requests using layer-7 HTTP and HTTPS protocols. Domain names and paths can be used to achieve finer granularities.

Figure 7-2 Ingress and Service



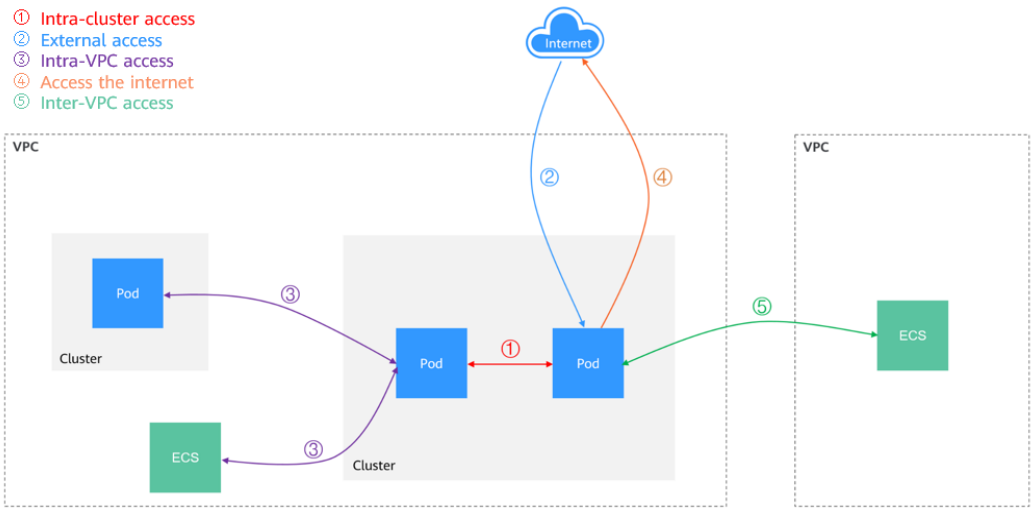
For details about the ingress, see **Overview**.

Access Scenarios

Workload access scenarios can be categorized as follows:

- Intra-cluster access: A ClusterIP Service is used for workloads in the same cluster to access each other.
- Access from outside a cluster: A Service (NodePort or LoadBalancer type) or an ingress is recommended for a workload outside a cluster to access workloads in the cluster.
  - Access through the public network: An EIP should be bound to the node or load balancer.
  - Access through the private network: The workload can be accessed through the internal IP address of the node or load balancer. If workloads are located in different VPCs, a peering connection is required to enable communication between different VPCs.
- The workload can access the external network as follows:
  - Accessing an intranet: The workload accesses the intranet address, but the implementation method varies depending on container network models. Ensure that the peer security group allows the access requests from the container CIDR block. For details, see **Configuring Intra-VPC Access**.
  - Accessing a public network: Assign an EIP to the node where the workload runs (when a VPC network or tunnel network is used), bind an EIP to the pod IP address (when Cloud Native Network 2.0 is used), or configure SNAT rules through the NAT gateway. For details, see **Accessing the Internet from a Container**.

Figure 7-3 Network access diagram



7.2 Container Network

7.2.1 Overview

The container network assigns IP addresses to pods in a cluster and provides networking services. In CCE, you can select the following network models for your cluster:

- **Cloud Native Network 2.0**
- **VPC network**
- **Tunnel network**

Network Model Comparison

Table 7-1 describes the differences of network models supported by CCE.

CAUTION

After a cluster is created, the network model cannot be changed.

Table 7-1 Network model comparison

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Application scenarios	• Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high performance requirements, such as web applications, and middle-end and back-end services with a small number of access requests. • Large-scale networking: Different from the VPC network that is limited by the VPC route quota, the container tunnel network does not have any restriction	• High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, such as AI computing and big data computing. • Small- and medium-scale networks: Due to the limitation on VPC route tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000.	• High performance requirements: Cloud Native Network 2.0 uses VPC networks to construct container networks, eliminating the need for tunnel encapsulation or NAT when containers communicate. This makes Cloud Native Network 2.0 ideal for scenarios that demand high bandwidth and low latency, such as live streaming and e-commerce flash sales. • Large-scale networking: Cloud Native Network 2.0 supports up to 2,000 ECS nodes and 100,000 pods.

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
	on the infrastructure. In addition, the container tunnel network controls the broadcast domain to the node level. The container tunnel network supports a maximum of 2000 nodes.		
Core technology	OVS	IPvlan and VPC route	VPC ENI/sub-ENI
Applicable clusters	CCE standard cluster	CCE standard cluster	CCE Turbo cluster
Container network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.
Interconnecting pods to a load balancer	Interconnected through a NodePort	Interconnected through a NodePort	Directly interconnected using a dedicated load balancer Interconnected using a shared load balancer through a NodePort
Managing container IP addresses	• Separate container CIDR blocks needed • Container CIDR blocks divided by node and dynamically added after being allocated	• Separate container CIDR blocks needed • Container CIDR blocks divided by node and statically allocated (the allocated CIDR blocks cannot be changed after a node is created)	Container CIDR blocks divided from a VPC subnet (You do not need to configure separate container CIDR blocks.)

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Network performance	Performance loss due to VXLAN encapsulation	No tunnel encapsulation, and cross-node traffic forwarded through VPC routers (The performance is so good that is comparable to that of the host network, but there is a loss caused by NAT.)	Container network integrated with VPC network, eliminating performance loss
Networking scale	A maximum of 2000 nodes are supported.	Suitable for small- and medium-scale networks due to the limitation on VPC route tables. It is recommended that the number of nodes be less than or equal to 1000. Each time a node is added to the cluster, a route is added to the VPC route tables (including the default and custom ones). Evaluate the cluster scale that is limited by the VPC route tables before creating the cluster. For details about route tables, see **Constraints**.	A maximum of 2000 nodes are supported. In a cloud-native network 2.0 cluster, containers' IP addresses are assigned from VPC CIDR blocks, and the number of containers supported is restricted by these blocks. Evaluate the cluster's scale limitations before creating it.

7.2.2 Cloud Native Network 2.0 Settings

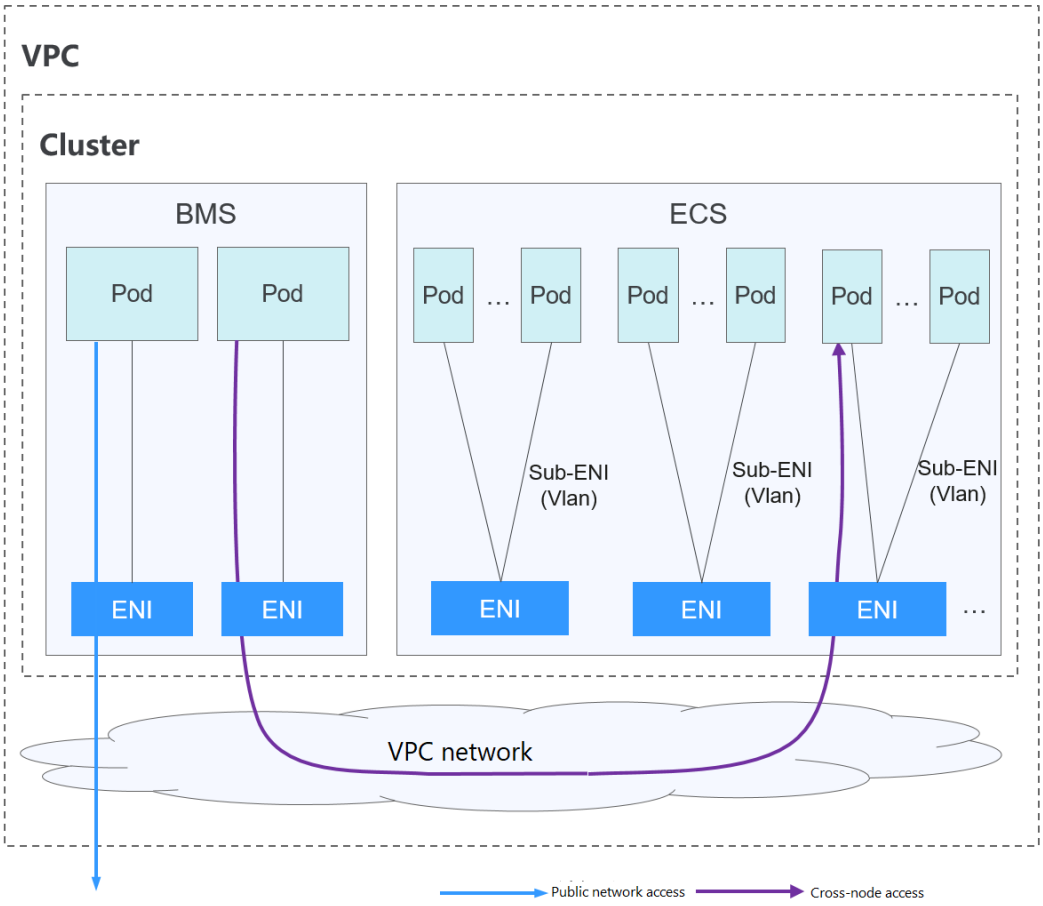
7.2.2.1 Cloud Native Network 2.0

Model Definition

Cloud Native Network 2.0 is a proprietary, next-generation container network model that combines the elastic network interfaces (ENIs) and supplementary network interfaces (sub-ENIs) of the Virtual Private Cloud (VPC). This allows ENIs or sub-ENIs to be directly bound to pods, giving each pod its own unique IP address within the VPC. Furthermore, it supports additional features like ELB

passthrough container, pod binding to a security group, and pod binding to an EIP. Because container tunnel encapsulation and NAT are not required, Cloud Native Network 2.0 enables higher network performance than the container tunnel network model and VPC network model.

Figure 7-4 Cloud Native Network 2.0



In a cluster using Cloud Native Network 2.0, pods rely on ENIs or sub-ENIs to connect to external networks.

- Pods running on BMS nodes use ENIs.
- Pods running on ECS nodes use sub-ENIs that are bound to the ECS' ENIs through VLAN sub-interfaces.
- To run a pod, it is necessary to bind ENIs to it. The number of pods that can run on a node depends on the number of ENIs that can be bound to the node and the number of ENI ports available on the node.
- Traffic for communications between pods on a node, communications between pods on different nodes, and access to networks outside a cluster is forwarded through the ENI or sub-ENI of the VPC.

Notes and Constraints

This network model is available only to CCE Turbo clusters.

Advantages and Disadvantages

Advantages

- VPCs serve as the foundation for constructing container networks. Every pod has its own network interface and IP address, which simplifies network problem-solving and enhances performance.
- In the same VPC, ENIs are directly bound to pods in a cluster, so that resources outside the cluster can directly communicate with containers within the cluster.

  NOTE

  Similarly, if the VPC is accessible to other VPCs or data centers, resources in other VPCs or data centers can directly communicate with containers in the cluster, provided there are no conflicts between the network CIDR blocks.
- The load balancing, security group, and EIP capabilities provided by VPC can be directly used by pods.

Disadvantages

Container networks are built on VPCs, with each pod receiving an IP address from the VPC CIDR block. As a result, it is crucial to plan the container CIDR block carefully before creating a cluster.

Application Scenarios

- High performance requirements: Cloud Native Network 2.0 uses VPC networks to construct container networks, eliminating the need for tunnel encapsulation or NAT when containers communicate. This makes Cloud Native Network 2.0 ideal for scenarios that demand high bandwidth and low latency, such as live streaming and e-commerce flash sales.
- Large-scale networking: Cloud Native Network 2.0 supports up to 2,000 ECS nodes and 100,000 pods.

Container IP Address Management

In Cloud Native Network 2.0, BMS nodes use ENIs and ECS nodes use sub-ENIs.

- The IP address of the pod is directly allocated from the VPC subnet configured for the container network. You do not need to allocate an independent small network segment to the node.
- To add an ECS node to a cluster, bind the ENI that carries the sub-ENI first. After the ENI is bound, you can bind the sub-ENI.
- Number of ENIs bound to an ECS node: **For clusters of v1.19.16-r40, v1.21.11-r0, v1.23.9-r0, v1.25.4-r0, v1.27.1-r0, and later versions, the value is 1. For clusters of earlier versions, the value is the maximum number of sub-ENIs that can be bound to the node divided by 64 (rounded up).**
- ENIs bound to an ECS node = **Number of ENIs used to bear sub-ENIs + Number of sub-ENIs currently used by pods + Number of pre-bound sub-ENIs**
- ENIs bound to a BMS node = **Number of ENIs currently used by pods + Number of pre-bound ENIs**

- When a pod is created, an available ENI is randomly allocated from the prebinding ENI pool of the node.
- When the pod is deleted, the ENI is released back to the ENI pool of the node.
- When a node is deleted, the ENIs are released back to the pool, and the sub-ENIs are deleted.

Cloud Native Network 2.0 supports **dynamic** and **threshold-based** ENI pre-binding policies. The following table lists the scenarios.

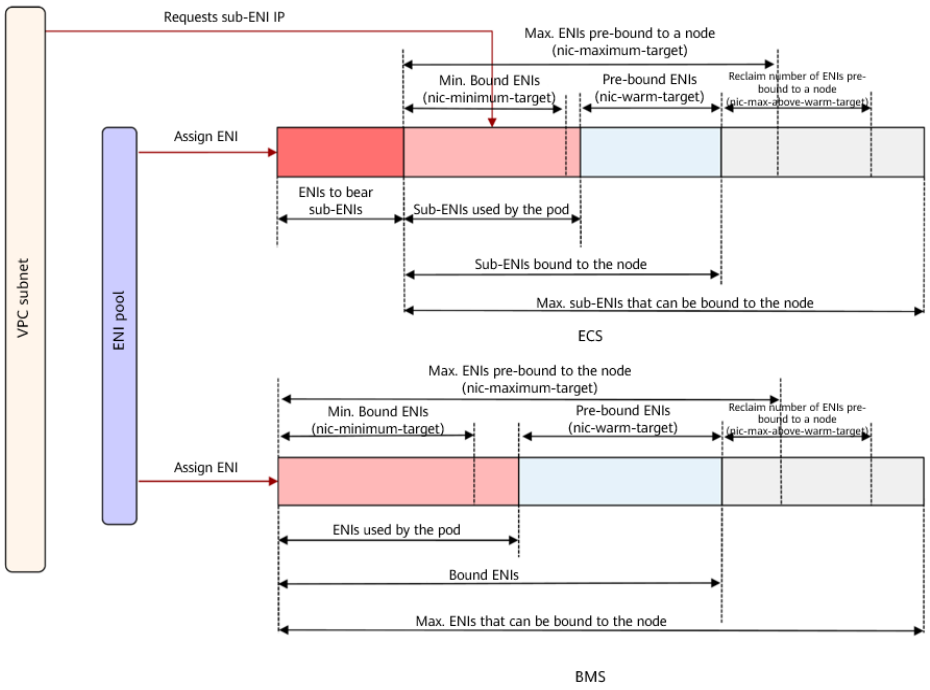
Table 7-2 Comparison between ENI pre-binding policies

Policy	Dynamic ENI Pre-binding Policy (Default)	Threshold-based ENI Pre-binding Policy (Being Deprecated)
Management policy	**nic-minimum-target**: minimum number of ENIs (pre-bound and unused + used) bound to a node. **nic-maximum-target**: If the number of ENIs bound to a node exceeds the value of this parameter, the system does not proactively pre-bind ENIs. **nic-warm-target**: minimum number of pre-bound ENIs on a node. **nic-max-above-warm-target**: ENIs are unbound and reclaimed only when the number of idle ENIs minus the number of **nic-warm-target** is greater than the threshold.	**Low threshold of the number of bound ENIs**: minimum number of ENIs (unused + used) bound to a node **High threshold of the number of bound ENIs**: maximum number of ENIs that can be bound to a node. If the number of ENIs bound to a node exceeds the value of this parameter, the system unbinds the idle ENIs.
Application scenario	Accelerates pod startup while improving IP resource utilization. This mode applies to scenarios where the number of IP addresses in the container network segment is insufficient. For details about the preceding parameters, see **Pre-binding ENIs for CCE Turbo Clusters**.	Applies to scenarios where the number of IP addresses in the container CIDR block is sufficient and the number of pods on nodes changes sharply but is fixed in a certain range.

NOTE

- For clusters from 1.19.16-r2, 1.21.5-r0, 1.23.3-r0 to 1.19.16-r4, 1.21.7-r0, and 1.23.5-r0, only the **nic-minimum-target** and **nic-warm-target** parameters are supported. The threshold-based pre-binding policy takes priority over the dynamic ENI pre-binding policy.
- For clusters of 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0 or later, the preceding parameters are supported. The dynamic ENI pre-binding policy takes priority over the threshold-based pre-binding policy.

Figure 7-5 Dynamic ENI pre-binding policy



CCE provides four parameters for the dynamic ENI pre-binding policy. Set these parameters properly.

Table 7-3 Parameters of the dynamic ENI pre-binding policy

Parameter	Default Value	Description	Suggestion
nic-minimum-target	10	Minimum number of ENIs bound to a node. The value can be a number or a percentage. • Value: The value must be a positive integer. For example, 10 indicates that at least 10 ENIs are bound to a node. If the ENI quota of a node is exceeded, the ENI quota is used. • Percentage: The value ranges from 1% to 100%. For example, 10%. If the ENI quota of a node is 128, at least 12 (rounded down) ENIs are bound to the node. Set both **nic-minimum-target** and **nic-maximum-target** to the same value or percentage.	Set these parameters based on the number of pods.

Parameter	Default Value	Description	Suggestion
nic-maximum-target	0	If the number of ENIs bound to a node exceeds the value of **nic-maximum-target**, the system does not proactively pre-bind ENIs. If the value of this parameter is greater than or equal to the value of **nic-minimum-target**, the check on the maximum number of the pre-bound ENIs is enabled. Otherwise, the check is disabled. The value can be a number or a percentage. • Value: The value must be a positive integer. For example, 0. The check on the maximum number of the pre-bound ENIs is disabled. If the ENI quota of a node is exceeded, the ENI quota is used. • Percentage: The value ranges from 1% to 100%. For example, 50%. If the ENI quota of a node is 128, the maximum number of the pre-bound ENI is 64 (rounded down). Set both **nic-minimum-target** and **nic-maximum-target** to the same value or percentage.	Set these parameters based on the number of pods.
nic-warm-target	2	Minimum number of pre-bound ENIs on a node. The value must be a number. When the value of **nic-warm-target** + the number of bound ENIs is greater than the value of **nic-maximum-target**, the system will pre-bind ENIs based on the difference between the value of **nic-maximum-target** and the number of bound ENIs.	Set this parameter to the number of pods that can be scaled out instantaneously within 10 seconds.

Parameter	Default Value	Description	Suggestion
nic-max-above-warm-target	2	Only when the number of idle ENIs on a node minus the value of **nic-warm-target** is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number. • Setting a larger value of this parameter slows down the recycling of idle ENIs and accelerates pod startup. However, the IP address usage decreases, especially when IP addresses are insufficient. Therefore, **exercise caution when increasing the value of this parameter**. • Setting a smaller value of this parameter accelerates the recycling of idle ENIs and improves the IP address usage. However, when a large number of pods increase instantaneously, the startup of some pods slows down.	Set this parameter based on the difference between the number of pods that are frequently scaled on most nodes within minutes and the number of pods that are instantly scaled out on most nodes within 10 seconds.

NOTE

The preceding parameters support global configuration at the cluster level and custom settings at the node pool level. The latter takes priority over the former.

The container networking component maintains a scalable pre-bound ENI pool for each node. The component checks and calculates the number of pre-bound ENIs or idle ENIs every 10 seconds.

- **Number of pre-bound ENIs = min(nic-maximum-target – Number of bound ENIs, max(nic-minimum-target – Number of bound ENIs, nic-warm-target – Number of idle ENIs)**
- **Number of ENIs to be unbound = min(Number of idle ENIs – nic-warm-target – nic-max-above-warm-target, Number of bound ENIs – nic-minimum-target)**

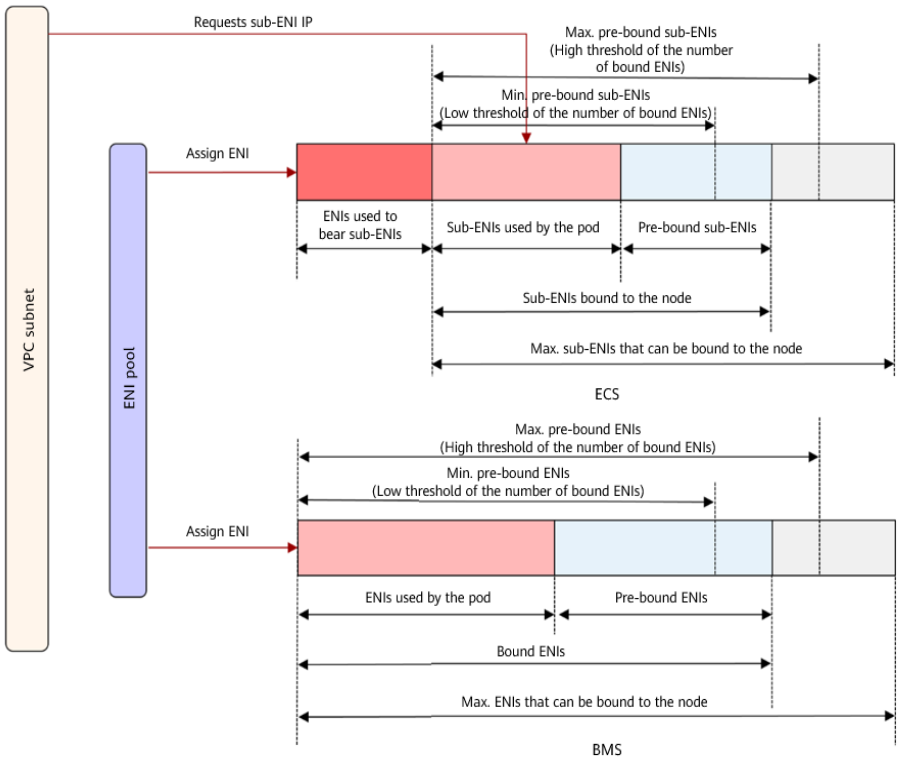
The number of pre-binding ENIs on the node remains in the following range:

- **Minimum number of ENIs to be pre-bound = min(max(nic-minimum-target – Number of bound ENIs, nic-warm-target), nic-maximum-target – Number of bound ENIs)**
- **Maximum number of ENIs to be pre-bound = max(nic-warm-target + nic-max-above-warm-target, Number of bound ENIs – nic-minimum-target)**

When a pod is created, an idle pre-bound ENI (the earliest unused one) will be preferentially allocated from the pool. If no idle ENI is available, a new ENI will be created or a new sub-ENI will be bound to the pod.

When the pod is deleted, the corresponding ENI is released back to the pre-bound ENI pool of the node, enters a 2 minutes cooldown period, and can be bind to another pod. If the ENI is not bound to any pod within 2 minutes, it will be released.

Figure 7-6 Threshold-based policy



CCE provides a configuration parameter for the threshold algorithms. You can set this parameter based on the service plan, cluster scale, and number of ENIs that can be bound to a node.

- **Low threshold of the number of bound ENIs**: Defaults to **0**, indicating the minimum number of ENIs (unused + used) bound to a node. Minimum number of pre-bound ENIs on an ECS node = Number of ENIs bound to the node at the low threshold x Number of sub-ENIs on the node. Minimum number of pre-bound ENIs on a BMS node = Number of ENIs bound to the node at the low threshold x Number of ENIs on the node.
- **High threshold of the number of bound ENIs**: Defaults to **0**, indicating the maximum number of ENIs that can be bound to a node. If the number of ENIs bound to a node exceeds the value of this parameter, the system unbinds the idle ENIs. Maximum number of pre-bound ENIs on an ECS node = Number of bound ENIs at the high threshold x Number of sub-ENIs on the node. Maximum number of pre-bound ENIs on a BMS node = Number of bound ENIs at the high threshold x Number of ENIs on the node.

The container networking component maintains a scalable ENI pool for each node.

- If **the number of bound ENIs (used ENIs + pre-bound ENIs) is less than the number of pre-bound ENIs at the low threshold**, ENIs are bound until **the two numbers are equal**.
- If **the number of bound ENIs (used ENIs + pre-bound ENIs) is greater than the number of pre-bound ENIs at the high threshold** and **the number of pre-bound ENIs is greater than 0**, the pre-bound ENIs that are not used for more than 2 minutes will be released periodically until **the number of bound ENIs = Number of pre-bound ENIs at the high threshold** or **the number of used ENIs is greater than the number of pre-bound ENIs at the high threshold** and **the number of pre-bound ENIs on the node is 0**.

Recommendation for CIDR Block Planning

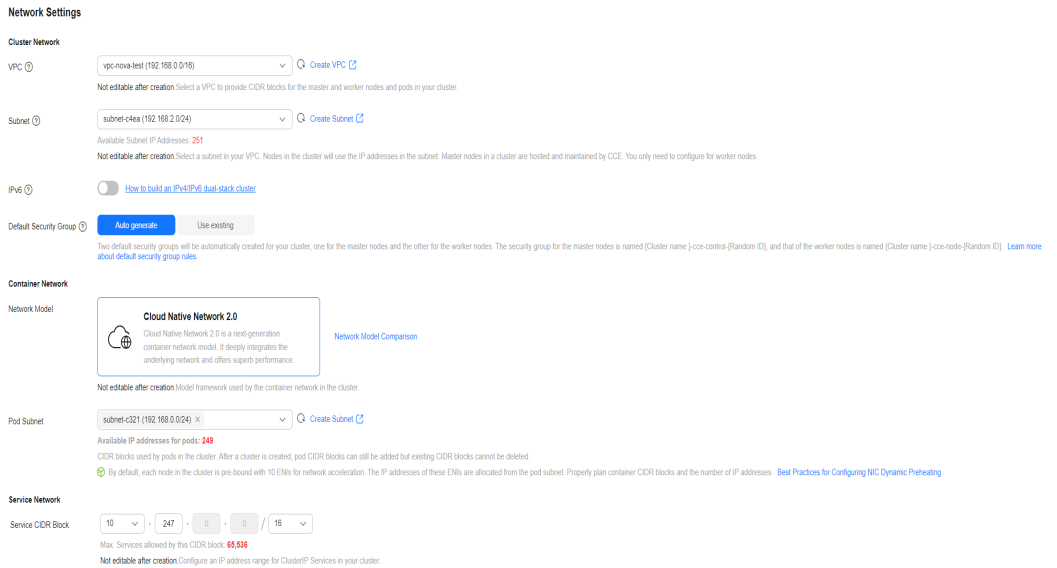
As explained in **Cluster Network Structure**, network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- All subnets (including extended subnets) in the VPC where the cluster resides cannot conflict with the Service CIDR blocks.
- Ensure that **each CIDR block has sufficient IP addresses**.
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses.

In Cloud Native Network 2.0, the container CIDR block and node CIDR block share the network IP addresses in a VPC. It is recommended that the container subnet and node subnet not use the same subnet. Otherwise, containers or nodes may fail to be created due to insufficient IP addresses.

In addition, a subnet can be added to the container CIDR block after a cluster is created to increase the number of available IP addresses. In this case, ensure that the added subnet does not conflict with other subnets in the container CIDR block.

Figure 7-7 Configuring CIDR blocks (during cluster creation)

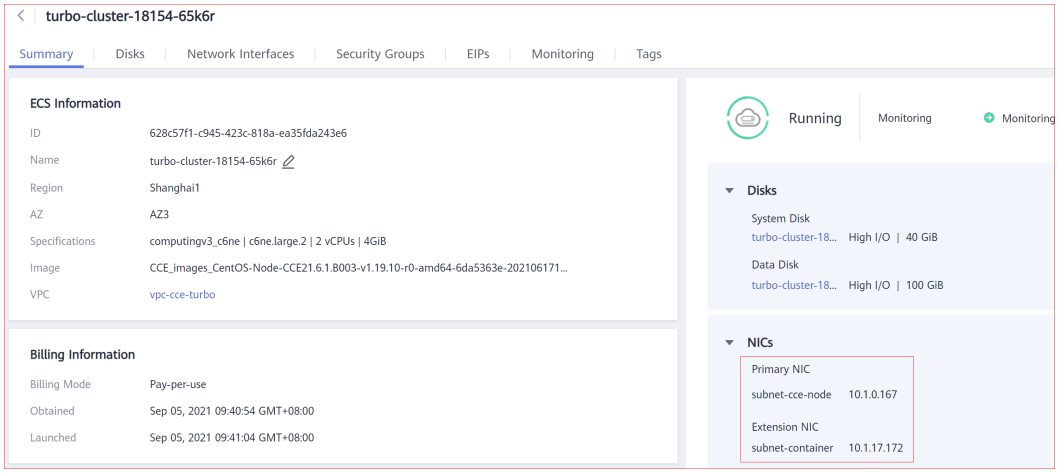


Example of Cloud Native Network 2.0 Access

In this example, a CCE Turbo cluster is created, and the cluster contains three ECS nodes.

You can check the basic information about a node on the ECS console. You can see that a primary network interface and an extended network interface are bound to the node. Both of them are ENIs. The IP address of the extended network interface belongs to the container CIDR block and is used to bind sub-ENIs to pods on the node.

Figure 7-8 Checking node ENIs



The following is an example of creating a workload in a cluster using Cloud Native Network 2.0:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a Deployment in the cluster.

Create the **deployment.yaml** file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 6
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
```



```
          cpu: 250m
          memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```

Step 3 Check the running pods.

```
kubectl get pod -owide
```

Command output:

```
NAME                       READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE
READINESS GATES
example-5bdc5699b7-54v7g   1/1     Running   0          7s    10.1.18.2    10.1.0.167   <none>           <none>
example-5bdc5699b7-6dzx5   1/1     Running   0          7s    10.1.18.216  10.1.0.186   <none>           <none>
example-5bdc5699b7-gq7xs   1/1     Running   0          7s    10.1.16.63   10.1.0.144   <none>           <none>
example-5bdc5699b7-h9rvb   1/1     Running   0          7s    10.1.16.125  10.1.0.167   <none>           <none>
example-5bdc5699b7-s9fts   1/1     Running   0          7s    10.1.16.89   10.1.0.144   <none>           <none>
example-5bdc5699b7-swq6q   1/1     Running   0          7s    10.1.17.111  10.1.0.167   <none>
<none>
```

The IP addresses of all pods are sub-ENIs, which are bound to the ENI (extended network interface) of the node.

For example, the IP address of the extended network interface of **node 10.1.0.167** is **10.1.17.172**. On the network interfaces console, you can see that there are three sub-ENIs bound to the extended network interface whose IP address is **10.1.17.172**. These sub-ENIs are the IP addresses of the pods running on the node.

Figure 7-9 Checking pod ENIs

Step 4 Log in to an ECS in the same VPC and access the IP address of a pod from outside the cluster.

In this example, the accessed pod IP address is *10.1.18.2*.

```
curl 10.1.18.2
```

If the following information is displayed, the workload can be properly accessed:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
```



```
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

Performance on Batch Creating Pods in a CCE Turbo Cluster

Pods in a CCE Turbo cluster request ENIs or sub-ENIs from VPC. Pods are bound with ENIs or sub-ENIs after pod scheduling is complete. The pod creation speed is constrained by how fast ENIs are created and bound. The following table describes the constraints.

Table 7-4 Time required for creating ENIs

Node Type	ENI Type	Maximum Number of Supported ENI	Binding ENI to Node	ENI Availability	Concurrency Control	Default Pre-Binding Configuration of ENI to Node
ECS	Sub-ENI	256	Specifying the ENI of a node to create a sub-ENI	Within 1 second	Tenant-level: 600/minute	For clusters of versions earlier than 1.19.16-r2, 1.21.5-r0, or 1.23.3-r0, ENI pre-binding is not supported. For clusters of versions between 1.19.16-r2, 1.21.5-r0, or 1.23.3-r0 and 1.19.16-r4, 1.21.7-r0, or 1.23.5-r0, dynamic ENI pre-binding is supported (**nic-minimum-target=10**; **nic-warm-target=2**). For clusters of 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0, and later, dynamic pre-binding is supported (**nic-minimum-target=10**; **nic-maximum-target=0**; **nic-warm-target=2**; **nic-max-above-warm-target=2**).

Node Type	ENI Type	Maximum Number of Supported ENI	Binding ENI to Node	ENI Availability	Concurrency Control	Default Pre-Binding Configuration of ENI to Node
BMS	ENI	128	Binding an ENI to a node	20s to 30s	Node-level: 3 concurrently	For clusters earlier than 1.19.16-r4, 1.21.7-r0, and 1.23.5-r0, the total number of ENIs is determined by the high and low thresholds (**nic-threshold=0.3:0.6**). For clusters of 1.19.16-r4, 1.21.7-r0, 1.23.5-r0, 1.25.1-r0, 1.28.1-r0, and later, dynamic pre-binding is supported (**nic-minimum-target=10**; **nic-maximum-target=0**; **nic-warm-target=2**; **nic-max-above-warm-target=2**).

NOTE

Pre-binding consumes container subnet IP addresses and affects the number of pods that can run in the cluster. You should properly plan and configure dynamic pre-binding based on the service scale. For details, see **Pre-Binding Container ENI for CCE Turbo Clusters**.

Creating a Pod on an ECS Node (Using Sub-ENIs)

- If no pre-bound ENI is available on the node to which a pod is scheduled, the API used for creating a sub-ENI is called to create a sub-ENI using the ENI of the node and allocate the sub-ENI to the pod.
- If a pre-bound ENI is available on the node to which a pod is scheduled, the unused sub-ENI that is created the earliest is allocated to the pod.
- Limited by the concurrent creation speed of sub-ENIs, a maximum of 600 pods can be created per minute without pre-binding. If a larger-scale creation is required, you can configure pre-binding for sub-ENIs as needed.

Creating a Pod on a BMS Node (Using ENIs)

- If no prebound ENI is available on the node to which a pod is scheduled, the API used for binding an ENI to a node is called to bind and allocate an ENI to the pod. It takes about 20 to 30 seconds to bind an ENI to a BMS node.
- If a pre-bound ENI is available on the node to which a pod is scheduled, the unused ENI that is created the earliest is allocated to the pod.
- Limited by the speed of binding ENIs to BMS nodes, three pods running on the same node can start in 20 seconds without pre-binding. Therefore, pre-bind all available ENIs for BMS nodes.

7.2.2.2 Configuring a Default Container Subnet for a CCE Turbo Cluster

Scenario

If the pod subnet configured during CCE Turbo cluster creation cannot meet service expansion requirements, you can add a pod subnet for the cluster.

Notes and Constraints

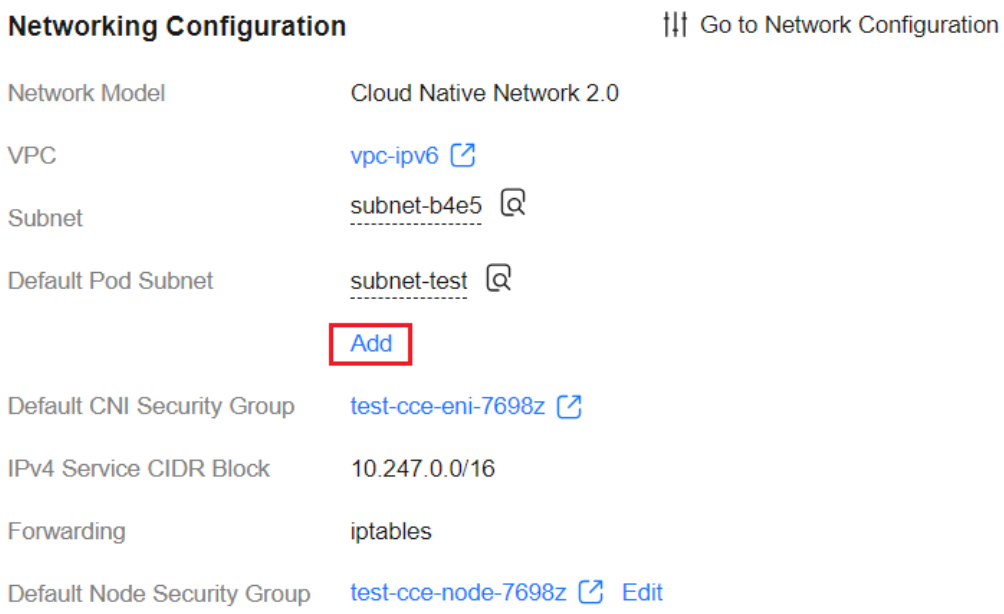
- This function is available only for CCE Turbo clusters of v1.19 or later.

Procedure for Adding a Default Subnet

Step 1 Log in to the CCE console and access the CCE Turbo cluster console.

Step 2 On the **Overview** page, locate the **Networking Configuration** area and click **Add**.

Figure 7-10 Adding a pod subnet



Step 3 Select a pod subnet in the same VPC. You can add multiple pod subnets at a time. If no other pod subnet is available, go to the VPC console and create one.

Figure 7-11 Selecting a pod subnet

Step 4 Click **OK**.

----End

Procedure for Deleting a Default Subnet

NOTE

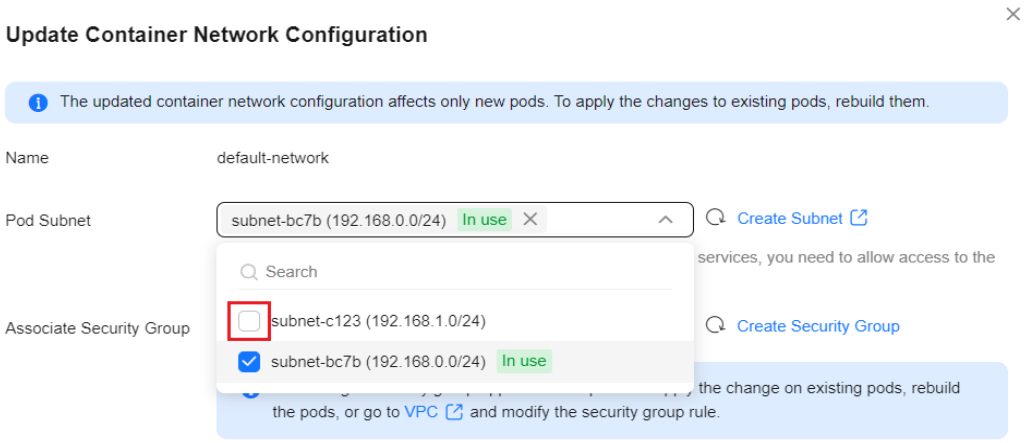
Pod subnets can be deleted from clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

Step 1 Log in to the CCE console and access the CCE Turbo cluster console.

Step 2 On the **Settings** page, click the **Network** tab.

Step 3 In the **Container Network** area, click **Update** in the **Operation** column of **default-network (Default Container Subnet)**.

Step 4 In the **Pod Subnet** area, **deselect** the subnets to be deleted and click **OK**.



NOTICE

- Deleting a pod subnet is risky. Make sure that no ENI in the cluster is using the subnet that you plan to delete. This includes ENIs that are being used by pods and prebound ENIs in the cluster.

  To find out if any ENIs are being used by a cluster, copy the ID of the subnet that you plan to delete. Then, use this ID to filter the ENIs and sub-ENIs associated with the VPC subnet on the **Network Interfaces** tab page. If the name or description of any of the filtered ENIs contains a cluster ID, then those ENIs are being used by the cluster.
- After a subnet is deleted, the security group of the cluster node will not automatically remove the rules associated with the subnet. Make sure that no ENIs in the cluster are still using the subnet and manually clear these rules that are associated with it.

----End

7.2.2.3 Binding a Security Group to a Pod Using an Annotation

Application Scenarios

In Cloud Native Network 2.0, pods use ENIs or sub-ENIs of the VPC. You can configure a security group for a pod using a pod's annotation.

Configure a security group in either of the following cases:

- To newly bind a security group to a pod, use annotation **yangtse.io/security-group-ids**.
- To bind more security groups to a pod, use annotation **yangtse.io/additional-security-group-ids**.

NOTE

The priority of the security group bound to a pod using annotation **yangtse.io/security-group-ids** is higher than those of the security groups in the **security group policy** (SecurityGroup) and **cluster container network configuration** (NetworkAttachmentDefinition).

Prerequisites

A CCE Turbo cluster is available and the cluster version meets the following requirements:

- v1.23: v1.23.16-r0 or later
- v1.25: v1.25.11-r0 or later
- v1.27: v1.27.8-r0 or later
- v1.28: v1.28.6-r0 or later
- v1.29: v1.29.2-r0 or later
- Versions later than v1.29

Using kubectl

- Create a workload with a security group configured. The security group bound to the pod is subject to the one configured using an annotation.

  NOTE

  If the pod has been bound to a security group, the configuration will be overwritten.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/security-group-ids: ***** # Security group ID. Use commas (,) to separate multiple security groups.
    spec:
      containers:
```



```
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

- Add an additional security group for the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/additional-security-group-ids: *****  # Security group ID. Use commas (,) to
separate multiple security groups.
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

Table 7-5 Annotations for configuring a security group for a pod

Annotation	Description	Value Range
yangtse.io/security-group-ids	Configure a security group for a pod. The security group of the pod is subject to the one configured using this annotation. If the pod already has a security group, the original security group will be overwritten.	Security group IDs. A maximum of five IDs are allowed. Use commas (,) to separate multiple security groups.

Annotation	Description	Value Range
yangtse.io/additional-security-group-ids	Add more security groups to a pod.	Security group IDs. The total number of newly added security group IDs and existing security group IDs cannot exceed 5. Use commas (,) to separate multiple security groups.

7.2.2.4 Binding a Security Group to a Workload Using a Security Group Policy

In Cloud Native Network 2.0, pods use VPC ENIs or sub-ENIs for networking. You can directly bind security groups and EIPs to pods. To bind CCE pods with security groups, CCE provides a custom resource object named **SecurityGroup**. Using this resource object, you can customize security isolation for workloads.

NOTE

The priority of the security group bound to pods using the security group policy is higher than that of the security group in the **NetworkAttachmentDefinition**.

Notes and Constraints

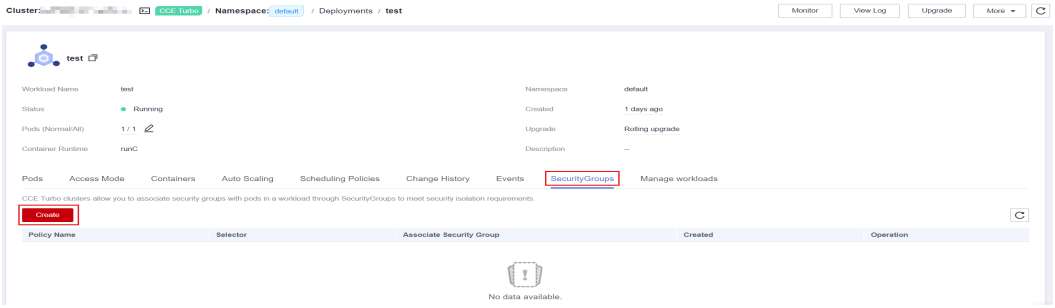
- This function is supported for CCE Turbo clusters of v1.19 and later. Upgrade your CCE Turbo clusters if their versions are earlier than v1.19.
- A workload can be bound to a maximum of five security groups.

Using the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. On the displayed page, click the desired workload name.

Step 3 Switch to the **SecurityGroups** tab and click **Create**.



Step 4 Set the parameters as described in **Table 7-6**.

Table 7-6 Configuration parameters

Parameter	Description	Example
Security Group Policy Name	Enter a security policy name. Enter 1 to 63 characters. The value must start with a lowercase letter and cannot end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.	security-group
Associate Security Group	The selected security group will be bound to the ENI or supplementary ENI of the selected workload. A maximum of five security groups can be selected from the drop-down list. You must select one or multiple security groups to create a SecurityGroup. If no security group has not been created, click **Create Security Group**. After the security group is created, click the refresh button. **NOTICE** • A maximum of five security groups can be selected. • Hover the cursor on the security group name, and you can obtain details about the security group.	64566556-bd6f-48fb-b2c6-df8f44617953 5451f1b0-bd6f-48fb-b2c6-df8f44617953

Step 5 After setting the parameters, click **OK**.

After the security group is created, the system automatically returns to the security group list page where you can see the new security group.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a description file named **securitygroup-demo.yaml**.

vi securitygroup-demo.yaml

For example, create the following SecurityGroup to bind all nginx workloads with two security groups 64566556-bd6f-48fb-b2c6-df8f44617953 and 5451f1b0-bd6f-48fb-b2c6-df8f44617953 that have been created in advance. An example is as follows:

```
apiVersion: crd.yangtse.cni/v1
kind: SecurityGroup
metadata:
  name: demo
```



```
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: nginx
  securityGroups:
  - id: 64566556-bd6f-48fb-b2c6-df8f44617953
  - id: 5451f1b0-bd6f-48fb-b2c6-df8f44617953
```

Table 7-7 describes the parameters in the YAML file.

Table 7-7 Description

Field	Description	Mandatory
apiVersion	API version. The value is **crd.yangtse.cni/v1**.	Yes
kind	Type of the object to be created.	Yes
metadata	Metadata definition of the resource object.	Yes
name	Name of the SecurityGroup.	Yes
namespace	Name of the namespace.	Yes
spec	Detailed description of the SecurityGroup.	Yes
podSelector	Used to define the workload to be associated with security groups in the SecurityGroup.	Yes
securityGroups	Security group ID.	Yes

Step 3 Run the following command to create the SecurityGroup:

kubectl create -f securitygroup-demo.yaml

If the following information is displayed, the SecurityGroup is being created.

```
securitygroup.crd.yangtse.cni/demo created
```

Step 4 Run the following command to check the SecurityGroup:

kubectl get sg

If the name of the created SecurityGroup is **demo** in the command output, the SecurityGroup is created successfully.

```
NAME                POD-SELECTOR                    AGE
all-no              map[matchLabels:map[app:nginx]]   4h1m
s001test            map[matchLabels:map[app:nginx]]   19m
demo                map[matchLabels:map[app:nginx]]   2m9s
```

----End

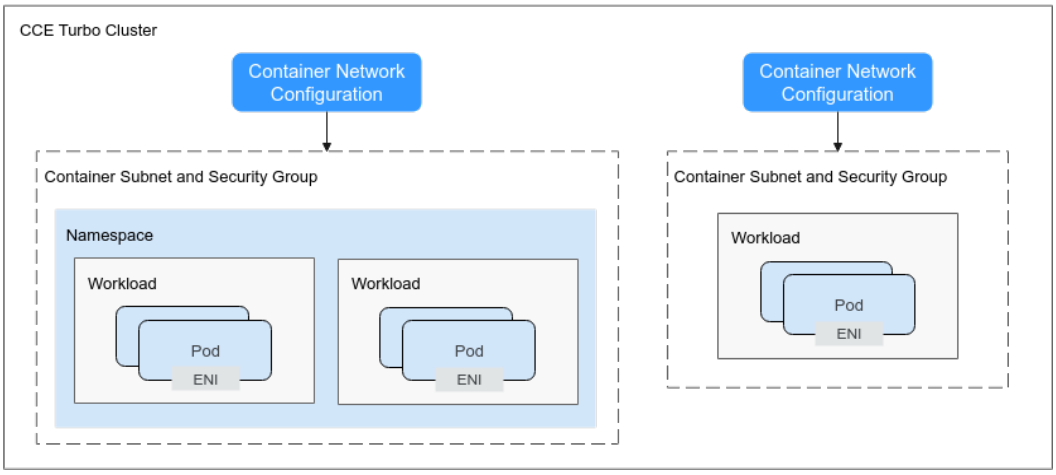
7.2.2.5 Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration

Scenario

In a CCE Turbo cluster, you can configure subnets and security groups for containers by namespace or workload using NetworkAttachmentDefinition **CRDs**.

To configure a particular container subnet and security group for a specified namespace or workload, create a container network configuration (NetworkAttachmentDefinition) and associate it with the target namespace or workload. In this way, service subnets can be planned or services can be securely isolated.

Figure 7-12 Custom container network settings



The following table lists the resources that a container network configuration can be associated with.

Table 7-8 Associated resources

Category	Resources a Container Network Configuration Can Associate with	
	Namespace	Workload
Subnet and security group configurations	All workloads created in the namespace associated with a container network configuration will use the same subnet and security group configurations.	The workloads using the same container network configuration will use the same subnet and security group configurations.
Supported cluster versions	Available only in CCE Turbo clusters of 1.23.8-r0, 1.25.3-r0, or later.	Available only in CCE Turbo clusters of 1.23.11-r0, 1.25.6-r0, 1.27.3-r0, 1.28.1-r0, or later.
Notes and Constraints	A namespace can be associated with only one container network configuration.	Only the custom container network configurations that are not associated with any namespace can be specified.

NOTE

- The priorities (in descending order) of the container network configurations used by a pod are as follows: Container network configuration directly associated with the pod > Container network configuration associated with the pod namespace > Default container network configuration of the cluster (**default-network**)
- If **default-network** is available in a cluster, it takes effect on all pods where no custom container network configuration has been configured. The default container subnet in the network settings on the **Overview** page is the container subnet in **default-network**. **default-network** cannot be deleted.
- If there is only one container network configuration in a cluster, it is the default container network configuration. If there are multiple container network configurations in a cluster, the one with annotation **yangtse.io/default-network: true** is the default container network configuration. If there is no default container network configuration in a cluster, the pod that is not associated with any container network configuration fails to be start after being created because no NIC can be allocated for the pod.

Notes and Constraints

- Only the default container network configuration supports dynamic pre-binding of container NICs. When the quota of node NICs is used up, the pod that uses the custom container network configuration attempts to unbind the pre-bound NIC of the default container network configuration, leading to slower pod startup. Therefore, if you need to frequently use the custom container network configuration, disable dynamic pre-binding of global container NICs in the target cluster. If you require high-speed pod elasticity using the default container network configuration, properly plan dynamic pre-binding of container NICs in the target node pool based on scheduling.
- If a workload with a fixed IP address needs to be associated with a new container network configuration, the fixed IP address will be invalid when pods are rebuilt. In this case, delete the workload, release the fixed IP address, and create a workload again.
- Before deleting a custom container network configuration, delete the pods (with the **cni.yangtse.io/network-status** annotation) created using the configuration in the target namespace. For details, see **Deleting a Container Network Configuration**.

Creating a Container Network Configuration of the Namespace Type

After such a container network configuration is created and associated with a namespace, all workloads created in the namespace will use the same subnet and security group configurations.

Operations on the Console

Step 1 Log in to the CCE console.

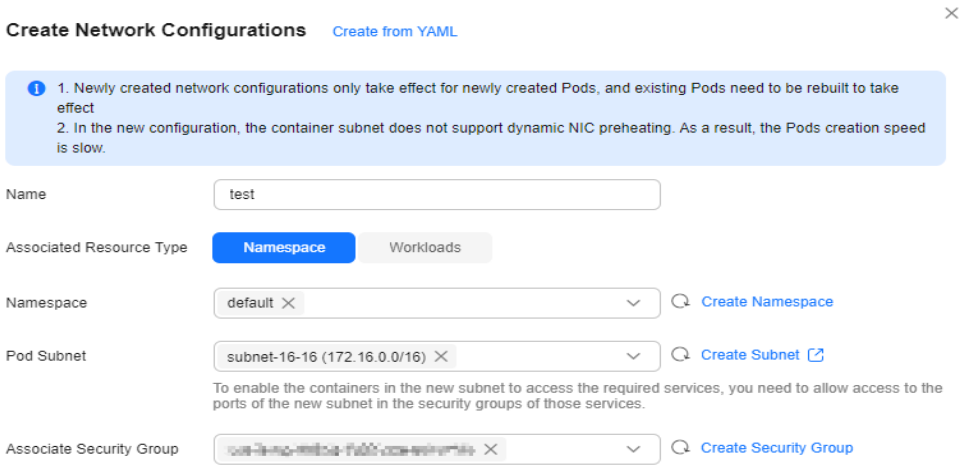
Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Network** tab.

Step 3 View the **Container Network Security Policy Configuration (Namespace Level)** and click **Add**. In the window that is displayed, configure parameters such as the pod subnet and security group.

- **Name**: Enter a name that contains a maximum of 63 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.

- **Associated Resource Type**: resource type associated with the custom container network configuration. For details, see **Table 7-8**. To create a container network configuration of the namespace type, select **Namespace**.
- **Namespace**: Select the namespace to be associated. The namespaces associated with different container network configurations must be unique. If no namespace is available, click **Create Namespace** to create one.
- **Pod Subnet**: Select a subnet. If no subnet is available, click **Create Subnet** to create one. After the subnet is created, click the refresh button. A maximum of 20 subnets can be selected for clusters of a version earlier than v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, or v1.30.4-r0. A maximum of 100 subnets can be selected for clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later.
- **Associate Security Group**: The default value is the container ENI security group. You can also click **Create Security Group** to create one. After the security group is created, click the refresh button. A maximum of five security groups can be selected.

Figure 7-13 Creating a container network configuration of the namespace type



Step 4 Click **OK**. After the creation, you will be redirected to the custom container network configuration list, where the new container network configuration is included.

Figure 7-14 Container network configuration list

----End

Operations Using kubectl

This section describes how to use kubectl to create a container network configuration of the namespace type.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Modify the **networkattachment-test.yaml** file.

```
vi networkattachment-test.yaml
```

Example file content:

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    yangtse.io/project-id: 05e38**
  name: example
  namespace: kube-system
spec:
  config: |
   {
    "type":"eni-neutron",
    "args":{
      "securityGroups":"41891**",
      "subnets":[
        {
          "subnetID":"27d95**"
        }
      ]
    },
    "selector":{
      "namespaceSelector":{
        "matchLabels":{
          "kubernetes.io/metadata.name":"default"
        }
      }
    }
   }
```

Table 7-9 Key parameters

Parameter	Mandatory	Type	Description
apiVersion	Yes	String	API version. The value is fixed at **k8s.cni.cncf.io/v1**.
kind	Yes	String	Type of the object to be created. The value is fixed at **NetworkAttachmentDefinition**.
yangtse.io/project-id	Yes	String	Project ID in the current region. For details, see **Obtaining a Project ID**.
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to **kube-system**.

Parameter	Mandatory	Type	Description
config	Yes	Table 7-10 object	Configuration content, which is a string in JSON format.

Table 7-10 config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at **eni-neutron**.
args	No	Table 7-11 object	Configuration parameters.
selector	No	Table 7-12 object	Namespace in which the configuration takes effect.

Table 7-11 args parameters

Parameter	Mandatory	Type	Description
securityGroups	No	String	Security group ID. If no security group is planned, ensure that the security group is the same as that in **default-network**. How to obtain: Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups**. Click the target security group name and copy the ID on the **Summary** tab page.
subnets	Yes	Array of subnetID Objects	List of container subnet IDs. At least one subnet ID must be entered. The format is as follows: `[{"subnetID":"27d95**"},{"subnetID":"827bb**"},{"subnetID":"bdd6b**"}]` Subnet ID not used by the cluster in the same VPC. How to obtain: Log in to the VPC console. In the navigation pane, choose **Virtual Private Cloud** > **Subnets**. Click the target subnet name and copy the **Subnet ID** on the **Summary** tab page.

Table 7-12 selector parameters

Parameter	Mandatory	Type	Description
namespaceSelector	No	matchLabels object	A Kubernetes standard selector. Enter the namespace label in the following format: `"matchLabels":{ "kubernetes.io/metadata.name":"default" }` The namespaces of different configurations cannot overlap.

Step 3 Create the container network configuration.

```
kubectl create -f networkattachment-test.yaml
```

If information similar to the following is displayed, the configuration has been created:

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

----End

Creating a Container Network Configuration of the Workload Type

After such a container network configuration is created, all workloads associated with it will use the same subnet and security group configurations.

Operations on the Console

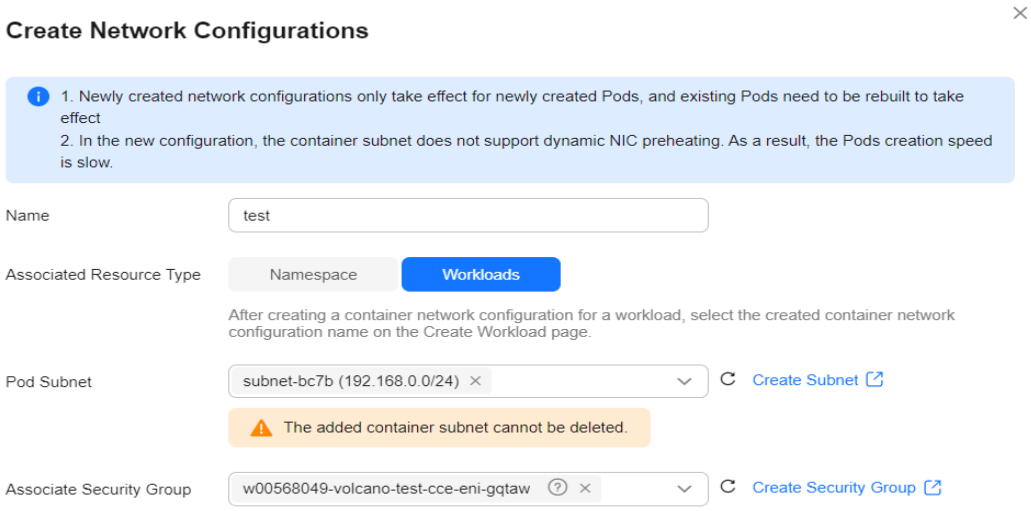
Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Network** tab.

Step 3 View the **Container Network Security Policy Configuration (Namespace Level)** and click **Add**. In the window that is displayed, configure parameters such as the pod subnet and security group.

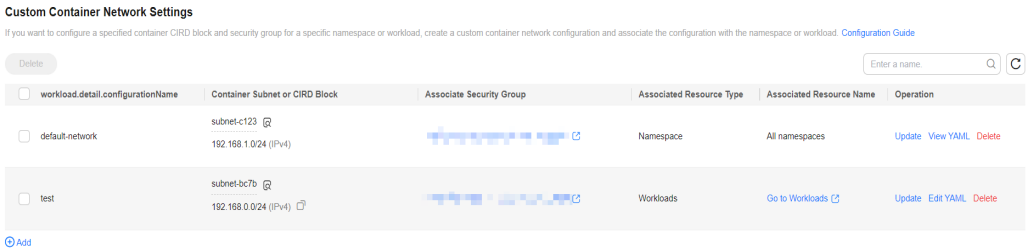
- **Name**: Enter a name that contains a maximum of 63 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.
- **Associated Resource Type**: resource type associated with the custom container network configuration. For details, see **Table 7-8**. To create a container network configuration of the workload type, select **Workload**.
- **Pod Subnet**: Select a subnet. If no subnet is available, click **Create Subnet** to create one. After the subnet is created, click the refresh button.
- **Associate Security Group**: The default value is the container ENI security group. You can also click **Create Security Group** to create one. After the security group is created, click the refresh button. A maximum of five security groups can be selected.

Figure 7-15 Creating a container network configuration of the workload type



Step 4 Click **OK**. After the creation, you will be redirected to the custom container network configuration list, where the new container network configuration is included.

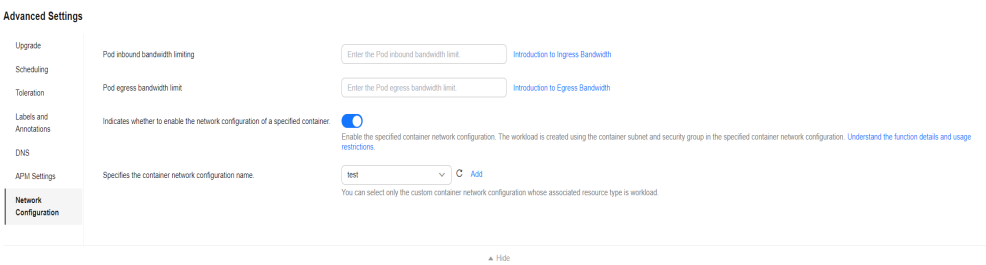
Figure 7-16 Container network configuration list



Step 5 When creating a workload, you can select a custom container network configuration.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner of the page. In the **Advanced Settings** area, choose **Network Configuration** and determine whether to enable a specified container network configuration.
3. Select an existing container network configuration. If no configuration is available, click **Add** to create one.

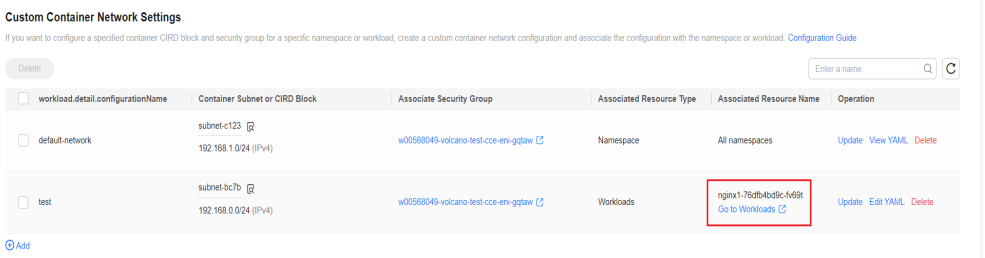
Figure 7-17 Selecting a container network configuration



4. After the configuration, click **Create Workload**.

   Return to the **Settings** page. In the container network configuration list, the name of the resource associated with the created container network configuration is displayed.

   Figure 7-18 Resource associated with a container network configuration



----End

Operations Using kubectl

This section describes how to use kubectl to create a container network configuration of the workload type.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Modify the **networkattachment-test.yaml** file.

```
vi networkattachment-test.yaml
```

Example file content:

```
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    yangtse.io/project-id: 80d5a**
  name: example
  namespace: kube-system
spec:
  config: |
   {
    "type":"eni-neutron",
    "args":{
      "securityGroups":"f4983**",
      "subnets":[
        {
          "subnetID":"5594b**"
        }
      ]
    }
   }
```

Table 7-13 Key parameters

Parameter	Mandatory	Type	Description
apiVersion	Yes	String	API version. The value is fixed at **k8s.cni.cncf.io/v1**.

Parameter	Mandatory	Type	Description
kind	Yes	String	Type of the object to be created. The value is fixed at **NetworkAttachmentDefinition**.
yangtse.io/project-id	Yes	String	Project ID in the current region. For details, see **Obtaining a Project ID**.
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to **kube-system**.
config	Yes	**Table 7-14** object	Configuration content, which is a string in JSON format.

Table 7-14 config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at **eni-neutron**.
args	No	**Table 7-15** object	Configuration parameters.

Table 7-15 args parameters

Parameter	Mandatory	Type	Description
securityGroups	No	String	Security group ID. If no security group is planned, ensure that the security group is the same as that in **default-network**. How to obtain: Log in to the VPC console. In the navigation pane, choose **Access Control** > **Security Groups**. Click the target security group name and copy the ID on the **Summary** tab page.

Parameter	Mandatory	Type	Description
subnets	Yes	Array of subnetID Objects	List of container subnet IDs. At least one subnet ID must be entered. The format is as follows: [{"subnetID":"27d95**"},{"subnetID":"827bb**"},{"subnetID":"bdd6b**"}] Subnet ID not used by the cluster in the same VPC. How to obtain: Log in to the VPC console. In the navigation pane, choose **Virtual Private Cloud** > **Subnets**. Click the target subnet name and copy the **Subnet ID** on the **Summary** tab page.

Step 3 Create a NetworkAttachmentDefinition.

```
kubectl create -f networkattachment-test.yaml
```

If information similar to the following is displayed, the configuration has been created.

```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

Step 4 Create a Deployment workload and associate it with the newly created container network configuration.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
        yangtse.io/network: "example" # Name of the custom container network configuration, which can be used to obtain all pods associated with the container network configuration by label
      annotations:
        yangtse.io/network: "example" # Name of the custom container network configuration
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

- **yangtse.io/network**: name of the specified custom container network configuration. Only a container network configuration that is not associated

with any namespace can be specified. Add this parameter to the label so that you can use the label to obtain all pods associated with this container network configuration.

----End

Verify That a Container Network Configuration Is Associated with a Namespace or Workload

Verify that a subnet and security group specified in the container network configuration have been associated with a workload. To check if a namespace is associated with a container network configuration, check whether a workload in the namespace is associated with a subnet and security group.

Step 1 Verify that a subnet is associated.

1. Check the IP address of a pod for a workload.

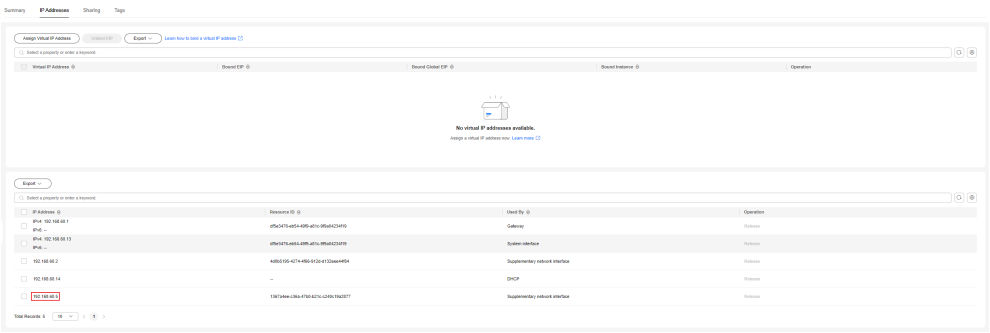
```
kubectl get pod -o wide
```

Information similar to the following is displayed:

```
NAME                     READY   STATUS    RESTARTS   AGE     IP              NODE
NOMINATED NODE   READINESS GATES
nginx-85dbdb8c5d-ng5bb   1/1     Running   0          5d18h   192.168.60.5    ca50a5ae-e1ef-41c6-
b3fc-6ebcd10a1e07   <none>           <none>
```

2. Access **Network Console**. In the navigation pane, choose **Virtual Private Cloud** > **Subnets**. On the displayed page, click the subnet name.
3. Click **IP Addresses**. If the IP address of the pod is displayed, the subnet is associated.

Figure 7-19 Viewing the IP address associated with a pod



Step 2 Verify that a security group is associated.

1. Switch back to **Network Console**. In the navigation pane, choose **Access Control** > **Security Groups**. On the displayed page, click the security group name.
2. On the **Associated Instances** tab, click **Supplementary Network Interfaces**.
3. View the private IP address of the pod in the **Private IP Address** column. If the private IP address of the pod is displayed, the security group is associated.

Figure 7-20 Viewing the IP address associated with a security group

----End

Deleting a Container Network Configuration

You can delete the new container network configuration or view its YAML file.

NOTE

Before deleting a container network configuration, delete all pods using the configuration. Otherwise, the deletion will fail.

1. Run the following command to filter the pods that uses the configuration in the cluster (**example** is used as an example):
   ```
   kubectl get pod -A -o=jsonpath="{.items[?(@.metadata.annotations.cni\.yangtse\.io/network-status=='[{\"name\":\"example\"}]')]['metadata.namespace', 'metadata.name']}"
   ```
   The command output contains the pod name and namespace associated with the configuration.
2. Delete the owner of the pod. The owner may be a Deployment, StatefulSet, DaemonSet, or Job.

7.2.2.6 Configuring a Static IP Address for a Pod

Application Scenarios

In Cloud Native Network 2.0, each pod is associated with an ENI, providing a static IP address to the StatefulSet pods (container ENI). This is a common practice in access control, service registration, service discovery, and log audit of static IP addresses.

For example, if a StatefulSet service needs to control the access of a cloud database, you can fix the pod IP address of the service and configure the security group of the cloud database to allow only the service IP address to access the database.

Notes and Constraints

- You can configure a static IP address for a pod only in CCE Turbo clusters of the following versions:
  - v1.23: v1.23.7-r0 or later
  - v1.25: v1.25.3-r0 or later
  - v1.25 or later
- Currently, only StatefulSet pods or pods without **ownerReferences** can be configured with static IP addresses. Deployments, DaemonSets, and other types of workloads cannot be configured with static IP addresses. In addition, pods with **hostNetwork** configured cannot be configured with static IP addresses.
- Do not configure static IP addresses for services that do not have specific requirements on pod IP addresses. Otherwise, the pod rebuilding takes a longer time and the IP address usage decreases.
- The annotations of the static IP address of the pod object cannot be directly modified. Otherwise, the modification does not take effect in the background. To modify the annotations, modify the **annotations** configuration in the **spec.template** field of the corresponding StatefulSet workload.

- If there are no ENIs left on the node where the pod with a static IP address is rebuilt and scheduled (the pre-bound ENIs also occupy the ENI quota), the static IP address ENIs preempt the pre-bound ENIs. In this case, the pod starts slightly slowly. If a node uses a static IP address, properly configure the dynamic pre-binding policy for the node to ensure that not all ENIs are pre-bound.

Using kubectl

You can add annotations to a StatefulSet to enable or disable the static IP address function of the pod.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        pod.alpha.kubernetes.io/initialized: 'true'
        yangtse.io/static-ip: 'true'
        yangtse.io/static-ip-expire-no-cascading: 'false'
        yangtse.io/static-ip-expire-duration: 5m
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```

Table 7-16 Annotations of the pod's static IP address

Annotation	Default Value	Description	Value Range
yangtse.io/static-ip	false	Specifies whether to enable the static IP address of a pod. This function is supported only for StatefulSet pods or pods without **ownerReferences**. This function is disabled by default.	**false** or **true**

Annotation	Default Value	Description	Value Range
yangtse.io/static-ip-expire-duration	5m	Specifies the interval for reclaiming the expired ENI of the static IP address after the pod with a static IP address is deleted.	The time format is Go time type, for example, 1h30m and 5m. For details, see **Go time type**.
yangtse.io/static-ip-expire-no-cascading	false	Specifies whether to disable cascading reclamation of StatefulSet workloads. The default value is **false**, indicating that the corresponding static IP address ENI will be deleted with the StatefulSet workload. If you want to retain the static IP address for a new StatefulSet with the same name during the interval for reclaiming the expired ENI, set the value to **true**.	**false** or **true**

7.2.2.7 Configuring an EIP for a Pod

Application Scenarios

In Cloud Native Network 2.0, pods use VPC ENIs or sub-ENIs for networking. You can directly bind EIPs to pods.

To associate an EIP with a pod, simply set the value of the **yangtse.io/pod-with-eip** annotation to **true** when creating the pod. Then, the EIP will be automatically allocated and bound to the pod.

Prerequisites

A CCE Turbo cluster is available and the cluster version meets the requirements listed in the following table.

Scenario	Cluster Version
Allocating an EIP with a pod	• v1.19: v1.19.16-r20 or later • v1.21: v1.21.10-r0 or later • v1.23: v1.23.8-r0 or later • v1.25: v1.25.3-r0 or later • Versions later than v1.25

Scenario	Cluster Version
Using an existing EIP for a pod	v1.23: v1.23.16-r0 or later v1.25: v1.25.11-r0 or later v1.27: v1.27.8-r0 or later v1.28: v1.28.6-r0 or later v1.29: v1.29.2-r0 or later Versions later than v1.29

Notes and Constraints

- To access a pod bound with an EIP from the Internet, add security group rules to allow the target request traffic.
- Only one EIP can be bound to a pod.
- Configure the EIP-related annotation when creating a pod. After the pod is created, the annotations related to the EIP cannot be modified.
- Do not perform operations on the EIP associated with a pod through the EIP console or API. Otherwise, the EIP may malfunction. The operations include changing the billing mode of the EIP, changing the EIP name, and deleting, unbinding, or binding the EIP.
- After an automatically allocated EIP is manually deleted, the network malfunctions. In this case, rebuild the pod.

Allocating an EIP with a Pod

If you set the **pod-with-eip** annotation to **true** when creating a pod, an EIP will be automatically allocated and bound to the pod.

NOTE

- CCE will automatically add cluster ID, namespace, and pod name labels to an EIP that has been allocated automatically.
- If a pod is created with an automatically allocated EIP, deleting the pod will also delete the EIP.

The following uses a Deployment named **nginx** as an example. For details about annotations, see **Table 7-18**.

- For an automatically allocated EIP with a **dedicated bandwidth** when you create a Deployment, you do not need to specify the bandwidth ID. The following shows an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```



```
      app: nginx
    annotations:
      yangtse.io/pod-with-eip: "true" # An EIP will be automatically allocated when the pod is
created.
      yangtse.io/eip-bandwidth-size: "5" # EIP bandwidth
      yangtse.io/eip-network-type: 5_bgp # EIP type
      yangtse.io/eip-charge-mode: bandwidth # EIP billing mode
      yangtse.io/eip-bandwidth-name: <eip_bandwidth_name> # EIP bandwidth name
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

Table 7-17 Annotations of an EIP with a dedicated bandwidth

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to allocate an EIP with a pod and bind the EIP to the pod	**false** or **true**
yangtse.io/eip-bandwidth-size	No	5	Bandwidth, in Mbit/s	The value range varies depending on the region and bandwidth billing mode. For details, see the purchase page on the EIP console. For example, in the AP-Singapore region, if an EIP is billed by bandwidth, the bandwidth ranges from 1 Mbit/s to 2000 Mbit/s; if an EIP is billed by traffic, the bandwidth ranges from 1 Mbit/s to 300 Mbit/s.

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-network-type	No	5_bgp	EIP type	The type varies depending on the region. For details, see the purchase page on the EIP console. For example, the following options are available in in the AP-Singapore region: • **5_bgp**: dynamic BGP
yangtse.io/eip-charge-mode	No	None	Billed by traffic or bandwidth **You are advised to configure this parameter.** If this parameter is left blank, no billing mode is specified. In this case, the default value of the EIP API in the region is used.	• **bandwidth**: billed by bandwidth • **traffic**: billed by traffic
yangtse.io/eip-bandwidth-name	No	Pod name	Bandwidth name	• Enter 1 to 64 characters. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • Minimum length: 1 character • Maximum length: 64 characters

- For an automatically allocated EIP with a **shared bandwidth** when you create a Deployment, you are required to specify the bandwidth ID. The following shows an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
```



```
replicas: 3
selector:
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
    annotations:
      yangtse.io/pod-with-eip: "true" # An EIP will be automatically allocated when the pod is created.
      yangtse.io/eip-network-type: 5_bgp # EIP type
      yangtse.io/eip-bandwidth-id: <eip_bandwidth_id> # Shared bandwidth ID of the EIP
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

Table 7-18 Annotations of an EIP with a shared bandwidth

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to allocate an EIP with a pod and bind the EIP to the pod	**false** or **true**
yangtse.io/eip-network-type	No	5_bgp	EIP type	• 5_bgp • 5_union • 5_sbgp The specific type varies with regions. For details, see the EIP console.

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-id	Mandatory when a shared bandwidth is used	None	ID of an existing bandwidth • If this parameter is not specified, the EIP with a dedicated bandwidth is used by default. For details about how to configure parameters for an EIP with a dedicated bandwidth, see Table 7-17. • Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	None

Using an Existing EIP for a Pod

If you configure the **yangtse.io/eip-id** annotation when creating a pod, the EIP will be automatically bound to the pod.

NOTE

- After an existing EIP is bound to a pod, CCE will automatically add the cluster ID, namespace, and pod name labels to the EIP.
- When a pod using an existing EIP is deleted, the EIP will be retained. CCE will automatically delete the labels for the cluster ID, namespace, and pod name that were added when the EIP was bound.

The following example shows how to create a Deployment named **nginx** with a single pod with an EIP automatically bound.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/eip-id: <eip_id> # ID of an existing EIP
    spec:
```



```
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

Table 7-19 Annotations for using an existing EIP

Annotation	Mandatory	Description
yangtse.io/eip-id	Yes	EIP ID **How to obtain**: Log in to the EIP console, click the name of the target EIP in the EIP list, and copy the ID field.

Checking Whether the EIP Bound to the Pod Is Available

After an EIP is allocated to a pod, the container networking controller binds the EIP to the pod and writes the allocation result back to the pod's **yangtse.io/allocated-ipv4-eip** annotation. The startup time of the pod's service containers may be earlier than the time when the EIP allocation result is written back.

You can configure an init container for the pod, associate the **yangtse.io/allocated-ipv4-eip** annotation with the init container through a downwardAPI volume, and check whether the EIP has been allocated in the init container. You can configure the init container as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
  annotations:
    yangtse.io/pod-with-eip: "true"
    yangtse.io/eip-bandwidth-size: "5"
    yangtse.io/eip-network-type: 5_bgp
    yangtse.io/eip-charge-mode: bandwidth
    yangtse.io/eip-bandwidth-name: "xxx"
spec:
  initContainers:
  - name: init
    image: busybox:latest
    command: ['timeout', '60', 'sh', '-c', "until grep -E '[0-9]+' /etc/eipinfo/allocated-ipv4-eip; do echo waiting for allocated-ipv4-eip; sleep 2; done"]
    volumeMounts:
      - name: eipinfo
        mountPath: /etc/eipinfo
  volumes:
    - name: eipinfo
      downwardAPI:
        items:
          - path: "allocated-ipv4-eip"
            fieldRef:
```



```
            fieldPath: metadata.annotations['yangtse.io/allocated-ipv4-eip']
...
```

7.2.2.8 Configuring a Static EIP for a Pod

Application Scenarios

In Cloud Native Network 2.0, static public IP addresses (EIPs) can be assigned to StatefulSets or pods created directly.

Constraints

- You can configure a static EIP for a pod only in CCE Turbo clusters of the following versions:
  - v1.19: v1.19.16-r20 or later
  - v1.21: v1.21.10-r0 or later
  - v1.23: v1.23.8-r0 or later
  - v1.25: v1.25.3-r0 or later
  - v1.25 or later
- The static EIP function must be enabled together with the function of automatically allocating an EIP for a pod. For details, see **Configuring an EIP for a Pod**.
- Only StatefulSet pods or pods created directly can be bound with static EIPs. Deployments, DaemonSets, and other types of workloads cannot be bound with static EIPs.
- After a static EIP is allocated, the EIP attributes cannot be modified through the pod within the EIP lifecycle (before the EIP expires or it is still being used by the pod).
- Do not configure a static EIP for services that do not have specific requirements on pod EIPs. Otherwise, the pod rebuilding takes a longer time.

Configuring a Static EIP for a Pod

When creating a pod to be bound with a static IP address, configure the EIP annotation. Then, an EIP will be automatically allocated and bound to the pod.

The following uses a StatefulSet named **nginx** as an example. For details about annotations, see **Table 7-20**.

- For an automatically allocated EIP with a **dedicated bandwidth** when you create a StatefulSet, you do not need to specify the bandwidth ID. The following shows an example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
```



```
    labels:
      app: nginx
    annotations:
      yangtse.io/static-eip: 'true'  # Static EIP bound to the pod
      yangtse.io/static-eip-expire-no-cascading: 'false'  # Deleting the EIP with the associated
workload
      yangtse.io/static-eip-expire-duration: 5m  # Interval for reclaiming expired EIPs
      yangtse.io/pod-with-eip: 'true'  # An EIP will be automatically allocated when the pod is
created.
      yangtse.io/eip-bandwidth-size: '5'  # EIP bandwidth
      yangtse.io/eip-network-type: 5_bgp  # EIP type
      yangtse.io/eip-charge-mode: bandwidth  # EIP billing mode
  spec:
    containers:
      - name: container-0
        image: nginx:alpine
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
    imagePullSecrets:
      - name: default-secret
```

- For an automatically allocated EIP with a **shared bandwidth** when you create a StatefulSet, you are required to specify the bandwidth ID. The following shows an example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx
spec:
  serviceName: nginx
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        yangtse.io/static-eip: 'true'  # Static EIP bound to the pod
        yangtse.io/pod-with-eip: 'true'  # An EIP will be automatically allocated when the pod is
created.
        yangtse.io/eip-network-type: 5_bgp  # EIP type
        yangtse.io/eip-bandwidth-id: <eip_bandwidth_id>  # Shared bandwidth ID of the EIP
    spec:
      containers:
        - name: container-0
          image: nginx:alpine
          resources:
            limits:
              cpu: 100m
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
      imagePullSecrets:
        - name: default-secret
```


Table 7-20 Annotations of the pod's static EIP

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/static-eip	Yes	false	Specifies whether to enable the static EIP of a pod. This function is supported only for StatefulSet pods or pods without **ownerReferences**. This function is disabled by default.	**false** or **true**
yangtse.io/static-eip-expire-duration	No	5m	Specifies the interval for reclaiming the expired static EIP after the pod with a static EIP is deleted.	The time format is Go time type, for example, 1h30m and 5m. For details, see **Go time type**.
yangtse.io/static-eip-expire-no-cascading	No	false	Specifies whether to disable cascading reclamation of StatefulSet workloads. The default value is **false**, indicating that the corresponding static EIP will be deleted with the StatefulSet workload. If you want to retain the static EIP for a new StatefulSet with the same name during the interval for reclaiming the expired EIP, set the value to **true**.	**false** or **true**

Table 7-21 Annotations of an EIP with a dedicated bandwidth

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to allocate an EIP with a pod and bind the EIP to the pod	**false** or **true**

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-size	No	5	Bandwidth, in Mbit/s	The value range varies depending on the region and bandwidth billing mode. For details, see the purchase page on the EIP console. For example, in the AP-Singapore region, if an EIP is billed by bandwidth, the bandwidth ranges from 1 Mbit/s to 2000 Mbit/s; if an EIP is billed by traffic, the bandwidth ranges from 1 Mbit/s to 300 Mbit/s.
yangtse.io/eip-network-type	No	5_bgp	EIP type	The type varies depending on the region. For details, see the purchase page on the EIP console. For example, the following options are available in in the AP-Singapore region: • **5_bgp**: dynamic BGP
yangtse.io/eip-charge-mode	No	None	Billed by traffic or bandwidth **You are advised to configure this parameter.** If this parameter is left blank, no billing mode is specified. In this case, the default value of the EIP API in the region is used.	• **bandwidth**: billed by bandwidth • **traffic**: billed by traffic

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-name	No	Pod name	Bandwidth name	• Enter 1 to 64 characters. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. • Minimum length: 1 character • Maximum length: 64 characters

Table 7-22 Annotations of an EIP with a shared bandwidth

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to allocate an EIP with a pod and bind the EIP to the pod	**false** or **true**
yangtse.io/eip-network-type	No	5_bgp	EIP type	• 5_bgp • 5_union • 5_sbgp The specific type varies with regions. For details, see the EIP console.

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-id	Mandatory when a shared bandwidth is used	None	ID of an existing bandwidth • If this parameter is not specified, the EIP with a dedicated bandwidth is used by default. For details about how to configure parameters for an EIP with a dedicated bandwidth, see **Table 7-17**. • Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	None

Deleting a Static EIP

After a pod is deleted, if another pod with the same name is created before the static EIP expires, the EIP can still be used. The static EIP is deleted only if there is no new pod with the name the same as that of the deleted pod before the EIP expires, or the function of deleting the EIP with the associated StatefulSet is enabled and the StatefulSet is deleted.

7.2.2.9 Configuring Shared Bandwidth for a Pod with IPv6 Dual-Stack ENIs

Application Scenarios

By default, pods with IPv6 dual-stack ENIs can access only the IPv6 private network. To access the public network, configure shared bandwidth for such pods.

Notes and Constraints

- Only CCE Turbo clusters are supported and the following constraints must be met:
  - IPv6 dual stack has been enabled for the cluster.
  - The cluster version is v1.23.8-r0, v1.25.3-r0, or later.
- The number of IPv6 ENIs that can be added to a shared bandwidth is limited by the tenant quota. The default value is 20.
- hostNetwork Pods are not supported.
- All types of workloads are supported. When configuring IPv6 shared bandwidth for workloads with the replicas attribute, such as Deployment and StatefulSet, ensure that the number of replicas and the maximum number of

pods during the upgrade are less than the remaining quota of IPv6 ENIs that can be added to the shared bandwidth.

- IPv6 dual-stack pod configured with shared bandwidth: When a pod is created, the CNI returns a success message only after the IPv6 dual-stack ENI is inserted into the shared bandwidth. When a pod is deleted, the IPv6 dual-stack ENI is removed from the shared bandwidth after the pod is completely deleted or the pod is in the deleting status for 30 seconds.
- If the IPv6 dual-stack ENI corresponding to the pod fails to be added to the shared bandwidth, an alarm event **FailedIPv6InsertBandwidth** is generated on the pod, for example, when the quota is exceeded or flow control is triggered. Rectify the fault based on the alarm event.
- On the **Shared Bandwidths** page of the EIP console, go to the target shared bandwidth details page, and click the **IPv6 Addresses** tab. The IPv6 dual-stack ENI whose **Associated Instance** is CCE is displayed. Do not remove the ENI directly on the page or using VPC APIs, otherwise, your services may be affected.

Using the CCE Console

When creating a workload, you can set the IPv6 shared bandwidth on the **Advanced Settings > Network Configuration** area.

Using kubectl

You can add annotations to a Deployment to specify the shared bandwidth to be added to the IPv6 dual-stack ENI of the pod. The following is an example:

```
...
spec:
  selector:
    matchLabels:
      app: demo
      version: v1
  template:
    metadata:
      annotations:
        yangtse.io/ipv6-bandwidth-id: "xxx"
```

- **yangtse.io/ipv6-bandwidth-id**: specifies the ID of the shared bandwidth. The IPv6 dual-stack ENIs corresponding to the pod will be added to the shared bandwidth. You can query the ID on the **Shared Bandwidths** page on the EIP console.

7.2.3 VPC Network Settings

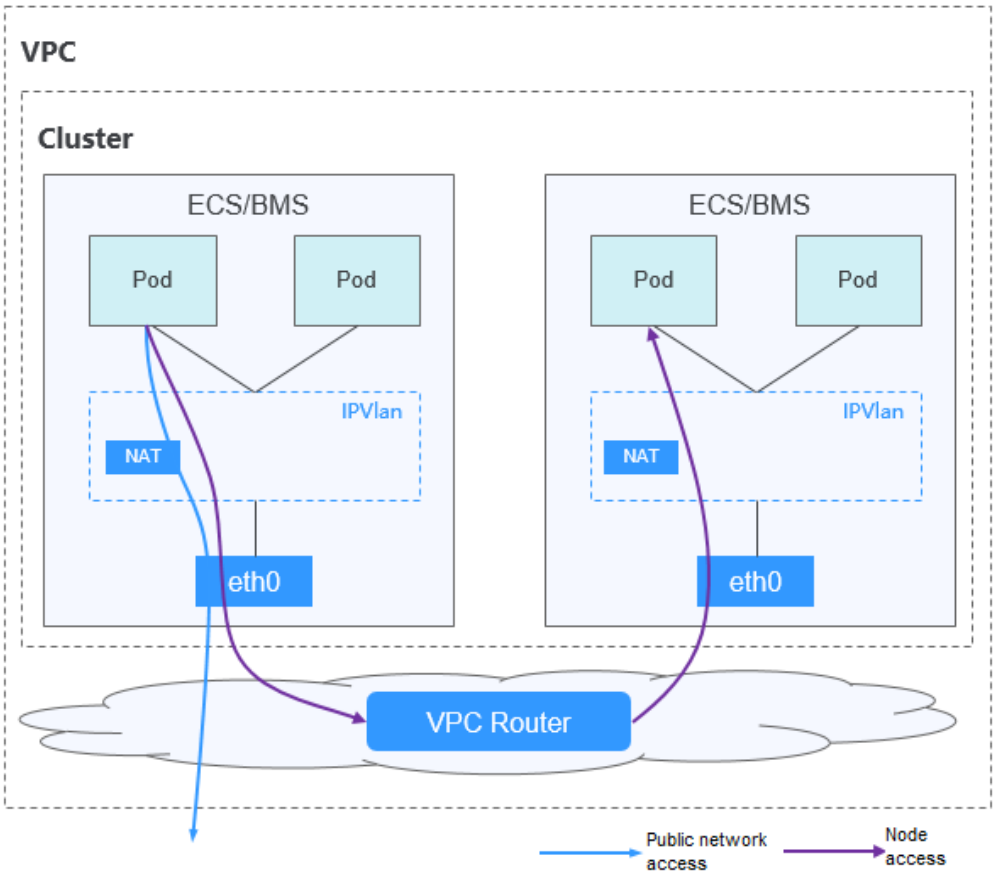
7.2.3.1 VPC Network Model

Model Definition

The VPC network model seamlessly combines VPC routing with the underlying network, making it ideal for high-performance scenarios. However, the maximum number of nodes allowed in a cluster is determined by the VPC route quota. In the VPC network model, container CIDR blocks are separate from node CIDR blocks. To allocate IP addresses to containers running on a node in a cluster, each node in

the cluster is allocated with a container CIDR block for a fixed number of IP addresses. The VPC network model outperforms the container tunnel network model in terms of performance because it does not have tunnel encapsulation overhead. When using the VPC network model in a cluster, the VPC route table automatically configures the routes between container CIDR blocks and VPC CIDR blocks. This means that pods within the cluster can be accessed directly from cloud servers in the same VPC, even if they are outside the cluster.

Figure 7-21 VPC network model



In a cluster using the VPC network model, network communication paths are as follows:

- Inter-pod communication on the same node: Packets are directly forwarded through IPvlan.
- Inter-pod communication on different nodes: The traffic accesses the default gateway by following the route specified in the VPC route table and then is forwarded to the target pod on another node using VPC routing.
- Pod communication with the Internet: When a container in a cluster needs to access the Internet, CCE uses NAT to translate the container's IP address into the node IP address so that the pod communicates externally using the node IP address.

> **NOTE**
>
> In a cluster using a VPC network, 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 are regarded as private CIDR blocks of the cluster by default. If the VPC to which the cluster resides uses a secondary CIDR block, operations such as creating or resetting a node will also add the secondary CIDR block to the private CIDR blocks.
>
> If a pod tries to access a private CIDR block, the source node will not perform NAT on the pod IP address. Instead, the upper-layer VPC can directly send the pod data packet to the destination, which means, the pod IP address is directly used to communicate with the private CIDR block in the cluster.
>
> If your VPC network cluster is of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, CCE allows you to use **nonMasqueradeCIDRs** to configure private CIDR blocks for your cluster to tailor your cluster to different application needs. For details, see **Accessing an IP Address Outside a Cluster That Uses a VPC Network Using Source Pod IP Addresses in the Cluster**.

Advantages and Disadvantages

Advantages

- High performance and simplified network fault locating are achieved by eliminating the need for tunnel encapsulation.
- A VPC route table automatically configures routes between container CIDR blocks and VPC CIDR blocks. This enables resources in the VPC to directly communicate with containers in the cluster.

  > **NOTE**
  >
  > Similarly, if the VPC is accessible to other VPCs or data centers and the VPC route table includes routes to the container CIDR blocks, resources in other VPCs or data centers can directly communicate with containers in the cluster, provided there are no conflicts between the network CIDR blocks.

Disadvantages

- The number of nodes is limited by the VPC route quota.
- Each node is assigned a CIDR block with a fixed size, which results in IP address wastage in the container CIDR block.
- Pods cannot directly use features like EIPs and security groups.

Application Scenarios

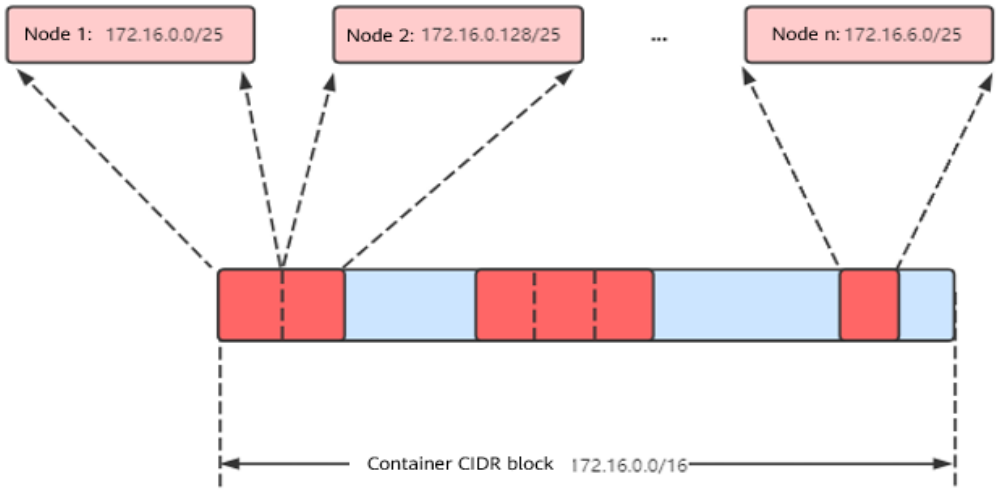
- High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, such as AI computing and big data computing.
- Small- and medium-scale networks: Due to the limitation on VPC route tables, it is recommended that the number of nodes in a cluster be less than or equal to 1000.

Container IP Address Management

The VPC network model assigns container IP addresses based on the following guidelines:

- Container CIDR blocks are separate from node CIDR blocks.
- IP addresses are allocated by node. One CIDR block with a fixed size (configurable) is allocated to each node in a cluster from the container CIDR block.
- A container CIDR block assigns CIDR blocks to new nodes in a cyclical sequence.
- IP addresses from the CIDR blocks assigned to a node are allocated to pods scheduled to that node in a cyclical manner.

Figure 7-22 IP address management of the VPC network



Maximum number of nodes that can be created in the cluster using the VPC network = Number of IP addresses in the container CIDR block/Number of IP addresses in the CIDR block allocated to the node by the container CIDR block

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. The mask of the container CIDR block allocated to a node is 25. That is, the number of container IP addresses on each node is 128. Therefore, a maximum of 512 (65536/128) nodes can be created. The number of nodes that can be added to a cluster is also determined by the available IP addresses in the node subnet and the scale of the cluster. For details, see **Recommendation for CIDR Block Planning**.

Recommendation for CIDR Block Planning

As explained in **Cluster Network Structure**, network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- **The three CIDR blocks cannot overlap.** Otherwise, a conflict occurs. All subnets (including those created from the secondary CIDR blocks) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.
- Ensure that **each CIDR block has sufficient IP addresses**.
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.

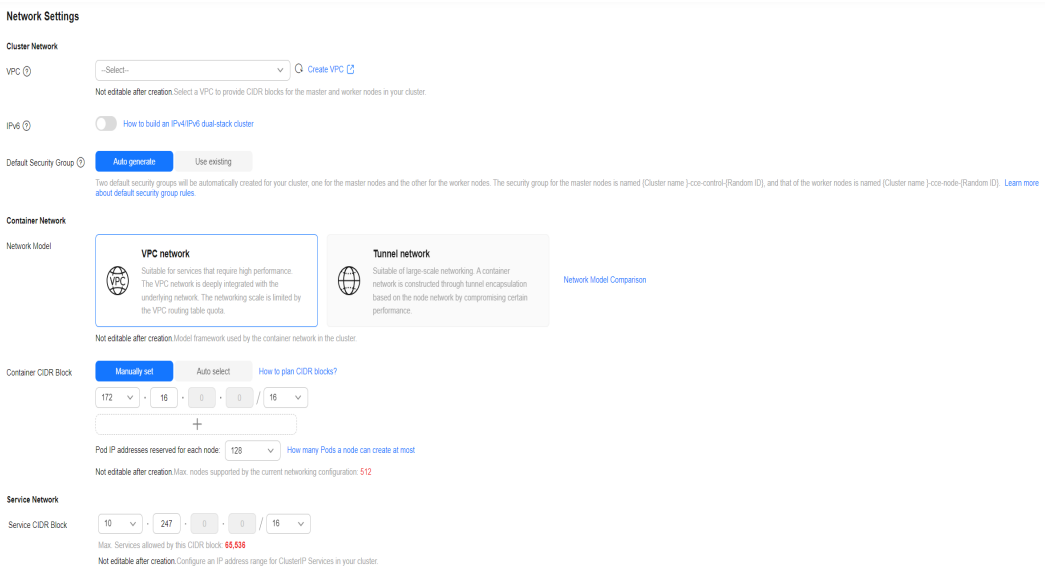
- The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings. For details, see **Maximum Number of Pods That Can Be Created on a Node**.

Assume that a cluster contains 200 nodes and the network model is VPC network.

In this case, the number of available IP addresses in the selected subnet must be greater than 200. Otherwise, nodes cannot be created due to insufficient IP addresses.

The container CIDR block is 172.16.0.0/16, and the number of available IP addresses is 65536. As described in **Container IP Address Management**, the VPC network is allocated a CIDR block with a fixed size (using the mask to determine the maximum number of container IP addresses allocated to each node). For example, if the upper limit is 128, the cluster supports a maximum of 512 (65536/128) nodes.

Figure 7-23 Configuring a container CIDR block (during cluster creation)



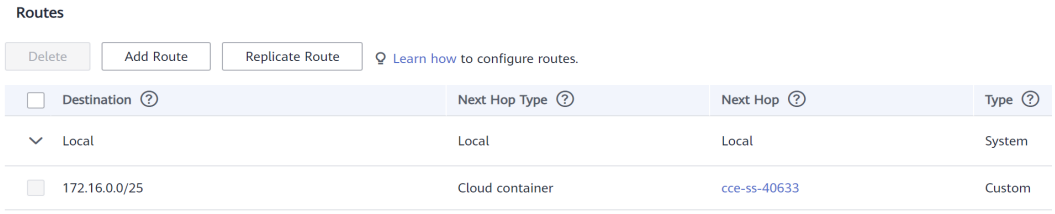
Example of VPC Network Access

In this example, a cluster using the VPC network model is created, and the cluster contains one node.

On the VPC console, locate the VPC to which the cluster belongs and check the VPC route table.

You can find that CCE has created a custom route in the route table. This route has a destination address corresponding to the container CIDR block assigned to the node, and the next hop is directed towards the target node. In the example, the container CIDR block for the cluster is 172.16.0.0/16, with 128 container IP addresses assigned to each node. Therefore, the node's container CIDR block is 172.16.0.0/25, providing a total of 128 container IP addresses.

Figure 7-24 Routes



When a container IP address is accessed, the VPC route will forward the traffic to the next-hop node that corresponds to the destination address. The following is an example:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a Deployment in the cluster.

Create the **deployment.yaml** file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
      imagePullSecrets:
        - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```

Step 3 Check the running pods.

```
kubectl get pod -owide
```

Command output:

```
NAME                       READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE
READINESS GATES
example-86b9779494-l8qrw   1/1     Running   0          14s   172.16.0.6   192.168.0.99   <none>
<none>
example-86b9779494-svs8t   1/1     Running   0          14s   172.16.0.7   192.168.0.99   <none>
<none>
example-86b9779494-x8kl5   1/1     Running   0          14s   172.16.0.5   192.168.0.99   <none>
<none>
example-86b9779494-zt627   1/1     Running   0          14s   172.16.0.8   192.168.0.99   <none>
<none>
```

Step 4 Use a cloud server in the same VPC to directly access a pod's IP address from outside the cluster. You can also access a pod using its IP address within the pod or from a node in the cluster. In the following example, access a pod's IP address within the pod. *example-86b9779494-l8qrw* is the pod name, and *172.16.0.7* is the pod IP address.


```
kubectl exec -it example-86b9779494-l8qrw -- curl 172.16.0.7
```

If the following information is displayed, the workload can be properly accessed:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

7.2.3.2 Adding a Container CIDR Block for a Cluster

Scenario

If the container CIDR block configured during CCE cluster creation cannot meet service expansion requirements, you can add a container CIDR block for the cluster.

Notes and Constraints

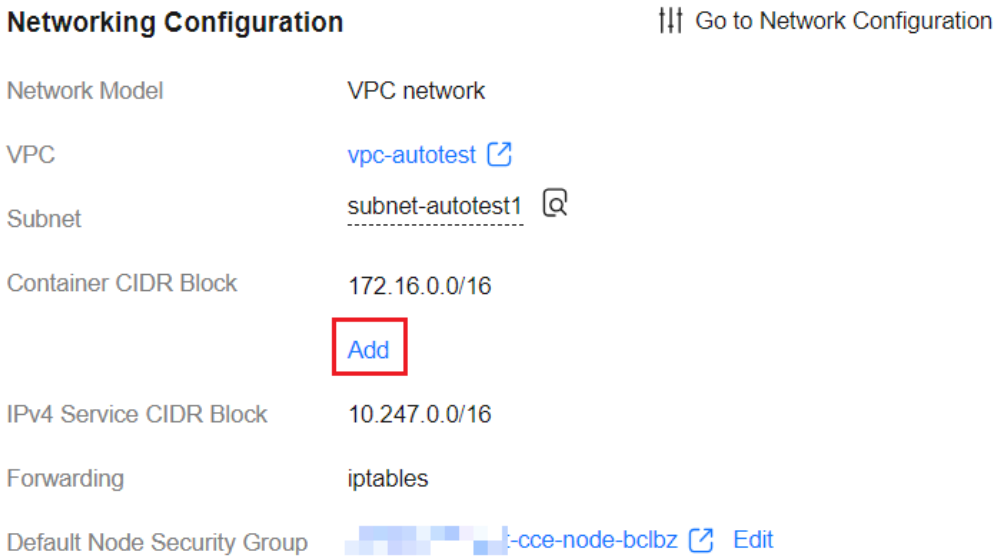
- This function is available only for clusters of v1.19 or later using a VPC network.
- An added container CIDR block cannot be deleted.

Adding a Container CIDR Block for a CCE Standard Cluster

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 On the **Overview** page, locate the **Networking Configuration** area and click **Add**.

Figure 7-25 Adding container CIDR block

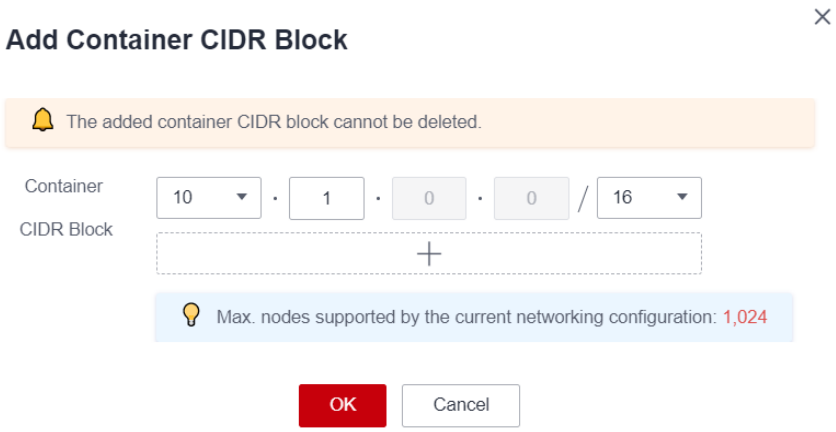


Step 3 Configure the container CIDR block to be added. You can click + to add multiple container CIDR blocks at a time.

NOTE

New container CIDR blocks cannot conflict with service CIDR blocks, VPC CIDR blocks, and existing container CIDR blocks.

Figure 7-26 Configuring the container CIDR block



Step 4 Click **OK**.

----End

7.2.4 Tunnel Network Settings

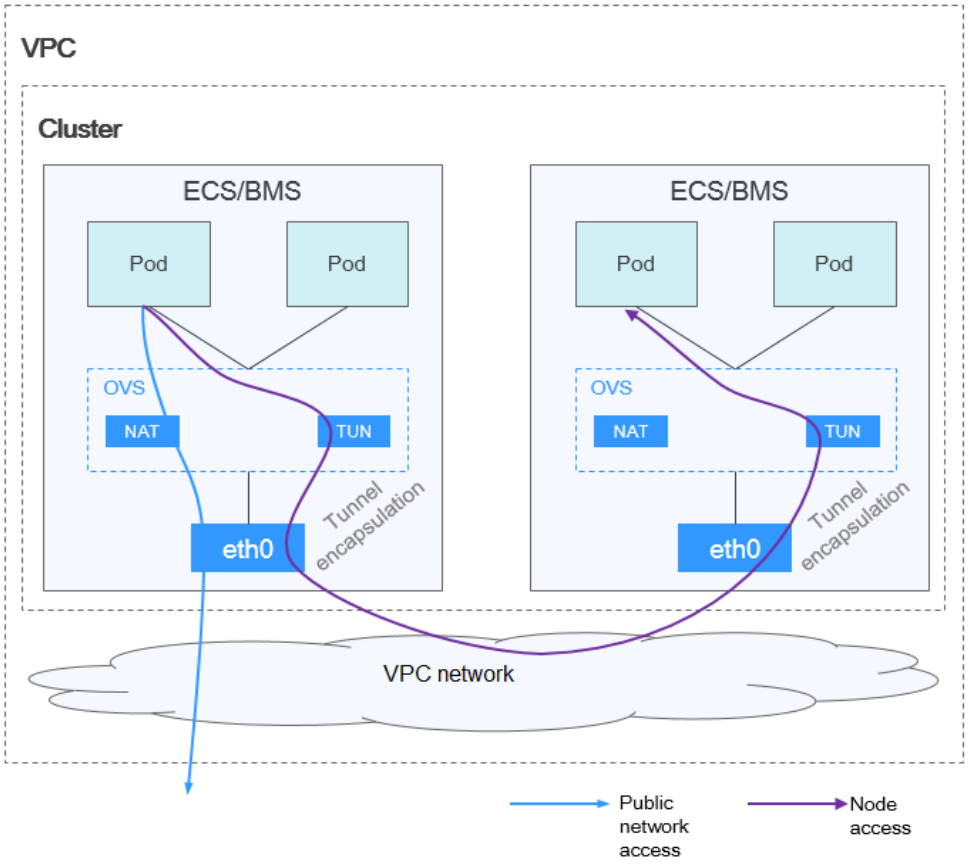
7.2.4.1 Tunnel Network Model

Model Definition

A container tunnel network creates a separate network plane for containers by using tunnel encapsulation on the host network plane. The container tunnel network of a CCE cluster uses VXLAN for tunnel encapsulation and Open vSwitch as the virtual switch backend. VXLAN is a protocol that encapsulates Ethernet packets into UDP packets to transmit them through tunnels. Open vSwitch is an open-source virtual switch software that provides functions such as network isolation and data forwarding.

While there may be some performance costs, packet encapsulation and tunnel transmission allow for greater interoperability and compatibility with advanced features, such as network policy-based isolation, in most common scenarios.

Figure 7-27 Container tunnel network



In a cluster using the container tunnel model, the communication paths between pods on the same node and between pods on different nodes are different.

- Inter-pod communication on the same node: Packets are directly forwarded via the OVS bridge on the node.
- Inter-pod communication on different nodes: Packets are encapsulated in the OVS bridge and then forwarded to the target pod on the peer node through the host NIC.

Advantages and Disadvantages

Advantages

- The container network is decoupled from the node network and is not limited by the VPC quotas and response speed (such as the number of VPC routes, number of ENIs, and creation speed).
- Network isolation is supported. For details, see **Configuring Network Policies to Restrict Pod Access**.
- Bandwidth limits are supported.
- Large-scale networking with a maximum of 2000 nodes is supported.

Disadvantages

- High encapsulation overhead results in poor performance and makes it difficult to locate network faults.
- Pods cannot directly use features like EIPs and security groups.
- Container IP addresses cannot be directly accessed by external networks.

Application Scenarios

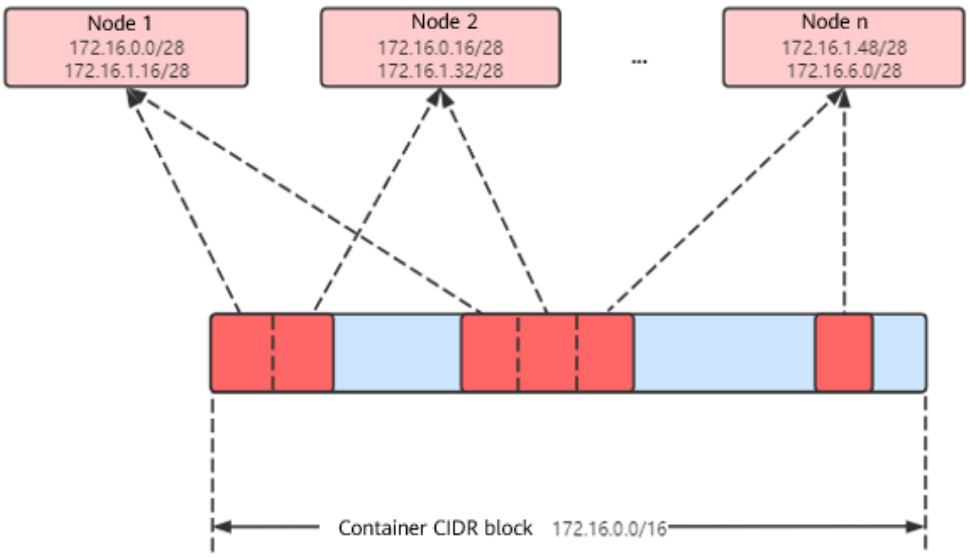
- Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high performance requirements, such as web applications, and middle-end and back-end services with a small number of access requests.
- Large-scale networking: Different from the VPC network that is limited by the VPC route quota, the container tunnel network does not have any restriction on the infrastructure. In addition, the container tunnel network controls the broadcast domain to the node level. The container tunnel network supports a maximum of 2000 nodes.

Container IP Address Management

The container tunnel network allocates container IP addresses according to the following rules:

- Container CIDR blocks are separate from node CIDR blocks.
- IP addresses are allocated by node. One or more CIDR blocks with a fixed size (16 by default) are allocated to each node in a cluster from the container CIDR block.
- When the IP addresses on a node are used up, you can apply for a new CIDR block.
- A container CIDR block assigns CIDR blocks to new nodes or existing nodes in a cyclical sequence.
- IP addresses from one or more CIDR blocks assigned to a node are allocated to pods scheduled to that node in a cyclical manner.

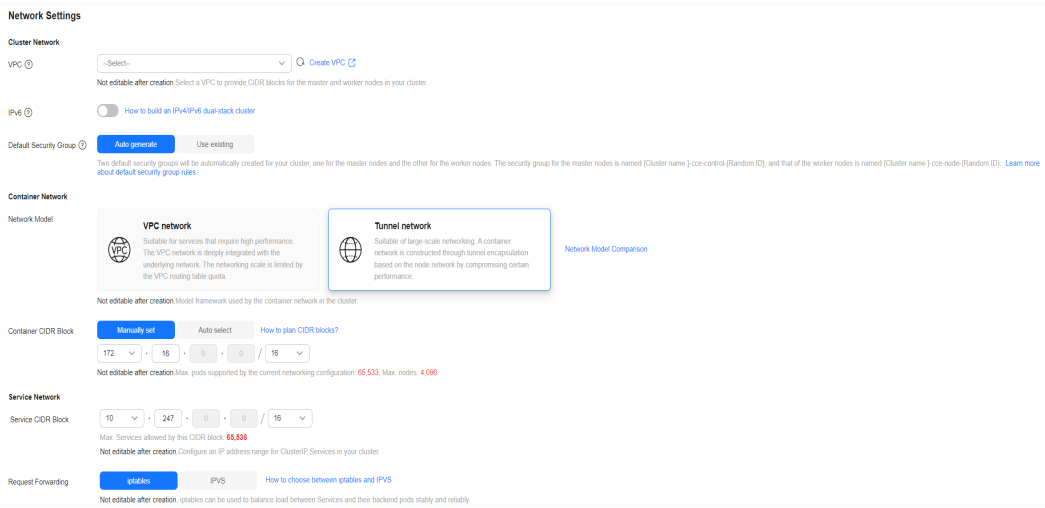
Figure 7-28 IP address allocation of the container tunnel network



Maximum number of nodes that can be created in the cluster using the container tunnel network = Number of IP addresses in the container CIDR block/Size of the IP CIDR block allocated to the node by the container CIDR block at a time (16 by default)

For example, if the container CIDR block is 172.16.0.0/16, the number of IP addresses is 65536. The mask of the container CIDR block allocated to a node is 28. That is, a total of 16 container IP addresses are allocated each time. Therefore, a maximum of 4096 (65536/16) nodes can be created. This is an extreme case. If 4096 nodes are created, a maximum of 16 pods can be created for each node because only a CIDR block with 16 IP addresses is allocated to each node. The number of nodes that can be added to a cluster is also determined by the available IP addresses in the node subnet and the scale of the cluster.

Figure 7-29 Selecting a network model (when creating the cluster)



Recommendation for CIDR Block Planning

As explained in **Cluster Network Structure**, network addresses in a cluster are divided into the cluster network, container network, and service network. When planning network addresses, consider the following factors:

- **The three CIDR blocks cannot overlap**. Otherwise, a conflict occurs. All subnets (including those created from the secondary CIDR blocks) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.
- Ensure that **each CIDR block has sufficient IP addresses**.
  - The IP addresses in the cluster CIDR block must match the cluster scale. Otherwise, nodes cannot be created due to insufficient IP addresses.
  - The IP addresses in the container CIDR block must match the service scale. Otherwise, pods cannot be created due to insufficient IP addresses. The number of pods that can be created on each node also depends on other parameter settings. For details, see **Maximum Number of Pods That Can Be Created on a Node**.

Example of Container Tunnel Network Access

The following is an example of creating a workload in a cluster using the container tunnel network model:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a Deployment in the cluster.

Create the **deployment.yaml** file. The following shows an example:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: example
  namespace: default
spec:
  replicas: 4
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: container-0
          image: 'nginx:perl'
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Create the workload.

```
kubectl apply -f deployment.yaml
```


Step 3 Check the running pods.

```
kubectl get pod -owide
```

Command output:

```
NAME                       READY   STATUS    RESTARTS   AGE     IP          NODE          NOMINATED NODE
READINESS GATES
example-5bdc5699b7-5rvq4   1/1     Running   0          3m28s   10.0.0.20   192.168.0.42   <none>
<none>
example-5bdc5699b7-984j9   1/1     Running   0          3m28s   10.0.0.21   192.168.0.42   <none>
<none>
example-5bdc5699b7-lfxkm   1/1     Running   0          3m28s   10.0.0.22   192.168.0.42   <none>
<none>
example-5bdc5699b7-wjcmg   1/1     Running   0          3m28s   10.0.0.52   192.168.0.64   <none>
<none>
```

Step 4 Use a cloud server in the same VPC to directly access a pod's IP address from outside the cluster. The access failed.

You can access a pod using its IP address within the pod or from a node in the cluster. In the following example, access a pod's IP address within the pod. *example-5bdc5699b7-5rvq4* is the pod name, and *10.0.0.21* is the pod IP address.

```
kubectl exec -it example-5bdc5699b7-5rvq4 -- curl 10.0.0.21
```

If the following information is displayed, the workload can be properly accessed:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

7.2.5 Pod Network Settings

7.2.5.1 Configuring hostNetwork for Pods

Scenario

Kubernetes allows pods to directly use the host/node network. When a pod is configured with **hostNetwork: true**, applications running in the pod can directly view the network interface of the host where the pod is located.

Configuration

Add **hostNetwork: true** to the pod definition.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      hostNetwork: true
      containers:
      - image: nginx:alpine
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

The configuration succeeds if the pod IP is the same as the node IP.

```
$ kubectl get pod -owide
NAME                READY   STATUS    RESTARTS   AGE     IP          NODE        NOMINATED NODE
READINESS GATES
nginx-6fdf99c8b-6wwft   1/1     Running   0          3m41s   10.1.0.55   10.1.0.55   <none>          <none>
```

Precautions

When a pod uses a host network, it uses a host port and its IP address is the same as the host's IP address. Before using a host network, verify that the pod port does not conflict with any service port on the host. Use a host network only if a workload pod must access a specific host port.

When using the host network, you access a pod on a node through a node port. Therefore, **allow access from the security group port of the node**. Otherwise, the access fails.

In addition, using the host network requires you to reserve host ports for the pods. When using a Deployment to deploy hostNetwork pods, ensure that **the number of pods does not exceed the number of nodes**. Otherwise, multiple pods will be scheduled onto the node, and they will fail to start due to port conflicts. For example, in the preceding example nginx YAML, if two pods (setting **replicas** to **2**) are deployed in a cluster with only one node, one pod cannot be created. The pod logs will show that the Nginx cannot be started because the port is occupied.

CAUTION

Do not schedule multiple pods that use the host network on the same node. Otherwise, when a ClusterIP Service is created to access a pod, the cluster IP address cannot be accessed.

```
$ kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/2     2            1           67m
```



```
$ kubectl get pod
NAME                      READY   STATUS             RESTARTS   AGE
nginx-6fdf99c8b-6wwft   1/1     Running            0          67m
nginx-6fdf99c8b-rglm7   0/1     CrashLoopBackOff   13         44m
$ kubectl logs nginx-6fdf99c8b-rglm7
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: bind() to [::]:80 failed (98: Address in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address in use)
2022/05/11 07:18:11 [emerg] 1#1: still could not bind()
nginx: [emerg] still could not bind()
```

7.2.5.2 Configuring QoS for a Pod

Scenario

Bandwidth preemption occurs between different containers deployed on the same node, which may cause service jitter. You can configure bandwidth limitation for the pod to solve this problem.

Specifications

The following table lists the bandwidth limitation specifications of pods.

Specifications	Tunnel	VPC	Cloud Native Network 2.0	Cloud Native Network 2.0 + DataPlane V2
Supported cluster versions	All versions	Clusters of v1.19.10 and later	Clusters of v1.19.10 and later	v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, or later
Egress bandwidth limitation	Supported	Supported	Supported	Supported

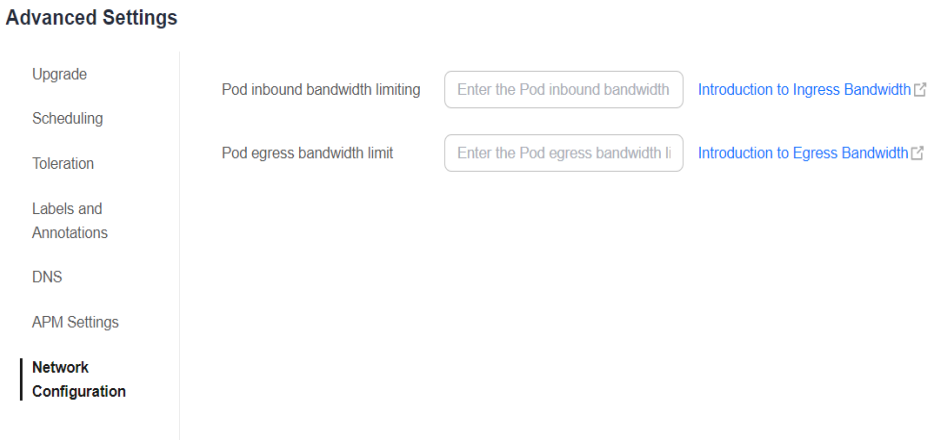
Specifications	Tunnel	VPC	Cloud Native Network 2.0	Cloud Native Network 2.0 + DataPlane V2
Ingress bandwidth limitation	Supported	Supported	Supported	Not supported
Scenarios where bandwidth limitation is not supported	None	None	• Pod access to cloud service CIDR blocks such as 100.125.0.0/16 • Pod health check	• Pod access to cloud service CIDR blocks such as 100.125.0.0/16 • Pod health check
Bandwidth limitation range	Only the rate limit in the unit of Mbit/s or Gbit/s is supported, for example, 100 Mbit/s and 1 Gbit/s. The minimum value is 1 Mbit/s and the maximum value is 4.29 Gbit/s.			The minimum value is 1 Kbit/s, and the maximum value is 1 Pbit/s.

- After DataPlane V2 network acceleration is enabled, pods on Huawei Cloud EulerOS 2.0 use Earliest Departure Time (EDT) to limit the egress bandwidth. The ingress bandwidth limitation is not supported. In other network modes, a Token Bucket Filter (TBF) qdisc is used to limit the bandwidth.
- Pod bandwidth limitation applies to regular containers (runC as the container runtime), not secure containers (Kata Containers as the container runtime).
- Pod bandwidth limitation does not apply to hostNetwork pods.

Using the CCE Console

When creating a workload on the console, you can set pod ingress and egress bandwidth limits by clicking **Network Configuration** in the **Advanced Settings** area.

Figure 7-30 Network settings



Using kubectl

You can add annotations to a workload to specify its egress and ingress bandwidth.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
  labels:
    app: test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
      annotations:
        kubernetes.io/ingress-bandwidth: 100M
        kubernetes.io/egress-bandwidth: 100M
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

- **kubernetes.io/ingress-bandwidth**: ingress bandwidth of the pod
- **kubernetes.io/egress-bandwidth**: egress bandwidth of the pod

If these two parameters are not specified, the bandwidth is not limited.

NOTE

After modifying the ingress or egress bandwidth limit of a pod, restart the container for the modification to take effect. After annotations are modified in a pod not managed by workloads, the container will not be restarted, so the bandwidth limits do not take effect. You can create a pod again or manually restart the container.

7.2.5.3 Configuring Network Policies to Restrict Pod Access

Network policies are designed by Kubernetes to restrict pod access. It is equivalent to a firewall at the application layer to enhance network security. The capabilities supported by network policies depend on the capabilities of the network add-ons of the cluster.

By default, if a namespace does not have any policy, pods in the namespace accept traffic from any source and send traffic to any destination.

The following selectors are available for network policies:

- **namespaceSelector**: selects particular namespaces for which all pods should be allowed as ingress sources or egress destinations.
- **podSelector**: selects particular pods in the same namespace as the network policy which should be allowed as ingress sources or egress destinations.
- **IPBlock**: selects particular IP blocks to allow as ingress sources or egress destinations.

Relationships Between Network Policies and Cluster Types

Cluster Type	**CCE Standard Cluster**	**CCE Turbo Cluster**
Network Model	**Tunnel**	**Cloud Native Network 2.0**
NetworkPolicy	Enabled by default	Disabled by default (To use network policies, enable DataPlane V2 when creating a cluster.)
Data plane implementation	OpenvSwitch	eBPF
Cluster version for inbound rules	All versions	v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, or later
Cluster version for outbound rules	v1.23 and later	
Selector for inbound rules	namespaceSelector podSelector	namespaceSelector podSelector IPBlock
Selector for outbound rules	namespaceSelector podSelector IPBlock	
Supported OS	EulerOS CentOS HCE 2.0	HCE 2.0
IPv6 network policies	Not supported	Supported
Secure containers	Not supported	Not supported

Cluster Type	CCE Standard Cluster	CCE Turbo Cluster
Network Model	**Tunnel**	**Cloud Native Network 2.0**
IPBlock scope	Not limited	Only IP blocks outside the cluster (The configuration for container CIDR blocks and node IP addresses does not take effect.)
Limit ClusterIP access through workload labels	Not supported	Supported
Limit the internal cloud server CIDR block of 100.125.0.0/16	Supported	Not supported
SCTP	Not supported	Not supported
Always allow access to pods on a node from other nodes	Supported	Supported
Configure EndPort in network policies	Not supported	Not supported

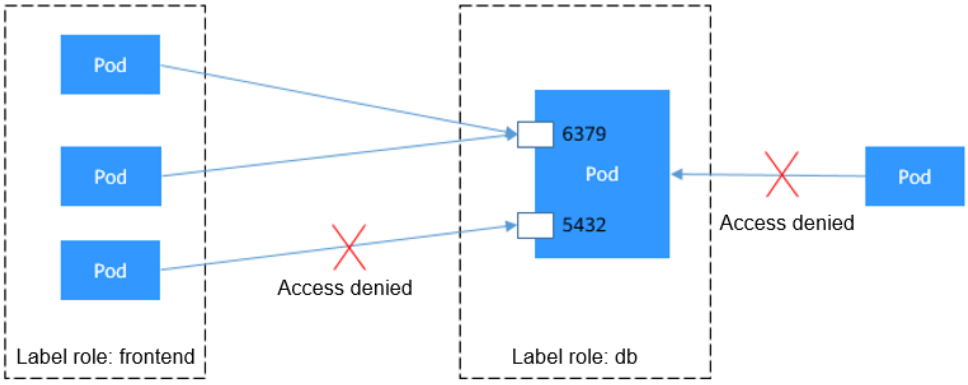
NOTE

- Secure containers (such as Kata as the container runtime) are not supported by network policies.
- If you upgrade a CCE standard cluster with a tunnel network to a version that supports egress rules in in-place mode, the rules will not work because the node OS is not upgraded. In this case, reset the node.
- When a network policy is enabled for a cluster that uses a tunnel network, the source IP address of a pod accessing the CIDR block of a Service will be recorded in the optional field of the reported IP address data. This enables the configuration of network policy rules on the destination pod, taking into account the source IP address of the pod.

Using Ingress Rules Through YAML

- **Scenario 1: Use a network policy to limit access to a pod to only pods with specific labels.**

Figure 7-31 podSelector



The pod labeled with **role=db** only permits access to its port 6379 from pods labeled with **role=frontend**. To do so, perform the following operations:

a. Create the **access-demo1.yaml** file.

```
vim access-demo1.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo1
  namespace: default
spec:
  podSelector:                # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:                    # This is an ingress rule.
  - from:
    - podSelector:            # Only allow the access of the pods labeled with role=frontend.
        matchLabels:
          role: frontend
    ports:                    # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

b. Run the following command to create the network policy based on the **access-demo1.yaml** file:

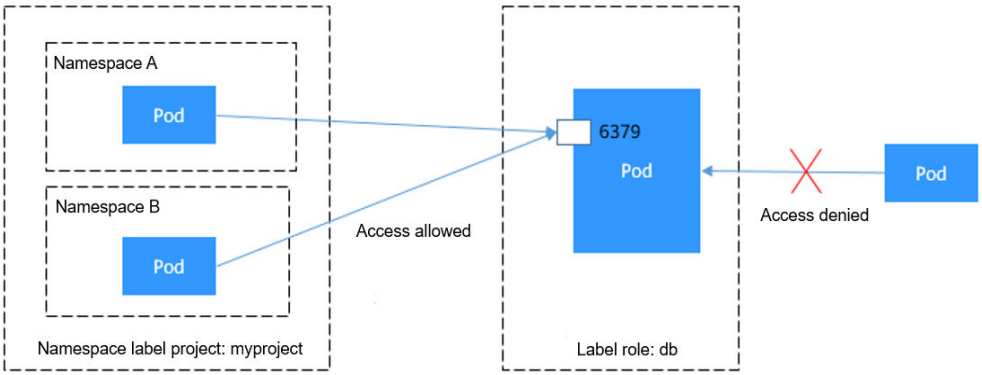
```
kubectl apply -f access-demo1.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo1 created
```

- **Scenario 2: Use a network policy to limit access to a pod to only pods in a specific namespace.**

Figure 7-32 namespaceSelector



The pod labeled with **role=db** only permits access to its port 6379 from pods in the namespace labeled with **project=myproject**. To do so, perform the following operations:

a. Create the **access-demo2.yaml** file.

```
vim access-demo2.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo2
spec:
  podSelector:                # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:                    # This is an ingress rule.
  - from:
    - namespaceSelector:      # Only allow the access of the pods in the namespace labeled with project=myproject.
        matchLabels:
          project: myproject
    ports:                    # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

b. Run the following command to create the network policy based on the **access-demo2.yaml** file:

```
kubectl apply -f access-demo2.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo2 created
```

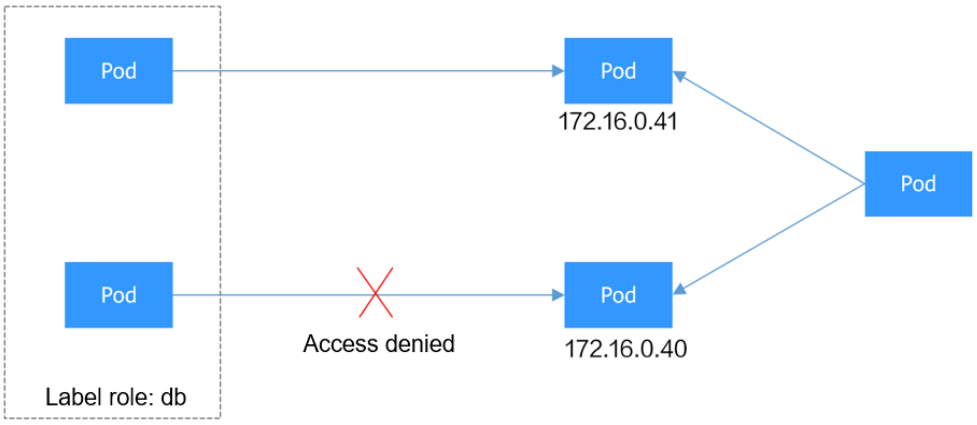
Using Egress Rules Through YAML

NOTE

The clusters of v1.23 or later using a tunnel network support egress rules.

- **Scenario 1: Use a network policy to limit a pod's access to specific addresses.**

Figure 7-33 IPBlock



The pod labeled with **role=db** only permits access to the 172.16.0.16/16 CIDR block, excluding 172.16.0.40/32 within it. To do so, perform the following operations:

a. Create the **access-demo3.yaml** file.

```
vim access-demo2.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo3
  namespace: default
spec:
  policyTypes:                  # Must be specified for an egress rule.
    - Egress
  podSelector:                  # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  egress:                       # Egress rule
  - to:
    - ipBlock:
        cidr: 172.16.0.16/16    # Allow access to this CIDR block in the outbound direction.
        except:
        - 172.16.0.40/32        # Block access to this address in the CIDR block.
```

b. Run the following command to create the network policy based on the **access-demo3.yaml** file:

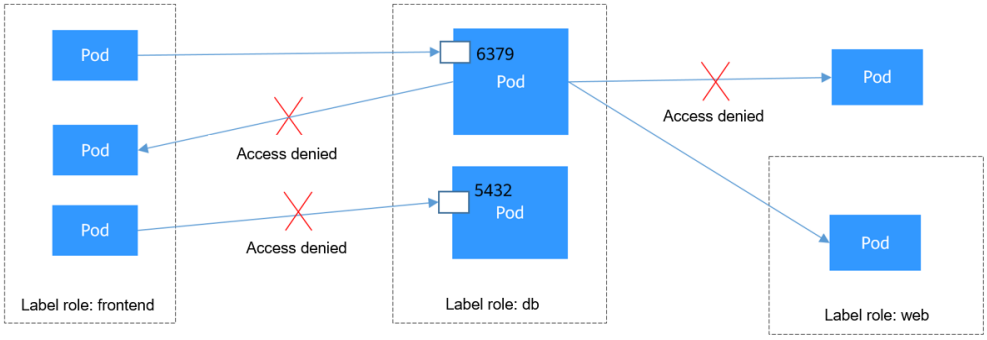
```
kubectl apply -f access-demo3.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo3 created
```

- **Scenario 2: Use a network policy to limit access to a pod to only pods with specific labels and this pod can only access specific pods.**

Figure 7-34 Using both ingress and egress



The pod labeled with **role=db** only permits access to its port 6379 from pods labeled with **role=frontend**, and this pod can only access the pods labeled with **role=web**. You can use the same rule to configure both ingress and egress in a network policy. To do so, perform the following operations:

a. Create the **access-demo4.yaml** file.

```
vim access-demo2.yaml
```

File content:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-demo4
  namespace: default
spec:
  policyTypes:
  - Ingress
  - Egress
  podSelector:                # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:                    # This is an ingress rule.
  - from:
    - podSelector:            # Only allow the access of the pods labeled with role=frontend.
        matchLabels:
          role: frontend
    ports:                    # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
  egress:                     # Egress rule
  - to:
    - podSelector:            # Only pods with the role=web label can be accessed.
        matchLabels:
          role: web
```

b. Run the following command to create the network policy based on the **access-demo4.yaml** file:

```
kubectl apply -f access-demo4.yaml
```

Expected output:

```
networkpolicy.networking.k8s.io/access-demo4 created
```

Creating a Network Policy on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Policies** in the navigation pane, click the **Network Policies** tab, and click **Create Network Policy** in the upper right corner.

- **Policy Name**: Specify a network policy name.
- **Namespace**: Select a namespace in which the network policy is applied.
- **Selector**: Enter a label, select the pod to be associated, and click **Add**. You can also click **Reference Workload Label** to use the label of an existing workload.
- **Inbound Rule**: Click + to add an inbound rule. For details about parameter settings, see **Table 7-23**.

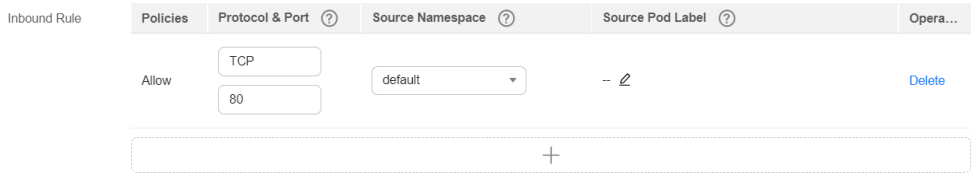


Table 7-23 Adding an inbound rule

Parameter	Description
Protocol & Port	Select the protocol type and port. Currently, TCP and UDP are supported.
Source Namespace	Select a namespace whose objects can be accessed. If this parameter is not specified, the object belongs to the same namespace as the current policy.
Source Pod Label	Allow accessing the pods with this label. If this parameter is not specified, all pods in the namespace can be accessed.

- **Outbound Rule**: Click + to add an outbound rule. For details about parameter settings, see **Table 7-24**.

Table 7-24 Adding an outbound rule

Parameter	Description
Protocol & Port	Select the protocol type and port. Currently, TCP and UDP are supported. If this parameter is not specified, the protocol type is not limited.

Parameter	Description
Destination CIDR Block	Allows requests to be routed to a specified CIDR block (and not to the exception CIDR blocks). Separate the destination and exception CIDR blocks using a vertical bar (|). If there are multiple exception CIDR blocks, separate them using commas (,). For example, 172.17.0.0/16|172.17.1.0/24,172.17.2.0/24 indicates that 172.17.0.0/16 is accessible, but not for 172.17.1.0/24 and 172.17.2.0/24.
Destination Namespace	Select a namespace whose objects can be accessed. If this parameter is not specified, the object belongs to the same namespace as the current policy.
Destination Pod Label	Allow accessing the pods with this label. If this parameter is not specified, all pods in the namespace can be accessed.

Step 3 Click **OK**.

----End

7.2.5.4 DataPlane V2 Network Acceleration

DataPlane V2 can be enabled for clusters that use a Cloud Native Network 2.0. After this function is enabled, eBPF redirection will be enabled for higher performance.

DataPlane V2	Description
Technical implementation	DataPlane V2 integrates open-source **Cilium** to provide capabilities such as Service network acceleration and network policies.
Supported cluster versions	v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, or later
Usage	• When creating a CCE Turbo cluster, select Cloud Native Network 2.0 and enable **DataPlane V2**. **NOTICE** • After DataPlane V2 is enabled, secure containers (Kata Containers as the container runtime) are not supported. • Enabled DataPlane V2 cannot be disabled. • DataPlane V2 can only be enabled for new clusters. • After DataPlane V2 is enabled, the **Guaranteed Egress Network Bandwidth** capability of cloud native hybrid deployment cannot be enabled.
Supported OS	Huawei Cloud EulerOS 2.0

DataPlane V2	Description
Accelerated data link	When a pod or node accesses the cluster IP address and external IP address of a Service, eBPF is used to directly resolve the Service address to the address of a pod at the Service backend.
Performance optimization	• Pods or nodes use eBPF to directly access Services, eliminating the need for iptables or IPVS-based forwarding on the nodes. This greatly reduces request latency. The network performance is less affected in large-scale clusters, and the scalability is better. • EDT is used to limit the egress bandwidth. This makes bandwidth limitation more accurate and resource consumption lower.
Bandwidth	After DataPlane V2 network acceleration is enabled, pods on Huawei Cloud EulerOS 2.0 use EDT to limit the egress bandwidth. The ingress bandwidth limitation is not supported. In other network modes, a TBF qdisc is used to limit the bandwidth. For details, see **Configuring QoS for a Pod**.
NetworkPolicy	• The implementation of network policies is different from that of container tunnel networks. For details, see **Configuring Network Policies to Restrict Pod Access**. – The IPBlock selector can only select CIDR blocks outside a cluster. – The IPBlock selector does not have good support for the **except** keyword, so this keyword is not recommended. – If a network policy of the egress type is used, the pod fails to access the IP addresses of the hostNetwork pod and node in the cluster. • The **CiliumNetworkPolicy** and **CiliumClusterwideNetworkPolicy** APIs do not support **Node Selector**, **DNS policies**, or **L7 network policies**.
Resource consumption	The resident cilium-agent process on each node is responsible for eBPF network acceleration. Each cilium-agent process may occupy 80 MiB of memory. Each time a pod is added, the cilium-agent memory consumption may increase by 10 KiB.

Components

After DataPlane V2 is enabled, components listed in the following table are installed.

Component	Description	Resource Type
cilium-operator	• Synchronizes CRDs. • Removes the **node.cilium.io/agent-not-ready** taint of a node. • Tunes and recycles internal resources.	Deployment
yangtse-cilium	• Installs the auxiliary CNI (cilium-cni) for CCE to adapt to Cilium. • Deploys cilium-agent.	DaemonSet

Change History

Add-on Version	Cluster Version	New Feature	Community Version
1.08	v1.27 v1.28 v1.29 v1.30 v1.31	• Added Cloud Native Network 2.0 for CCE Turbo clusters. • Disabled host-based firewalls (by setting **enable-host-firewall=false**). • Disabled L7 network policies (by setting **enable-l7-proxy=false**).	**v1.14.1**

7.3 Service

7.3.1 Overview

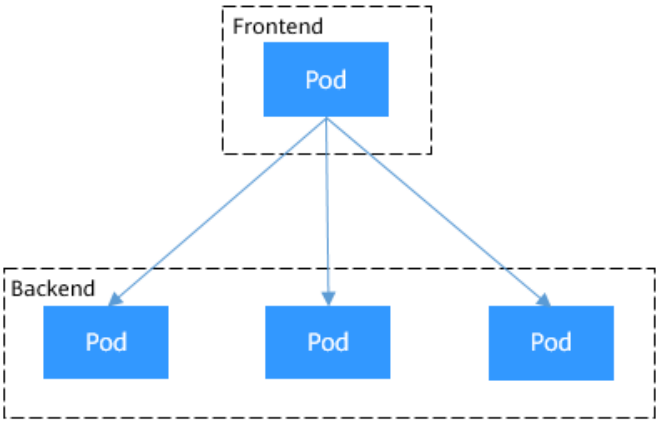
Direct Access to a Pod

After a pod is created, the following problems may occur if you directly access the pod:

- The pod can be deleted and recreated at any time by a controller such as a Deployment, and the result of accessing the pod becomes unpredictable.
- The IP address of the pod is allocated only after the pod is started. Before the pod is started, the IP address of the pod is unknown.
- An application is usually composed of multiple pods that run the same image. Accessing pods one by one is not efficient.

For example, an application uses Deployments to create the frontend and backend. The frontend calls the backend for computing, as shown in **Figure 7-35**. Three pods are running in the backend, which are independent and replaceable. When a backend pod is re-created, the new pod is assigned with a new IP address, of which the frontend pod is unaware.

Figure 7-35 Inter-pod access

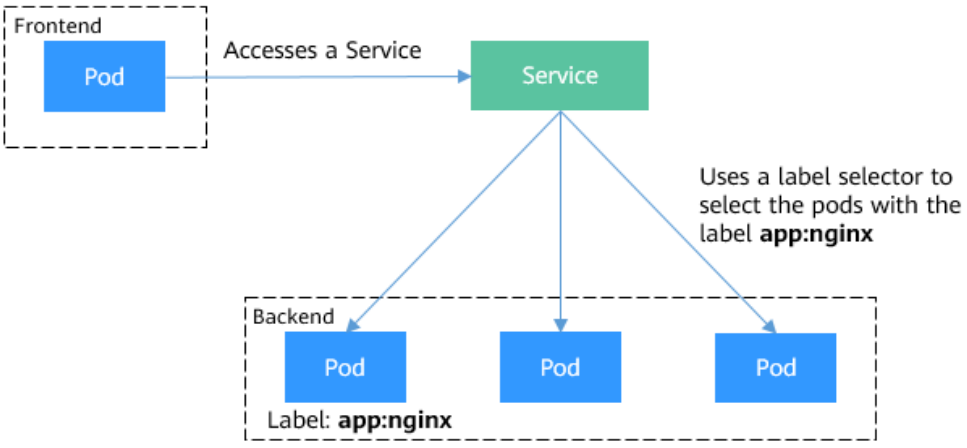


Using Services for Pod Access

Kubernetes Services are used to solve the preceding pod access problems. A Service has a fixed IP address. (When a CCE cluster is created, a Service CIDR block is set, which is used to allocate IP addresses to Services.) A Service forwards requests accessing the Service to pods based on labels, and at the same time, perform load balancing for these pods.

In the preceding example, a Service is added for the frontend pod to access the backend pods. In this way, the frontend pod does not need to be aware of the changes on backend pods, as shown in **Figure 7-36**.

Figure 7-36 Accessing pods through a Service



Service Types

Kubernetes allows you to specify a Service of a required type. The values and actions of different types of Services are as follows:

- **ClusterIP**

  ClusterIP Services allow workloads in the same cluster to use their cluster-internal domain names to access each other.

- **NodePort**

  A Service is exposed on each node's IP address at a static port (NodePort). A ClusterIP Service, to which the NodePort Service will route, is automatically created. By requesting <NodeIP>:<NodePort>, you can access a NodePort Service from outside the cluster.

- **LoadBalancer**

  LoadBalancer Services can access workloads from the public network through a load balancer, which is more reliable than EIP-based access. LoadBalancer Services are recommended for accessing workloads from outside the cluster.

- **DNAT**

  A DNAT gateway translates addresses for cluster nodes and allows multiple cluster nodes to share an EIP. DNAT Services provide higher reliability than EIP-based NodePort Services. You do not need to bind an EIP to a single node and requests can still be distributed to the workload even any of the nodes inside is down.

externalTrafficPolicy (Service Affinity)

For a NodePort and LoadBalancer Service, requests are first sent to the node port, then the Service, and finally the pod backing the Service. The backing pod may be not located in the node receiving the requests. By default, the backend workload can be accessed from any node IP address and service port. If the pod is not on the node that receives the request, the request will be redirected to the node where the pod is located, which may cause performance loss.

The **externalTrafficPolicy** parameter in a Service is used to determine whether the external traffic can be routed to the local nodes or cluster-wide endpoints. The following is an example:

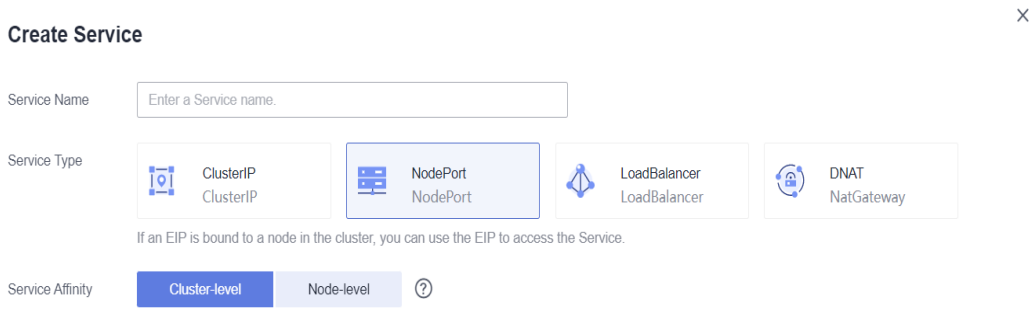
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
```

If the value of **externalTrafficPolicy** is **Local**, requests sent from *Node IP address:Service port* will be forwarded only to the pod on the local node. If the node does not have a pod, the requests are suspended.

If the value of **externalTrafficPolicy** is **Cluster**, requests are forwarded within the cluster and the backend workload can be accessed from any node IP address and service port.

If **externalTrafficPolicy** is not set, the default value **Cluster** will be used.

When creating a NodePort on the CCE console, you can configure this parameter using the **Service Affinity** option.



The following table compares the two options of **externalTrafficPolicy**.

Table 7-25 Comparison of the two types of service affinity

Dimension	externalTrafficPolicy (Service Affinity)	
	Cluster-level (Cluster)	**Node-level (Local)**
Application scenario	This mode applies to scenarios where high performance is not required and the source IP address of the client does not need to be retained. This mode brings more balanced load to each node in the cluster.	This mode applies to scenarios where high performance is required and the source IP address of the client needs to be retained. However, traffic is forwarded only to the node where the container resides, and source IP address translation is not performed.
Access mode	The IP addresses and access ports of all nodes in a cluster can access the workload associated with the Service.	Only the IP address and access port of the node where the workload is located can access the workload associated with the Service.
Obtaining client source IP address	The source IP address of the client cannot be obtained.	The source IP address of the client can be obtained.
Access performance	Service access will cause performance loss due to route redirection, and the next hop for a data packet may be another node.	Service access will not cause performance loss due to route redirection.
Load balancing	Traffic propagation has good overall load balancing.	There is a potential risk of unbalanced traffic propagation.

Dimension	externalTrafficPolicy (Service Affinity)	
	Cluster-level (Cluster)	Node-level (Local)
Other special case	None	In different container network models and service forwarding modes, accessing Services from within the cluster may fail. For details, see **Why a Service Fail to Be Accessed from Within the Cluster**.

Why a Service Fail to Be Accessed from Within the Cluster

If the service affinity of a Service is set to the node level, that is, the value of **externalTrafficPolicy** is **Local**, the Service may fail to be accessed from within the cluster (specifically, nodes or containers). Information similar to the following is displayed:

```
upstream connect error or disconnect/reset before headers. reset reason: connection failure
Or
curl: (7) Failed to connect to 192.168.10.36 port 900: Connection refused
```

It is common that a load balancer in a cluster cannot be accessed. The reason is as follows: When Kubernetes creates a Service, kube-proxy adds the access address of the load balancer as an external IP address (External-IP, as shown in the following command output) to iptables or IPVS. If a client inside the cluster initiates a request to access the load balancer, the address is considered as the external IP address of the Service, and the request is directly forwarded by kube-proxy without passing through the load balancer outside the cluster.

```
# kubectl get svc nginx
NAME    TYPE          CLUSTER-IP      EXTERNAL-IP                PORT(S)       AGE
nginx   LoadBalancer  10.247.76.156   123.**.**.**,192.168.0.133  80:32146/TCP  37s
```

When the value of **externalTrafficPolicy** is **Local**, the access failures in different container network models and service forwarding modes are as follows:

NOTE

- For a multi-pod workload, ensure that all pods are accessible. Otherwise, there is a possibility that the access to the workload fails.
- In a CCE Turbo cluster that utilizes Cloud Native Network 2.0, node-level affinity is supported only when the Service backend is connected to a hostNetwork pod.
- The table lists only the scenarios where the access may fail. Other scenarios that are not listed in the table indicate that the access is normal.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
NodePort Service	Public/ Private network	Same node as the service pod	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Different nodes from the service pod	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	The access is successful.	The access is successful.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on different nodes from the service pod	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.	Access the IP address and NodePort on the node where the server is located: The access is successful. Access the IP address and NodePort on a node other than the node where the server is located: The access failed.
LoadBalancer Service using a dedicated load balancer	Private network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
DNAT gateway Service	Public network	Same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on the same node as the service pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on different nodes from the service pod	The access failed.	The access failed.	The access failed.	The access failed.

Service Type Released on the Server	Access Type	Request Initiation Location on the Client	Tunnel Network Cluster (IPVS)	VPC Network Cluster (IPVS)	Tunnel Network Cluster (iptables)	VPC Network Cluster (iptables)
LoadBalancer Service using a Dedicated load balancer (Local) for interconnection with NGINX Ingress Controller	Private network	Same node as cceaddon-nginx-ingress-controller pod	The access failed.	The access failed.	The access failed.	The access failed.
		Other containers on the same node as the cceaddon-nginx-ingress-controller pod	The access failed.	The access failed.	The access failed.	The access failed.

The following methods can be used to solve this problem:

- (**Recommended**) In the cluster, use the ClusterIP Service or service domain name for access.
- Set **externalTrafficPolicy** of the Service to **Cluster**, which means cluster-level service affinity. Note that this affects source address persistence.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","
eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Leveraging the pass-through feature of the Service, kube-proxy is bypassed when the ELB address is used for access. The ELB load balancer is accessed

first, and then the workload. For details, see **Configuring Passthrough Networking for a LoadBalancer Service**.

NOTE

- In a CCE standard cluster, after passthrough networking is configured for a dedicated load balancer, the private IP address of the load balancer cannot be accessed from the node where the workload pod resides or other containers on the same node as the workload.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the passthrough settings of Services connected to the same load balancer must be the same.
- If node-level (local) service affinity is used, **kubernetes.io/elb.pass-through** is automatically set to **onlyLocal** to enable pass-through.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.pass-through: "true"
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","
eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

7.3.2 ClusterIP

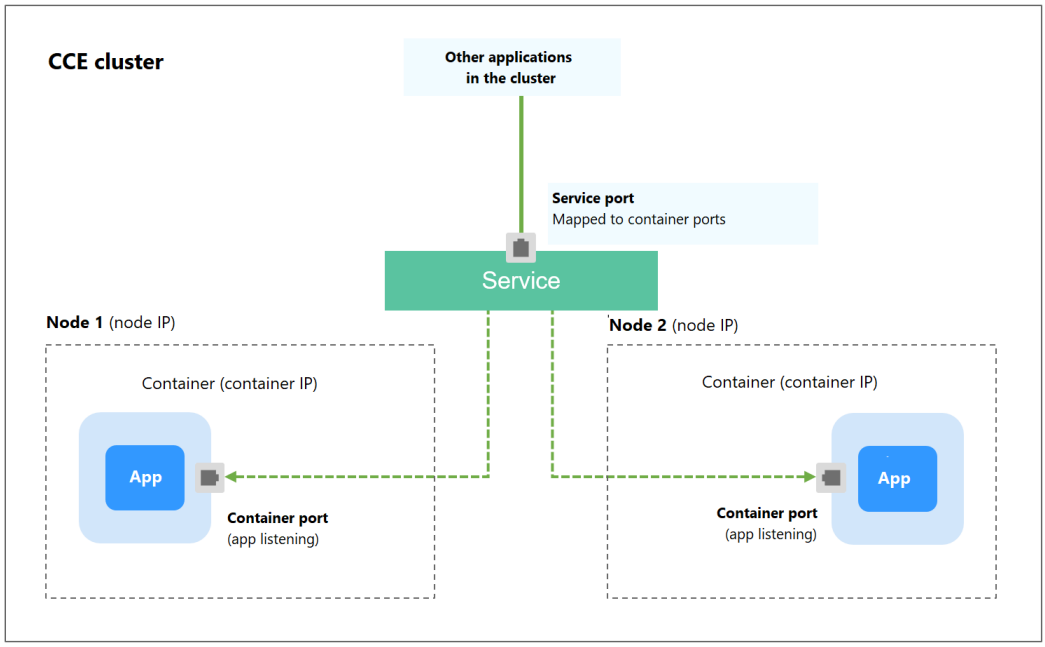
Scenario

ClusterIP Services allow workloads in the same cluster to use their cluster-internal domain names to access each other.

The cluster-internal domain name format is *<Service name>.<Namespace of the workload>***.svc.cluster.local:***<Port>*, for example, **nginx.default.svc.cluster.local:80**.

Figure 7-37 shows the mapping relationships between access channels, container ports, and access ports.

Figure 7-37 Intra-cluster access (ClusterIP)



Creating a ClusterIP Service

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure intra-cluster access parameters.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **ClusterIP**.
- **Namespace**: namespace that the workload belongs to.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Protocol Version**: Select the IP address of different versions based on service requirements. For details, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**. **This parameter is available only in clusters of v1.15 or later with IPv6 enabled (set during cluster creation).**
- **Ports**
  - **Protocol**: protocol used by the Service.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.

Step 4 Click **OK**.

----End

Setting the Access Type Using kubectl

You can configure Service access using kubectl. This section uses an Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-clusterip-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-clusterip-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-clusterip-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-clusterip
spec:
  ports:
  - name: service0
    port: 8080              # Port for accessing a Service
    protocol: TCP           # Protocol used for accessing a Service. The value can be TCP or UDP.
    targetPort: 80          # Port used by a Service to access the target container. This port is closely related to the applications running in a container. In this example, the Nginx image uses port 80 by default.
  selector:                 # Label selector. A Service selects a pod based on the label and forwards the requests for accessing the Service to the pod. In this example, select the pod with the app:nginx label.
    app: nginx
  type: ClusterIP           # Type of a Service. ClusterIP indicates that a Service is only reachable from within the cluster.
```

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload has been created:

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running:


```
NAME                     READY     STATUS            RESTARTS   AGE
nginx-2601814895-znhbr   1/1       Running           0          15s
```

Step 4 Create a Service.

kubectl create -f nginx-clusterip-svc.yaml

If information similar to the following is displayed, the Service is being created:

```
service "nginx-clusterip" created
```

kubectl get svc

If information similar to the following is displayed, the Service has been created, and a cluster-internal IP address has been assigned to the Service.

```
# kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes        ClusterIP   10.247.0.1     <none>        443/TCP    4d6h
nginx-clusterip   ClusterIP   10.247.74.52   <none>        8080/TCP   14m
```

Step 5 Access the Service.

A Service can be accessed from containers or nodes in a cluster.

Create a pod, access the pod, and run the **curl** command to access *IP address:Port* or the domain name of the Service, as shown in the following figure.

The domain name suffix can be omitted. In the same namespace, you can directly use **nginx-clusterip:8080** for access. In other namespaces, you can use **nginx-clusterip.default:8080** for access.

```
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.247.74.52:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # curl nginx-clusterip.default.svc.cluster.local:8080
...
<h1>Welcome to nginx!</h1>
...
/ # curl nginx-clusterip.default:8080
...
<h1>Welcome to nginx!</h1>
...
```



```
/ # curl nginx-clusterip:8080
...
<h1>Welcome to nginx!</h1>
...
```

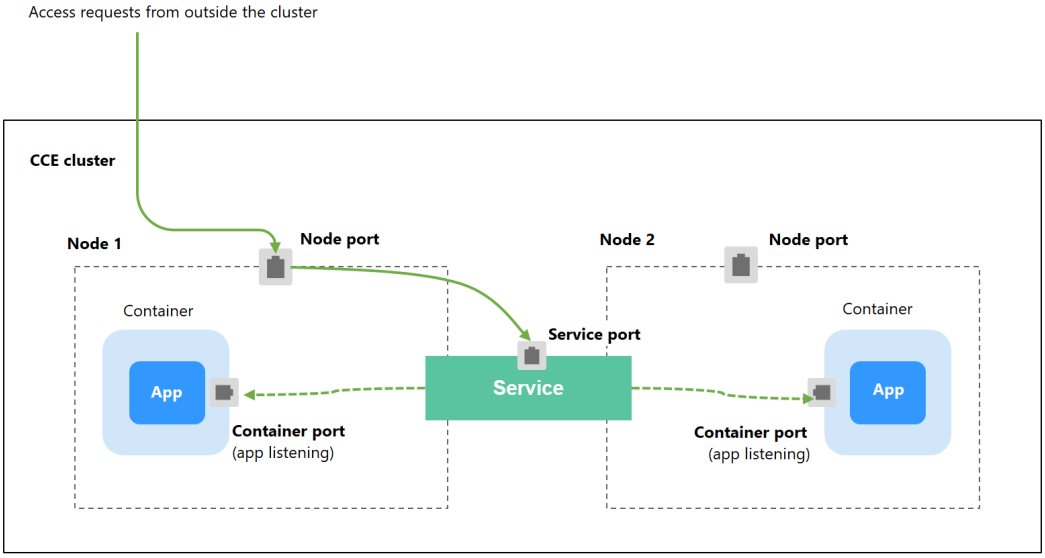
----End

7.3.3 NodePort

Scenario

A Service is exposed on each node's IP address at a static port (NodePort). When you create a NodePort Service, Kubernetes automatically allocates an internal IP address (ClusterIP) of the cluster. When clients outside the cluster access <NodeIP>:<NodePort>, the traffic will be forwarded to the target pod through the ClusterIP of the NodePort Service.

Figure 7-38 NodePort access



Notes and Constraints

- By default, a NodePort Service is accessed within a VPC. To use an EIP to access a NodePort Service through public networks, bind an EIP to the node in the cluster in advance.
- After a Service is created, if the affinity setting is switched from the cluster level to the node level, the connection tracing table will not be cleared. Do not modify the Service affinity setting after the Service is created. To modify it, create a Service again.
- In a CCE Turbo cluster, node-level affinity is supported only when the Service backend is connected to a hostNetwork pod.
- In VPC network mode, when container A is published through a NodePort service and the service affinity is set to the node level (that is, **externalTrafficPolicy** is set to **local**), container B deployed on the same node cannot access container A through the node IP address and NodePort service.
- When a NodePort service is created in a cluster of v1.21.7 or later, the port on the node is not displayed using **netstat** by default. If the cluster forwarding

mode is **iptables**, run the **iptables -t nat -L** command to view the port. If the cluster forwarding mode is **IPVS**, run the **ipvsadm -Ln** command to view the port.

Creating a NodePort Service

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure intra-cluster access parameters.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **NodePort**.
- **Namespace**: namespace that the workload belongs to.
- **Service Affinity**: For details, see **externalTrafficPolicy (Service Affinity)**.
  - **Cluster level**: The IP addresses and access ports of all nodes in a cluster can access the workload associated with the Service. Service access will cause performance loss due to route redirection, and the source IP address of the client cannot be obtained.
  - **Node level**: Only the IP address and access port of the node where the workload is located can access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **IPv6**: This function is disabled by default. After this function is enabled, the cluster IP address of the Service changes to an IPv6 address. For details, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**. **This parameter is available only in clusters of v1.15 or later with IPv6 enabled (set during cluster creation).**
- **Ports**
  - **Protocol**: protocol used by the Service.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Node Port**: You are advised to select **Auto**. You can also specify a port. The default port ranges from 30000 to 32767.

Step 4 Click **OK**.

----End

Using kubectl

You can configure Service access using kubectl. This section uses an Nginx workload as an example to describe how to configure a NodePort Service using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-nodeport-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-nodeport-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-nodeport-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-nodeport
spec:
  ports:
  - name: service
    nodePort: 30000     # Node port. The value ranges from 30000 to 32767.
    port: 8080          # Port for accessing a Service
    protocol: TCP       # Protocol used for accessing a Service. The value can be TCP or UDP.
    targetPort: 80      # Port used by a Service to access the target container. This port is closely related to the applications running in a container. In this example, the Nginx image uses port 80 by default.
  selector:             # Label selector. A Service selects a pod based on the label and forwards the requests for accessing the Service to the pod. In this example, select the pod with the app:nginx label.
    app: nginx
  type: NodePort        # Service type. NodePort indicates that the Service is accessed through a node port.
```

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload has been created:

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.


```
NAME                     READY     STATUS            RESTARTS   AGE
nginx-2601814895-qhxqv   1/1       Running           0          9s
```

Step 4 Create a Service.

kubectl create -f nginx-nodeport-svc.yaml

If information similar to the following is displayed, the Service is being created:

```
service "nginx-nodeport" created
```

kubectl get svc

If information similar to the following is displayed, the Service has been created:

```
# kubectl get svc
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes       ClusterIP   10.247.0.1     <none>        443/TCP          4d8h
nginx-nodeport   NodePort    10.247.30.40   <none>        8080:30000/TCP   18s
```

Step 5 Access the Service.

By default, a NodePort Service can be accessed by using *Any node IP address:Node port*.

The Service can be accessed from a node in another cluster in the same VPC or in another pod in the cluster. If a public IP address is bound to the node, you can also use the public IP address to access the Service. Create a container in the cluster and access the container by using *Node IP address:Node port*.

```
# kubectl get node -owide
NAME          STATUS   ROLES    AGE    INTERNAL-IP    EXTERNAL-IP   OS-IMAGE               KERNEL-
VERSION               CONTAINER-RUNTIME
10.100.0.136   Ready    <none>   152m   10.100.0.136   <none>        CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64   docker://18.9.0
10.100.0.5     Ready    <none>   152m   10.100.0.5     <none>        CentOS Linux 7 (Core)
3.10.0-1160.25.1.el7.x86_64   docker://18.9.0
# kubectl run -i --tty --image nginx:alpine test --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # curl 10.100.0.136:30000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

----End

7.3.4 LoadBalancer

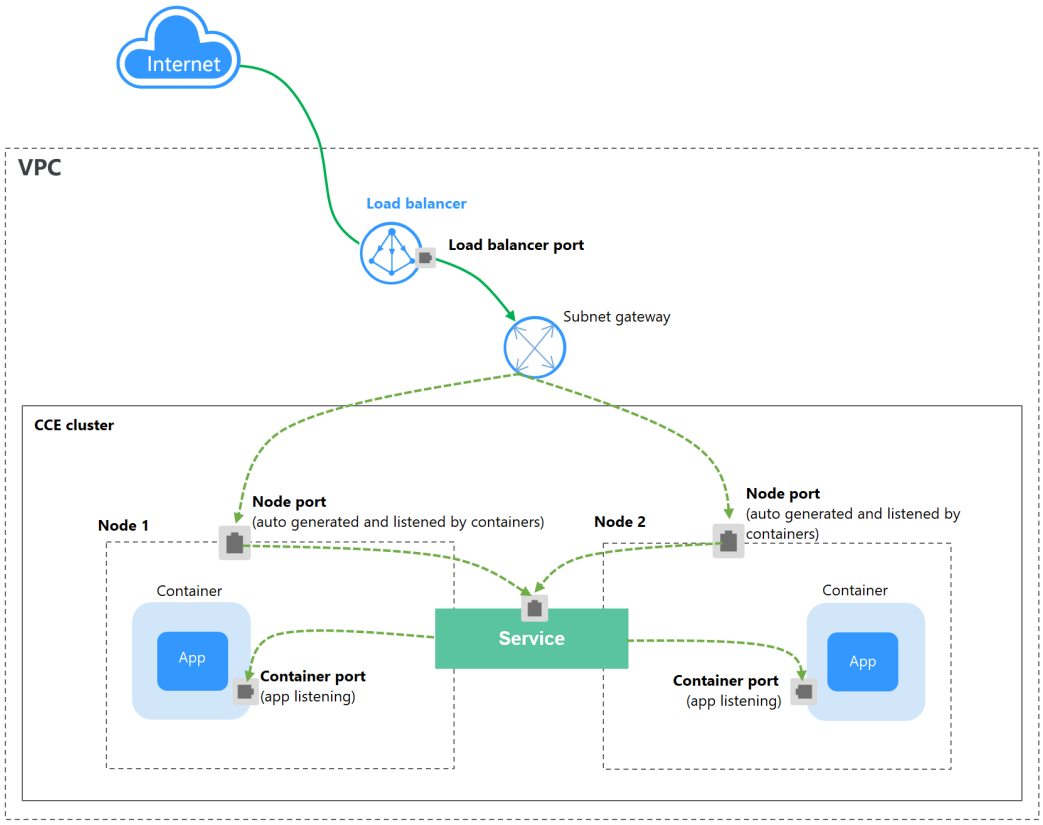
7.3.4.1 Creating a LoadBalancer Service

Scenario

LoadBalancer Services can access workloads from the public network through a load balancer, which is more reliable than EIP-based access. The LoadBalancer access address is in the format of *IP address of public network load balancer:Access port*, for example, **10.117.117.117:80**.

In this access mode, requests are transmitted through an ELB load balancer to a node and then forwarded to the destination pod through the Service.

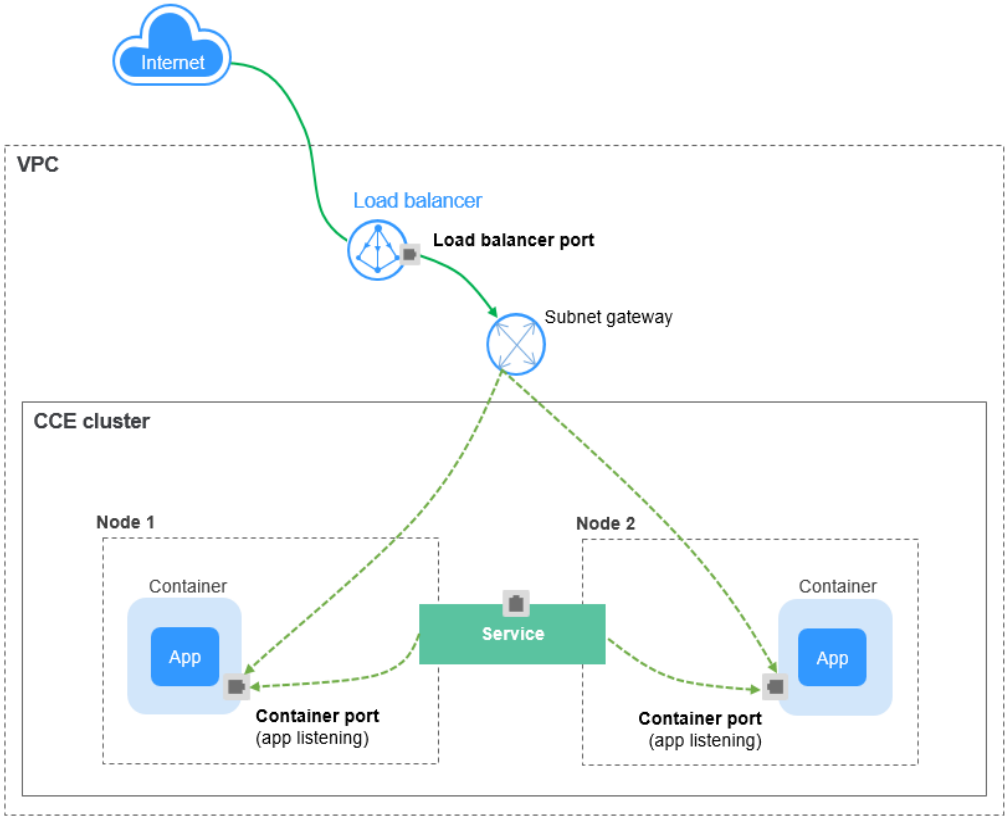
Figure 7-39 LoadBalancer



When **CCE Turbo clusters and dedicated load balancers** are used, passthrough networking is supported to reduce service latency and ensure zero performance loss.

External access requests are directly forwarded from a load balancer to pods. Internal access requests can be forwarded to a pod through a Service.

Figure 7-40 Passthrough networking



Notes and Constraints

- LoadBalancer Services allow workloads to be accessed from public networks through ELB. This access mode has the following restrictions:
  - Automatically created load balancers should not be used by other resources. Otherwise, these load balancers cannot be completely deleted.
  - Do not change the listener name for the load balancer in clusters of v1.15 and earlier. Otherwise, the load balancer cannot be accessed.
- After a Service is created, if the affinity setting is switched from the cluster level to the node level, the connection tracing table will not be cleared. Do not modify the Service affinity setting after the Service is created. To modify it, create a Service again.
- If service affinity is set to the node level (that is, **externalTrafficPolicy** is set to **Local**), the cluster may fail to access the Service by using the ELB address. For details, see **Why a Service Fail to Be Accessed from Within the Cluster**.
- In a CCE Turbo cluster that utilizes Cloud Native Network 2.0, node-level affinity is supported only when the Service backend is connected to a hostNetwork pod.
- Dedicated ELB load balancers can be used only in clusters of v1.17 and later.
- Dedicated load balancers must support private networks (with private IP addresses). If the Service needs to support HTTP, the dedicated load balancers must be of the application (HTTP/HTTPS) type.

- When the cluster service forwarding (proxy) mode is IPVS, the node IP cannot be configured as the external IP of the Service. Otherwise, the node is unavailable.
- If you have an IPVS-backed cluster and use the same ELB load balancer for both the ingress and Service in the same cluster or for the Service ports in different clusters, you will not be able to access the ingress from the nodes or containers in the cluster, or the Service ports in other clusters. This is because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge, which intercepts the load balancer traffic. Use separate load balancers for these ingresses and Services.

Using the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure parameters.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Namespace**: namespace that the workload belongs to.
- **Service Affinity**: For details, see **externalTrafficPolicy (Service Affinity)**.
  - **Cluster level**: The IP addresses and access ports of all nodes in a cluster can access the workload associated with the Service. Accessing the Service will result in a decrease in performance due to route redirection, and the source IP address of the client cannot be obtained.
  - **Node level**: Only the IP address and access port of the node where the workload is located can access the workload associated with the Service. Accessing the Service will not result in a decrease in performance due to route redirection, and the source IP address of the client can be obtained.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Protocol Version**: This function is disabled by default. After this function is enabled, the cluster IP address of the Service can be set to an IPv6 address. For details, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**. **This parameter is available only in clusters of v1.15 or later with IPv6 enabled (set during cluster creation).**
- **Load Balancer**: Select a load balancer type and creation mode.

  A load balancer can be dedicated or shared. A dedicated load balancer supports **Network (TCP/UDP)**, **Application (HTTP/HTTPS)**, or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.

  You can select **Use existing** or **Auto create** to obtain a load balancer. For details about the configuration of different creation modes, see **Table 7-26**.

Table 7-26 Load balancer configurations

How to Create	Configuration
Use existing	Only the load balancers in the same VPC as the cluster can be selected. If no load balancer is available, click **Create Load Balancer** to create one on the ELB console.
Auto create	– **Instance Name**: Enter a load balancer name. – **Enterprise Project**: This parameter is available only for enterprise users who have enabled an enterprise project. Enterprise projects facilitate project-level management and grouping of cloud resources and users. – **AZ**: available only to dedicated load balancers. You can create load balancers in multiple AZs to improve service availability. You can deploy a load balancer in multiple AZs for high availability. – **Frontend Subnet**: available only to dedicated load balancers. It is used to allocate IP addresses for load balancers to provide services externally. – **Backend Subnet**: available only to dedicated load balancers. It is used to allocate IP addresses for load balancers to access the backend service. – **Network Specifications**, **Application-oriented Specifications**, or **Specifications** (available only to dedicated load balancers) ▪ **Fixed**: applies to stable traffic, billed based on specifications. – **EIP**: If you select **Auto create**, you can configure the billing mode and size of the public network bandwidth. – **Resource Tag**: You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.

Set ELB: You can click **Edit** and configure the load balancing algorithm and sticky session.

– **Algorithm**: Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash.

NOTE

- **Weighted round robin**: Requests are forwarded to different servers based on their weights, which indicate server processing performance. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. This algorithm is often used for short connections, such as HTTP services.
- **Weighted least connections**: In addition to the weight assigned to each server, the number of connections processed by each backend server is considered. Requests are forwarded to the server with the lowest connections-to-weight ratio. Building on **least connections**, the **weighted least connections** algorithm assigns a weight to each server based on their processing capability. This algorithm is often used for persistent connections, such as database connections.
- **Source IP hash**: The source IP address of each request is calculated using the hash algorithm to obtain a unique hash key, and all backend servers are numbered. The generated key allocates the client to a particular server. This enables requests from different clients to be distributed in load balancing mode and ensures that requests from the same client are forwarded to the same server. This algorithm applies to TCP connections without cookies.

– **Sticky Session**: This function is disabled by default.

After this function is enabled, the sticky session type will be automatically selected based on the protocol type supported by the listener.

- If the listener's frontend protocol is TCP or UDP, sticky sessions can use source IP addresses. This means that access requests from the same IP address will be directed to the same backend server or pod.
- If the listener's frontend protocol is HTTP or HTTPS, sticky sessions can use load balancer cookies.

NOTE

When the **distribution policy** uses the source IP hash, sticky session cannot be set.

- **Health Check**: Configure health check for the load balancer.
  – **Global health check**: applies only to ports using the same protocol. You are advised to select **Custom health check**.
  – **Custom health check**: applies to **ports** using different protocols. For details about the YAML configuration for custom health check, see **Configuring Health Check on Multiple Ports of a LoadBalancer Service**.

Table 7-27 Health check parameters

Parameter	Description
Protocol	When the protocol of **Port** is set to **TCP**, the TCP and HTTP protocols are supported. When the protocol of **Port** is set to **UDP**, the UDP protocol is supported. – **Check Path** (supported only by HTTP for health check): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.

Parameter	Description
Port	By default, the service port (NodePort or container port of the Service) is used for health check. You can also specify another port for health check. After the port is specified, a service port named **cce-healthz** will be added for the Service. – **Node Port**: If a shared load balancer is used or no ENI instance is associated, the node port is used as the health check port. If this parameter is not specified, a random port is used. The value ranges from 30000 to 32767. – **Container Port**: When a dedicated load balancer is associated with an ENI instance, the container port is used for health check. The value ranges from 1 to 65535.
Check Period (s)	Specifies the maximum interval between health checks. The value ranges from 1 to 50.
Timeout (s)	Specifies the maximum timeout duration for each health check. The value ranges from 1 to 50.
Max. Retries	Specifies the maximum number of health check retries. The value ranges from 1 to 10.

- **Ports**
  - **Protocol**: protocol used by the Service.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.

    NOTE

    ELB allows you to create listeners that listen to ports in a certain range. You can add a maximum of 10 port ranges that do not overlap with each other for each listener.

    To configure port ranges for load balancer listeners, ensure the following conditions are met:

    - The cluster version must be v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, v1.30.1-r0, or later.
    - A dedicated load balancer is used and the TCP, TLS, or UDP protocol is selected.
    - This function relies on ELB capabilities. Before using this function, check whether it is supported in the current region. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.
  - **Frontend Protocol**: the frontend protocol of the load balancer listener for establishing a traffic distribution connection with the client. When a dedicated load balancer is selected, HTTP/HTTPS can be configured only when **Application (HTTP/HTTPS)** is selected.

  - **Health Check**: If **Health Check** is set to **Custom health check**, you can configure health check for ports using different protocols. For details, see **Table 7-27**.

    NOTE

    When a LoadBalancer Service is created, a random node port number (NodePort) is automatically generated.

- **Listener**
  - **SSL Authentication**: Select this option if **HTTPS/TLS** is enabled on the listener port. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication**: Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication**: If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate**: If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If HTTPS mutual authentication is required, HTTPS connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate**: If **HTTPS/TLS** is enabled on the listener port, you must select a server certificate. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **SNI**: If **HTTPS/TLS** is enabled on the listener port, you must determine whether to add an SNI certificate. Before adding an SNI certificate, ensure the certificate contains a domain name. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.

    If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.
  - **Security Policy**: If **HTTPS/TLS** is enabled on the listener port, you can select a security policy. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
  - **Backend Protocol**: If **HTTPS** is enabled on the listener port, HTTP or HTTPS can be used to access the backend server. The default value is **HTTP**. If **TLS** is enabled on the listener port, TCP or TLS can be used to access the backend server. The default value is **TCP**. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
  - **Access Control**
    - **Inherit ELB Configurations**: CCE does not modify the existing access control configurations on the ELB console.
    - **Allow all IP addresses**: No access control is configured.
    - **Trustlist**: Only the selected IP address group can access the load balancer.

- **Blocklist**: The selected IP address group cannot access the load balancer.

  NOTE

  For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can select a maximum of five IP address groups for access control at a time.

- **Advanced Options**

Configuration	Description	Restrictions
Transfer Listener Port Number	If this function is enabled, the listening port on the load balancer can be transferred to backend servers through the HTTP header of the packet.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.
Transfer Port Number in the Request	If this function is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.
Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.
Data Compression	If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. ▪ Brotli can compress all file formats. ▪ GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.

Configuration	Description	Restrictions
Idle Timeout (s)	Timeout for an idle client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.	This configuration is not supported if the port of a shared load balancer uses UDP.
Request Timeout (s)	Timeout for waiting for a request from a client. There are two cases: ■ If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. ■ If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.	This parameter is available only after **HTTP/HTTPS** is enabled on ports.
Response Timeout (s)	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.	This parameter is available only after **HTTP/HTTPS** is enabled on ports.
HTTP2	Whether to use HTTP/2 for a client to communicate with a load balancer. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.	This parameter is available only after **HTTPS** is enabled on ports.

- **Annotation**: The LoadBalancer Service has some advanced CCE functions, which are implemented by annotations. For details, see **Configuring LoadBalancer Services Using Annotations**.

Step 4 Click **OK**.

----End

Using kubectl to Create a Service (Using an Existing Load Balancer)

You can configure Service access using kubectl when creating a workload. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-elb-svc.yaml

NOTE

To enable sticky session, ensure anti-affinity is configured for the workload pods so that the pods are deployed onto different nodes. For details, see **Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance                   # Load balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN            # Load balancer algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP     # The sticky session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'    # Stickiness duration, which is measured in minutes
```



```
    kubernetes.io/elb.health-check-flag: 'on'              # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80     # Port for accessing the Service, which is also the listener port on the load balancer.
    protocol: TCP
    targetPort: 80  # Port used by a Service to access the target container. This port is closely related to the applications running in a container.
    nodePort: 31128  # Port number of the node. If this parameter is not specified, a random port number ranging from 30000 to 32767 is generated.
  type: LoadBalancer
```

The preceding example uses annotations to implement some advanced functions of load balancing, such as sticky session and health check. For details, see **Table 7-28**.

For more annotations and examples related to advanced functions, see **Configuring LoadBalancer Services Using Annotations**.

Table 7-28 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	ID of a load balancer. Mandatory when an existing load balancer is to be associated. **How to obtain**: On the management console, click **Service List**, and choose **Networking** > **Elastic Load Balance**. Click the name of the target load balancer. On the **Summary** tab page, find and copy the ID. **NOTE** The system preferentially connects to the load balancer based on the **kubernetes.io/elb.id** field. If this field is not specified, the **spec.loadBalancerIP** field is used (optional and available only in 1.23 and earlier versions). Do not use the **spec.loadBalancerIP** field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see **Deprecation**.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. Options: • **union**: shared load balancer • **performance**: dedicated load balancer, which can be used only in clusters of v1.17 and later. For details, see **Differences Between Shared and Dedicated Load Balancers**. **NOTE** If a LoadBalancer Service accesses an existing dedicated load balancer, the dedicated load balancer must support TCP/UDP networking.
kubernetes.io/elb.lb-algorithm	No	String	Specifies the load balancing algorithm of the backend server group. The default value is **ROUND_ROBIN**. Options: • **ROUND_ROBIN**: weighted round robin algorithm • **LEAST_CONNECTIONS**: weighted least connections algorithm • **SOURCE_IP**: source IP hash algorithm **NOTE** If this parameter is set to **SOURCE_IP**, the weight setting (**weight** field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-mode	No	String	Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server. • To disable sticky session, do not configure this parameter. • To enable sticky session, add this parameter and set it to **SOURCE_IP**. After this function is enabled, the sticky session type will be automatically selected based on the protocol type supported by the listener. – If the listener's frontend protocol is TCP or UDP, sticky sessions can use source IP addresses. – If the listener's frontend protocol is HTTP or HTTPS, sticky sessions can use load balancer cookies. For details about how to use an HTTP/HTTPS listener, see **Configuring HTTP/HTTPS for a LoadBalancer Service**. **NOTE** When **kubernetes.io/elb.lb-algorithm** is set to **SOURCE_IP** (source IP hash), sticky session cannot be enabled.
kubernetes.io/elb.session-affinity-option	No	**Table 7-29** object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave this parameter blank or set it to **on**. • Disabling health check: Set this parameter to **off**. If this parameter is enabled, the **kubernetes.io/elb.health-check-option** field must also be specified.
kubernetes.io/elb.health-check-option	No	**Table 7-30** object	ELB health check configuration items.

Table 7-29 elb.session-affinity-option data structure

Parameter	Mandatory	Type	Description
persistence_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when **elb.session-affinity-mode** is set to **SOURCE_IP**. Value range: 1 to 60. Default value: **60**

Table 7-30 elb.health-check-option data structure

Parameter	Mandatory	Type	Description
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: **5**
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: **10**
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: **3**
protocol	No	String	Health check protocol. Value options: TCP or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is **HTTP**. Default value: **/** Value range: 1-80 characters

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload has been created.

```
deployment/nginx created
```

kubectl get pod

If information similar to the following is displayed, the workload is running.

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xhw   1/1     Running   0          6s
```

Step 4 Create a Service.

kubectl create -f nginx-elb-svc.yaml

If information similar to the following is displayed, the Service has been created.

```
service/nginx created
```

kubectl get svc

If information similar to the following is displayed, the workload's access mode has been configured.

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP      10.247.0.1      <none>        443/TCP        3d
nginx        LoadBalancer   10.247.130.196  10.78.42.242  80:31540/TCP   51s
```

Step 5 Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

Figure 7-41 Accessing Nginx through the LoadBalancer Service

----End

Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can configure Service access using kubectl when creating a workload. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
```



```
    imagePullSecrets:
    - name: default-secret
```

vi nginx-elb-svc.yaml

NOTE

To enable sticky session, ensure anti-affinity is configured for the workload pods so that the pods are deployed onto different nodes. For details, see **Configuring Workload Affinity or Anti-affinity Scheduling (podAffinity or podAntiAffinity)**.

Example of a Service using a public network shared load balancer:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "vip_subnet_cidr_id": "******",
      "vip_address": "**.**.**.**",
      "eip_type": "5_bgp"
    }'
    kubernetes.io/elb.enterpriseID: '0'      # ID of the enterprise project to which the load balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN                 # Load balancer algorithm
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP          # The sticky session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'     # Stickiness duration, which is
measured in minutes
    kubernetes.io/elb.health-check-flag: 'on'                   # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
    kubernetes.io/elb.tags: key1=value1,key2=value2           # ELB resource tags
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Example Service using a public network dedicated load balancer (only for clusters of v1.17 and later):

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1626694478577",
      "bandwidth_chargemode": "bandwidth",
```



```
    "bandwidth_size": 5,
    "bandwidth_sharetype": "PER",
    "eip_type": "5_bgp",
    "vip_subnet_cidr_id": "******",
    "vip_address": "**.**.**.**",
    "elb_virsubnet_ids": ["******"],
    "ipv6_vip_virsubnet_id": "******",
    "available_zone": [
      ""
    ],
    "l4_flavor_name": "L4_flavor.elb.s1.small"
  }'
  kubernetes.io/elb.enterpriseID: '0'      # ID of the enterprise project to which the load balancer belongs
  kubernetes.io/elb.lb-algorithm: ROUND_ROBIN              # Load balancer algorithm
  kubernetes.io/elb.session-affinity-mode: SOURCE_IP       # The sticky session type is source IP address.
  kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'     # Stickiness duration, which is
measured in minutes
  kubernetes.io/elb.health-check-flag: 'on'              # Enable ELB health check.
  kubernetes.io/elb.health-check-option: '{
    "protocol":"TCP",
    "delay":"5",
    "timeout":"10",
    "max_retries":"3"
  }'
  kubernetes.io/elb.tags: key1=value1,key2=value2          # ELB resource tags
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

The preceding example uses annotations to implement some advanced functions of load balancing, such as sticky session and health check. For details, see **Table 7-31**.

For more annotations and examples related to advanced functions, see **Configuring LoadBalancer Services Using Annotations**.

Table 7-31 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. Options: • **union**: shared load balancer • **performance**: dedicated load balancer, which can be used only in clusters of v1.17 and later. For details, see **Differences Between Shared and Dedicated Load Balancers**.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.autocreate	Yes	**elb.autocreate** object	Whether to automatically create a load balancer associated with the Service. **Example** • Automatically created shared load balancer with an EIP bound: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • Automatically created shared load balancer with no EIP bound: {"type":"inner","name":"A-location-d-test"}
kubernetes.io/elb.subnet-id	None	String	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters of a version later than v1.11.7-r0. For details about how to obtain the value, see **What Is the Difference Between the VPC Subnet API and the OpenStack Neutron Subnet API?**
kubernetes.io/elb.enterpriseID	No	String	**Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.** This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created. If this parameter is not specified or is set to **0**, resources will be bound to the default enterprise project. **How to obtain**: Log in to the management console and choose **Enterprise** > **Project Management** on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.lb-algorithm	No	String	Specifies the load balancing algorithm of the backend server group. The default value is **ROUND_ROBIN**. Options: • **ROUND_ROBIN**: weighted round robin algorithm • **LEAST_CONNECTIONS**: weighted least connections algorithm • **SOURCE_IP**: source IP hash algorithm **NOTE** If this parameter is set to **SOURCE_IP**, the weight setting (**weight** field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.
kubernetes.io/elb.session-affinity-mode	No	String	Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server. • To disable sticky session, do not configure this parameter. • To enable sticky session, add this parameter and set it to **SOURCE_IP**, indicating that the sticky session is based on the source IP address. **NOTE** When **kubernetes.io/elb.lb-algorithm** is set to **SOURCE_IP** (source IP hash), sticky session cannot be enabled.
kubernetes.io/elb.session-affinity-option	No	**Table 7-29** object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave this parameter blank or set it to **on**. • Disabling health check: Set this parameter to **off**. If this parameter is enabled, the **kubernetes.io/elb.health-check-option** field must also be specified.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.health-check-option	No	Table 7-30 object	ELB health check configuration items.
kubernetes.io/elb.tags	No	String	Whether to add resource tags to a load balancer. This function is available only when the load balancer is automatically created, and the cluster is of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions. A tag is in the format of "key=value". Use commas (,) to separate multiple tags.

Table 7-32 elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: **cce-lb+service.UID**
type	No	String	Network type of the load balancer. • **public**: public network load balancer • **inner**: private network load balancer Default: **inner**
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is **cce-bandwidth-** ******. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth billing mode. • **bandwidth**: billed by bandwidth • **traffic**: billed by traffic Default: **bandwidth**

Parameter	Mandatory	Type	Description
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. ● The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. ● The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. ● The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. ● **PER**: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. ● **5_telcom**: China Telecom ● **5_union**: China Unicom ● **5_bgp**: dynamic BGP ● **5_sbgp**: static BGP The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet. This field can be specified only for clusters of v1.21 or later.

Parameter	Mandatory	Type	Description
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block. This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.
available_zone	Yes	Array of strings	AZ where the load balancer is located. You can obtain all supported AZs by **getting the AZ list**. This parameter is available only for dedicated load balancers.
l4_flavor_name	Yes	String	Flavor name of the layer-4 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers.
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of **l4_flavor_name**, that is, both are elastic specifications or fixed specifications.
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Do not use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

Parameter	Mandatory	Type	Description
ipv6_vip_virsubnet_id	No	String	ID of the IPv6 subnet where the load balancer resides. IPv6 must be enabled for the corresponding subnet. This parameter is mandatory only when the dual-stack clusters are used. This parameter is available only for dedicated load balancers.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.

```
deployment/nginx created
```

kubectl get pod

If information similar to the following is displayed, the workload is running.

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-2601814895-c1xhw   1/1     Running   0          6s
```

Step 4 Create a Service.

kubectl create -f nginx-elb-svc.yaml

If information similar to the following is displayed, the Service has been created.

```
service/nginx created
```

kubectl get svc

If information similar to the following is displayed, the workload's access mode has been configured.

```
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.247.0.1       <none>         443/TCP        3d
nginx        LoadBalancer   10.247.130.196   10.78.42.242   80:31540/TCP   51s
```

Step 5 Enter the URL in the address box of the browser, for example, **10.78.42.242:80**. **10.78.42.242** indicates the IP address of the load balancer, and **80** indicates the access port displayed on the CCE console.

The Nginx is accessible.

Figure 7-42 Accessing Nginx through the LoadBalancer Service

----End

7.3.4.2 Configuring LoadBalancer Services Using Annotations

You can add annotations to a YAML file to use some CCE advanced functions. This section describes the available annotations when a LoadBalancer service is created.

- **Interconnection with ELB**
- **Sticky Session**
- **Health Check**
- **HTTP or HTTPS**
- **SNI**
- **Dynamic Adjustment of the Weight of the Backend ECS**
- **Passthrough Capability**
- **Blocklist/Trustlist**
- **Host Network**
- **Timeout**
- **Resource Tags**
- **HTTP/2**
- **Enabling GZIP**
- **Configuring Graceful Exit for the Load Balancer Backend**
- **Obtaining Client IP Addresses**
- **Configuring a Custom EIP**
- **Configuring Port Ranges for a Listener**

Interconnection with ELB

Table 7-33 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. Options: • **union**: shared load balancer • **performance**: dedicated load balancer, which can be used only in clusters of v1.17 and later. For details, see **Differences Between Shared and Dedicated Load Balancers**.	v1.9 or later
kubernetes.io/elb.id	String	Mandatory **when an existing load balancer is to be associated**. ID of a load balancer. **How to obtain**: On the management console, click **Service List**, and choose **Networking** > **Elastic Load Balance**. Click the name of the target load balancer. On the **Summary** tab page, find and copy the ID. **NOTE** The system preferentially connects to the load balancer based on the **kubernetes.io/elb.id** field. If this field is not specified, the **spec.loadBalancerIP** field is used (optional and available only in 1.23 and earlier versions). Do not use the **spec.loadBalancerIP** field to connect to the load balancer. This field will be discarded by Kubernetes. For details, see **Deprecation**.	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.autocreate	Table 7-50	Mandatory **when load balancers are automatically created**. **Example** • Automatically created shared load balancer with an EIP bound: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • Automatically created shared load balancer with no EIP bound: {"type":"inner","name":"A-location-d-test"}	v1.9 or later
kubernetes.io/elb.enterpriseID	String	Optional **when load balancers are automatically created**. **Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.** This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created. If this parameter is not specified or is set to **0**, resources will be bound to the default enterprise project. **How to obtain**: Log in to the management console and choose **Enterprise** > **Project Management** on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.	v1.15 or later
kubernetes.io/elb.subnet-id	String	Optional **when load balancers are automatically created**. ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters of a version later than v1.11.7-r0.	Mandatory for clusters of a version earlier than v1.11.7-r0 Discarded in clusters of a version later than v1.11.7-r0

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.lb-algorithm	String	Specifies the load balancing algorithm of the backend server group. The default value is **ROUND_ROBIN**. Options: • **ROUND_ROBIN**: weighted round robin algorithm • **LEAST_CONNECTIONS**: weighted least connections algorithm • **SOURCE_IP**: source IP hash algorithm **NOTE** If this parameter is set to **SOURCE_IP**, the weight setting (**weight** field) of backend servers bound to the backend server group is invalid, and sticky session cannot be enabled.	v1.9 or later

The following shows how to use the preceding annotations:

- Associate an existing load balancer. For details, see **Using kubectl to Create a Service (Using an Existing Load Balancer)**.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                  # Load balancer ID. Replace it with the
actual value.
    kubernetes.io/elb.class: performance                  # Load balancer type
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN           # Load balancer algorithm
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

- Automatically create a load balancer. For details, see **Using kubectl to Create a Service (Automatically Creating a Load Balancer)**.

  Shared load balancer:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1551163379627",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp"
    }'
    kubernetes.io/elb.enterpriseID: '0'             # ID of the enterprise project to which the load
```



```
balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN     # Load balancer algorithm
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Dedicated load balancer:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.autocreate: '{
      "type": "public",
      "bandwidth_name": "cce-bandwidth-1626694478577",
      "bandwidth_chargemode": "bandwidth",
      "bandwidth_size": 5,
      "bandwidth_sharetype": "PER",
      "eip_type": "5_bgp",
      "available_zone": [
        ""
      ],
      "l4_flavor_name": "L4_flavor.elb.s1.small"
    }'
    kubernetes.io/elb.enterpriseID: '0'             # ID of the enterprise project to which the load
balancer belongs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN     # Load balancer algorithm
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```


Sticky Session

Table 7-34 Annotations for sticky session

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.session-affinity-mode	String	Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server. • To disable sticky session, do not configure this parameter. • To enable sticky session, add this parameter and set it to **SOURCE_IP**, indicating that the sticky session is based on the source IP address. **NOTE** When **kubernetes.io/elb.lb-algorithm** is set to **SOURCE_IP** (source IP hash), sticky session cannot be enabled.	v1.9 or later
kubernetes.io/elb.session-affinity-option	Table 7-53	Sticky session timeout.	v1.9 or later

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance                   # Load balancer type
    kubernetes.io/elb.session-affinity-mode: SOURCE_IP     # The sticky session type is source IP address.
    kubernetes.io/elb.session-affinity-option: '{"persistence_timeout": "30"}'     # Stickiness duration, which is measured in minutes
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```


Health Check

Table 7-35 Annotations for health check

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.health-check-flag	String	Whether to enable the ELB health check. • Enabling health check: Leave this parameter blank or set it to **on**. • Disabling health check: Set this parameter to **off**. If this parameter is enabled, the **kubernetes.io/elb.health-check-option** field must also be specified.	v1.9 or later
kubernetes.io/elb.health-check-option	**Table 7-51**	ELB health check configuration items.	v1.9 or later
kubernetes.io/elb.health-check-options	**Table 7-52**	ELB health check configuration items. Each Service port can be configured separately, and you can configure only some ports. **NOTE** Either **kubernetes.io/elb.health-check-option** or **kubernetes.io/elb.health-check-options** can be configured.	v1.19.16-r5 or later v1.21.8-r0 or later v1.23.6-r0 or later v1.25.2-r0 or later

- The following shows how to use **kubernetes.io/elb.health-check-option**:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance                   # Load balancer type
    kubernetes.io/elb.health-check-flag: 'on'              # Enable ELB health check.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
```



```
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

- For details about how to use **kubernetes.io/elb.health-check-options**, see **Configuring Health Check on Multiple Ports of a LoadBalancer Service**.

HTTP or HTTPS

Table 7-36 Annotations for using HTTP or HTTPS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.protocol-port	String	If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, • **protocol**: specifies the protocol used by the listener port. The value can be **http** or **https**. • **ports**: Service ports specified by **spec.ports[].port**.	v1.19.16 or later
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.	v1.19.16 or later

For details about application scenarios and use cases, see **Configuring HTTP/HTTPS for a LoadBalancer Service**.

SNI

Table 7-37 Annotations for using SNIs

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tls-certificate-ids	String	In ELB, the IDs of SNI certificates that must contain a domain name are separated by commas (,). To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

HTTPS must be enabled. For details, see **Configuring SNI for a LoadBalancer Service**.

Dynamic Adjustment of the Weight of the Backend ECS

Table 7-38 Annotations for dynamically adjusting the weight of the backend ECS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.adaptive-weight	String	Dynamically adjust the weight of the load balancer backend server based on the number pods on the server. In this way, the requests received by each pod are more balanced. • **true**: enabled • **false**: disabled	v1.21 or later

NOTE

This parameter is invalid in passthrough networking, where dedicated load balancers are used in a CCE Turbo cluster.

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                 # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance                # Load balancer type
    kubernetes.io/elb.adaptive-weight: 'true'           # Enable dynamic adjustment of the weight of
```



```
the backend ECS.
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Passthrough Capability

Table 7-39 Annotations for passthrough capability

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.pass-through	String	Whether the access requests from within the cluster to the Service pass through the ELB load balancer.	v1.19 or later

For details about application scenarios and use cases, see **Configuring Passthrough Networking for a LoadBalancer Service**.

Blocklist/Trustlist

Table 7-40 Annotations for ELB access control

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.acl-id	String	• If this parameter is not specified, CCE does not modify access control on the ELB. • If this parameter is left empty, all IP addresses are allowed to access the load balancer. • If this parameter is set to the IP address group ID of the load balancer, access control is enabled and you need to configure an IP address blocklist or trustlist for the load balancer. Additionally, you need to configure both **kubernetes.io/elb.acl-status** and **kubernetes.io/elb.acl-type**. **NOTE** For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can enter up to five IP address groups and separate them with commas (,). **How to obtain**: Log in to the console. In the **Service List**, choose **Networking** > **Elastic Load Balance**. On the Network Console, choose **Elastic Load Balance** > **IP Address Groups** and copy the **ID** of the target IP address group. For details, see **Creating an IP Address Group**.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later
kubernetes.io/elb.acl-status	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **on**: Access control is enabled. • **off**: Access control is disabled.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later
kubernetes.io/elb.acl-type	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **black**: indicates a blocklist. The selected IP address group cannot access the load balancer. • **white**: indicates a trustlist. Only the selected IP address group can access the load balancer.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance               # Load balancer type
    kubernetes.io/elb.acl-id: <your_acl_id>            # ID of an IP address group for accessing a load
balancer
    kubernetes.io/elb.acl-status: 'on'                 # Enable access control.
    kubernetes.io/elb.acl-type: 'white'                # Trustlist for access control
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

For details about application scenarios and use cases, see **Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Service**.

Host Network

Table 7-41 Annotations for host network

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/hws-hostNetwork	String	If the pod uses **hostNetwork**, the ELB forwards the request to the host network after this annotation is used. Options: • **true**: enabled • **false** (default): disabled	v1.9 or later

The following shows how to use the preceding annotations:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance               # Load balancer type
    kubernetes.io/hws-hostNetwork: 'true'              # The load balancer forwards the request to the
host network.
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
```



```
  targetPort: 80
type: LoadBalancer
```

Timeout

Table 7-42 Annotation for configuring timeout

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.keepalive_timeout	String	Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request. Value: • For TCP listeners, the value ranges from **10** to **4000** (in seconds). The default value is **300**. • For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from **0** to **4000** (in seconds). The default value is **60**. • For UDP listeners, this parameter does not take effect.	Dedicated load balancers: v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, or later Shared load balancers: v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

For details about application scenarios and use cases, see **Configuring Timeout for a LoadBalancer Service**.

Resource Tags

Table 7-43 Annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tags	String	Add resource tags to a load balancer. This parameter can be configured only when a load balancer is automatically created. A tag is in the format of "key=value". Use commas (,) to separate multiple tags.	v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later

For details about application scenarios and use cases, see **Using kubectl to Create a Service (Automatically Creating a Load Balancer)**.

HTTP/2

Table 7-44 Annotations for using HTTP/2

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.http2-enable	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Options: • **true**: enabled • **false**: disabled (default value) Note: **HTTP/2 can be enabled or disabled only when the listener uses HTTPS.** This parameter is invalid and defaults to **false** when the listener protocol is HTTP.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

For details about application scenarios and use cases, see **Configuring HTTP/2 for a LoadBalancer Service**.

Enabling GZIP

Table 7-45 Annotations for enabling GZIP

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.gzip-enabled	String	LoadBalancer Services support data compression, which reduces the size of files to be transferred, improves file transfer efficiency, and reduces the bandwidth needed for the transmission. If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. By default, data compression is disabled. The files in the following format can be compressed: • Brotli can compress all file formats. • GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss +xml application, atom+xml application, xml application, or json This function is available only for HTTP/HTTPS listeners of dedicated load balancers. If the advanced configuration for enabling data compression or the target annotation is deleted, the ELB configuration will not be modified.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

For details about application scenarios and use cases, see **Configuring GZIP Data Compression for a LoadBalancer Service**.

Configuring Graceful Exit for the Load Balancer Backend

You are allowed to configure graceful exit time for the backend servers of load balancers associated with Services providing layer-4 load balancing for clusters. The default setting for graceful exit time is enabled and set to 300s.

NOTE

ELB does not offer this feature to everyone. If you want to use it, you need to submit a service ticket to ELB.

Table 7-46 Annotations for the graceful exit of the load balancer backend

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.connection-drain-enable	String	Specifies whether to enable graceful exit for the load balancer backend.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later
kubernetes.io/elb.connection-drain-timeout	String	Specifies the graceful exit time of the load balancer backend. The value ranges from **10** to **4000**, in seconds.	

Obtaining Client IP Addresses

Table 7-47 Annotations for obtaining client IP addresses

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.transparent-client-ip	String	This parameter is available only when a shared load balancer is used to create a LoadBalancer Service that is TCP/UDP-compliant. • **true**: Enabling the function of obtaining the client source IP address. • **false**: Disabling the function of obtaining the client source IP address.	v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later

For details about application scenarios and use cases, see **Enabling a LoadBalancer Service to Obtain the Client IP Address**.

Configuring a Custom EIP

Table 7-48 Annotations of custom EIP configurations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.custom-eip-id	String	ID of the custom EIP, which can be seen on the EIP console The EIP must be bindable.	v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, v1.30.1-r0, or later

For details about application scenarios and use cases, see **Configuring a Custom EIP for a LoadBalancer Service**.

Configuring Port Ranges for a Listener

Table 7-49 Annotations for configuring port ranges for a listener

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.port-ranges	String	If a dedicated load balancer is used and the TCP, UDP, or TLS protocol is selected, you can create a listener that listens to ports within a certain range from 1 to 65535. You can add a maximum of 10 port ranges that do not overlap for each listener. The parameter value is in the following format, where **ports_name** and **port** must be unique: '{"*<ports_name_1>*":["*<port_1>*,*<port_2>*","*<port_3>*,*<port_4>*"], "*<ports_name_2>*":["*<port_5>*,*<port_6>*","*<port_7>*,*<port_8>*"]}' For example, the port names are **cce-service-0** and **cce-service-1**, and the listener listens to ports 100–200 and 300–400, and 500–600 and 700–800, respectively. '{"cce-service-0":["100,200", "300,400"], "cce-service-1":["500,600", "700,800"]}' **NOTE** This function relies on ELB capabilities. Before using this function, check whether it is supported in the current region. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.	v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, v1.30.1-r0, or later

For details about application scenarios and use cases, see **Configuring a Range of Listening Ports for LoadBalancer Services**.

Parameters for Automatically Creating a Load Balancer

Table 7-50 elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: **cce-lb+service.UID**
type	No	String	Network type of the load balancer. ● **public**: public network load balancer ● **inner**: private network load balancer Default: **inner**
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is **cce-bandwidth-** ******. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth billing mode. ● **bandwidth**: billed by bandwidth ● **traffic**: billed by traffic Default: **bandwidth**
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. ● The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. ● The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. ● The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.

Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. • **PER**: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. • **5_telcom**: China Telecom • **5_union**: China Unicom • **5_bgp**: dynamic BGP • **5_sbgp**: static BGP The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet. This field can be specified only for clusters of v1.21 or later.
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block. This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.
available_zone	Yes	Array of strings	AZ where the load balancer is located. You can obtain all supported AZs by **getting the AZ list**. This parameter is available only for dedicated load balancers.
l4_flavor_name	Yes	String	Flavor name of the layer-4 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers.

Parameter	Mandatory	Type	Description
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of **l4_flavor_name**, that is, both are elastic specifications or fixed specifications.
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Do not use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`
ipv6_vip_virsubnet_id	No	String	ID of the IPv6 subnet where the load balancer resides. IPv6 must be enabled for the corresponding subnet. This parameter is mandatory only when the dual-stack clusters are used. This parameter is available only for dedicated load balancers.

Table 7-51 elb.health-check-option data structure

Parameter	Mandatory	Type	Description
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: **5**
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: **10**
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: **3**

Parameter	Mandatory	Type	Description
protocol	No	String	Health check protocol. Value options: TCP or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is **HTTP**. Default value: **/** Value range: 1-80 characters

Table 7-52 elb.health-check-options

Parameter	Mandatory	Type	Description
target_service_port	Yes	String	Port for health check specified by spec.ports. The value consists of the protocol and port number, for example, TCP:80.
monitor_port	No	String	Re-specified port for health check. If this parameter is not specified, the service port is used by default. **NOTE** Ensure that the port is in the listening state on the node where the pod is located. Otherwise, the health check result will be affected.
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: **5**
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: **10**
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: **3**
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP, or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is **HTTP**. Default value: **/** Value range: 1-80 characters

Table 7-53 elb.session-affinity-option data structure

Parameter	Mandatory	Type	Description
persistence_timeout	Yes	String	Sticky session timeout, in minutes. This parameter is valid only when **elb.session-affinity-mode** is set to **SOURCE_IP**. Value range: 1 to 60. Default value: **60**

7.3.4.3 Configuring HTTP/HTTPS for a LoadBalancer Service

Notes and Constraints

- Only clusters of v1.19.16 or later support HTTP or HTTPS.

Table 7-54 Scenarios where a load balancer supports HTTP or HTTPS

ELB Type	Application	Whether to Support HTTP or HTTPS	Description
Shared load balancer	Interconnecting with an existing load balancer	Supported	None
	Automatically creating a load balancer	Supported	None
Dedicated load balancer	Interconnecting with an existing load balancer	Supported	• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, or v1.27.1-r10, the load balancer flavor must **support both the layer-4 and layer-7 routing**. • For v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, and later versions, the load balancer flavor **must support layer-7 routing**.

ELB Type	Application	Whether to Support HTTP or HTTPS	Description
	Automatically creating a load balancer	Supported	• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, or v1.27.1-r10, the load balancer flavor must **support both the layer-4 and layer-7 routing**. • For v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10, v1.27.1-r10, and later versions, the load balancer flavor **must support layer-7 routing**.

- Do not connect an ingress and a Service that uses HTTP or HTTPS to the same listener of the same load balancer. Otherwise, a port conflict occurs.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure Service parameters. In this example, only mandatory parameters required for using HTTP/HTTPS are listed. For details about how to configure other parameters, see **Using the Console**.

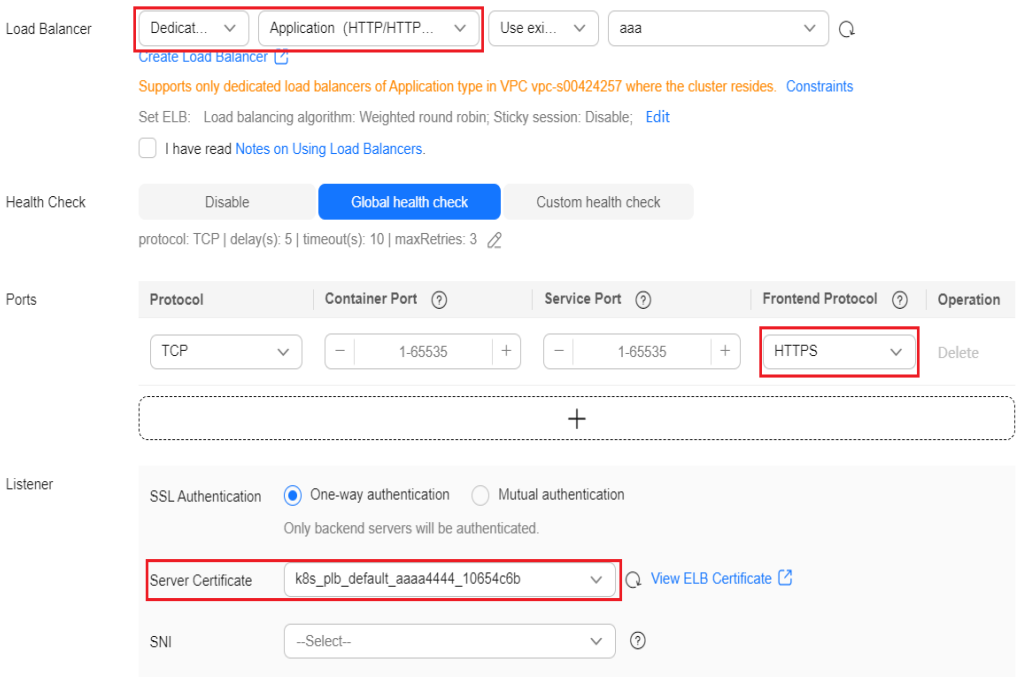
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared. To enable HTTP/HTTPS on the listener port of a dedicated load balancer, the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.

  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: specifies whether to enable HTTP/HTTPS on the listener port. For a **dedicated load balancer**, to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
- **Listener**
  - **SSL Authentication**: Select this option if **HTTPS** is enabled on the listener port. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication**: Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication**: If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate**: If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If HTTPS mutual authentication is required, HTTPS connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate**: If **HTTPS** is enabled on the listener port, you must select a server certificate. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **SNI**: If **HTTPS** is enabled on the listener port, you must determine whether to add an SNI certificate. Before adding an SNI certificate, ensure the certificate contains a domain name. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **Security Policy**: If **HTTPS** is enabled on the listener port, you can select a security policy. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
  - **Backend Protocol**: If **HTTPS** is enabled on the listener port, HTTP or HTTPS can be used to access the backend server. The default value is **HTTP**. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

  NOTE

  If multiple HTTPS Services are released, all listeners will use the same certificate configuration.

Figure 7-43 HTTP or HTTPS



Step 4 Click **OK**.

----End

Using kubectl

If a Service uses the HTTP or HTTPS protocol, it is important to take note of the following configuration requirements:

- Different ELB types and cluster versions have different requirements on flavors. For details, see **Table 7-54**.
- The two ports in **spec.ports** must correspond to those in **kubernetes.io/elb.protocol-port**. In this example, ports 443 and 80 are enabled with HTTPS and HTTP, respectively.

The following is a configuration example for automatically creating a dedicated load balancer, in which key configurations are marked in red:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
# Specify the Layer 4 and Layer 7 flavors in the parameters for automatically creating a load balancer.
    kubernetes.io/elb.autocreate: '
     {
         "type": "public",
         "bandwidth_name": "cce-bandwidth-1634816602057",
         "bandwidth_chargemode": "bandwidth",
         "bandwidth_size": 5,
         "bandwidth_sharetype": "PER",
         "eip_type": "5_bgp",
         "available_zone": [
             ""
         ],
         "l7_flavor_name": "L7_flavor.elb.s2.small",
         "l4_flavor_name": "L4_flavor.elb.s1.medium"
```



```
    }'
    kubernetes.io/elb.class: performance  # Dedicated load balancer
    kubernetes.io/elb.protocol-port: "https:443,http:80"  # HTTP/HTTPS and port number, which must be the same as the port numbers in spec.ports
    kubernetes.io/elb.cert-id: "17e3b4f4bc40471c86741dc3aa211379"  # Certificate ID of the LoadBalancer Service
  labels:
    app: nginx
    name: test
  name: test
  namespace: default
spec:
  ports:
  - name: cce-service-0
    port: 443
    protocol: TCP
    targetPort: 80
  - name: cce-service-1
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    version: v1
  sessionAffinity: None
  type: LoadBalancer
```

Table 7-55 Key parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	If a Service is TLS/HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, • **protocol**: specifies the protocol used by the listener port. The value can be **tls**, **http**, or **https**. • **port**: Service port specified by **spec.ports[].port**. In this example, ports 443 and 80 are enabled with HTTPS and HTTP, respectively. Therefore, the parameter value is *https:443,http:80*.
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the TLS/HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

7.3.4.4 Configuring SNI for a LoadBalancer Service

An SNI certificate is an extended server certificate that allows the same IP address and port number to provide multiple access domain names for external systems.

Different security certificates can be used based on the domain names requested by clients to ensure HTTPS communication security.

When configuring SNI, you need to add a certificate associated with a domain name. The client submits the requested domain name information when initiating an SSL handshake request. After receiving the SSL request, the load balancer searches for the certificate based on the domain name. If the certificate is found, the load balancer will return it to the client. If the certificate is not found, the load balancer will return the default server certificate.

NOTE

After SNI is configured, if you delete the SNI configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- You have created one or more SNI certificates in ELB and specified a domain name in these certificates. For details, see **Adding a Certificate**.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure Service parameters. In this example, only mandatory parameters required for using SNI are listed. For details about how to configure other parameters, see **Using the Console**.

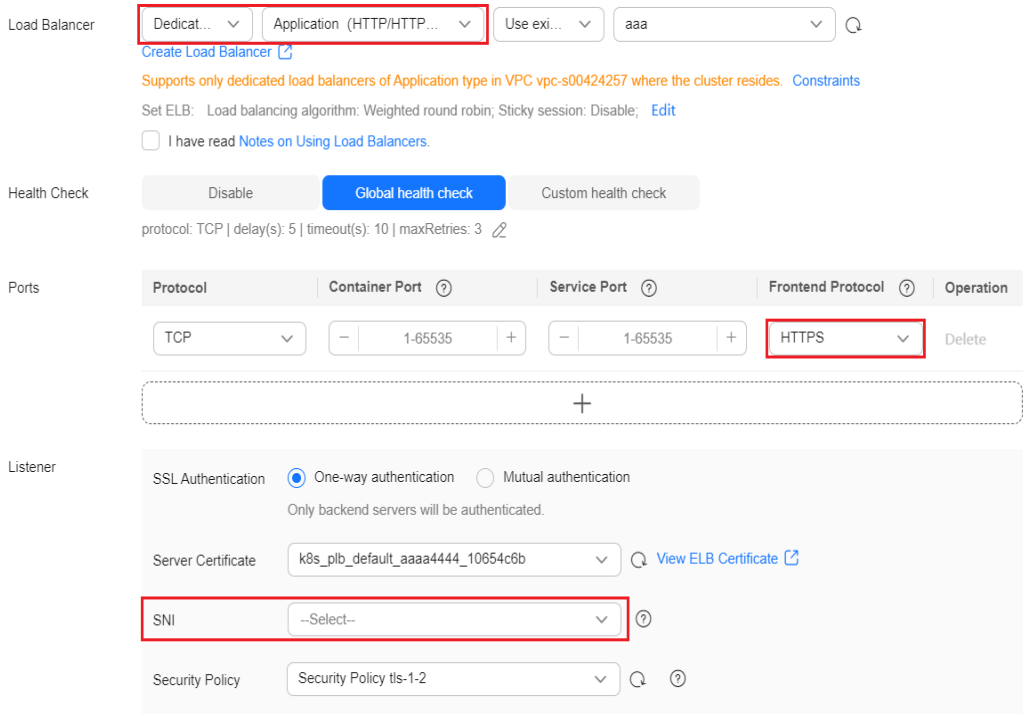
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared. To enable HTTP/HTTPS on the listener port of a dedicated load balancer, the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.

  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.

- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: In this example, HTTPS must be enabled for the Service to use SNI. For a **dedicated load balancer**, to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
- **Listener**
  - **SSL Authentication**: Select this option if **HTTPS** is enabled on the listener port. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication**: Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication**: If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate**: If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If HTTPS mutual authentication is required, HTTPS connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate**: Select a server certificate as the default certificate. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **SNI**: Add an SNI certificate containing a domain name. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.

    If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default.
  - **Security Policy**: If **HTTPS** is enabled on the listener port, you can select a security policy. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
  - **Backend Protocol**: If **HTTPS** is enabled on the listener port, HTTP or HTTPS can be used to access the backend server. The default value is **HTTP**. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Figure 7-44 SNI



Step 4 Click **OK**.

----End

Using kubectl

This section uses an existing load balancer as an example. An example YAML file of a SNI-compliant Service is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance        # Load balancer type
    kubernetes.io/elb.id: 65318265-4f01-4541-a654-fa74e439dfd3  # ID of an existing load balancer
    kubernetes.io/elb.protocol-port: https:80    # Port where SNI is to be enabled
    kubernetes.io/elb.cert-id: b64ab636f1614e1a960b5249c497a880    # HTTPS server certificate
    kubernetes.io/elb.tls-certificate-ids:
5196aa70b0f143189e4cb54991ba2286,8125d71fcc124aabbe007610cba42d60    # SNI certificate IDs
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
spec:
  selector:
    app: test
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.**.**.**       #  Private IP address of the load balancer
```


Table 7-56 Key parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, • **protocol**: specifies the protocol used by the listener port. The value can be **http** or **https**. • **ports**: Service ports specified by **spec.ports[].port**. For example, to use SNI, the Service protocol must be **https** and the Service port must be **80**. Therefore, the parameter value is **https:80**.
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.
kubernetes.io/elb.tls-certificate-ids	String	In ELB, the IDs of SNI certificates that must contain a domain name are separated by commas (,). If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

7.3.4.5 Configuring HTTP/2 for a LoadBalancer Service

Services can be exposed via HTTP/2. By default, HTTP/1.x is used between clients and load balancers. HTTP/2 can improve access performance between clients and load balancers, but HTTP/1.x is still used between load balancers and backend servers.

NOTE

- An HTTPS-compliant load balancer supports HTTP/2.
- After HTTP/2 is configured, if you delete the advanced configuration for enabling HTTP/2 on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

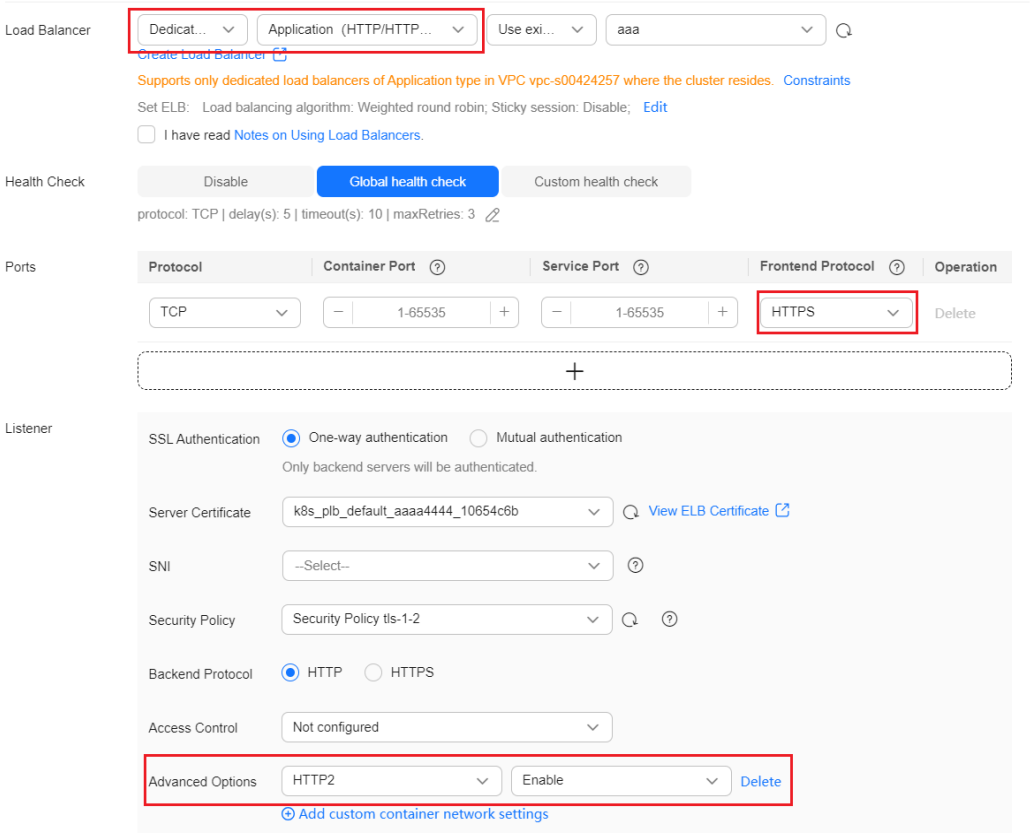
Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Using the Console**.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared. To enable HTTP/HTTPS on the listener port of a dedicated load balancer, the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.

  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: In this example, HTTPS must be enabled for the Service to use HTTP/2. For a **dedicated load balancer**, to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
- **Listener**
  - SSL authentication is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication**: Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication**: If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate**: If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If HTTPS mutual authentication is required, HTTPS connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate**: Select a server certificate when HTTPS is used. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **SNI**: Add an SNI certificate containing a domain name. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **Advanced Options**: Click **Add custom container network settings** and enable HTTP/2.

Figure 7-45 Enabling HTTP/2



Step 4 Click **OK**.

----End

Using kubectl

To enable HTTP/2, add the following annotation:

```
kubernetes.io/elb.http2-enable: 'true'
```

The following shows an example YAML file where an existing load balancer is associated:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: test
    version: v1
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 35cb350b-23e6-4551-ac77-10d5298f5204
    kubernetes.io/elb.protocol-port: https:443
    kubernetes.io/elb.cert-id: b64ab636f1614e1a960b5249c497a880
    kubernetes.io/elb.http2-enable: 'true'
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN
spec:
  selector:
    app: test
    version: v1
  externalTrafficPolicy: Cluster
```



```
ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 443
    protocol: TCP
type: LoadBalancer
loadBalancerIP: **.**.**.**
```

Table 7-57 HTTP/2 parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	If a Service is HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, ● **protocol**: specifies the protocol used by the listener port. The value can be **http** or **https**. ● **ports**: Service ports specified by **spec.ports[].port**. For example, to use HTTPS, the Service port must be **443**. Therefore, the parameter value is **https:443**.
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.
kubernetes.io/elb.http2-enable	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Options: ● **true**: enabled ● **false**: disabled (default value) Note: **HTTP/2 can be enabled or disabled only when the listener uses HTTPS.** This parameter is invalid and defaults to **false** when the listener protocol is HTTP.

7.3.4.6 Configuring an HTTP/HTTPS Header for a LoadBalancer Service

HTTP headers are a list of strings sent and received by both the client and server on every HTTP request and response. This section describes HTTP headers supported by HTTP and HTTP listeners.

NOTE

- HTTP/HTTPS headers rely on ELB. Before using HTTP/HTTPS headers in a Service, check whether HTTP/HTTPS headers are supported in the current region. For details, see **HTTP/HTTPS Headers**.
- After HTTP or HTTPS is configured, if you delete the HTTP or HTTPS configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Table 7-58 Headers

Header	Feature	Description
X-Forwarded-Port	Transfer Listener Port Number	If this option is enabled, the port number used by the listener will be transmitted to backend servers through the **X-Forwarded-Port** header.
X-Forwarded-For-Port	Transfer Port Number in the Request	If this option is enabled, the port number used by the client will be transmitted to backend servers through the **X-Forwarded-For-Port** header.
X-Forwarded-Host	Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

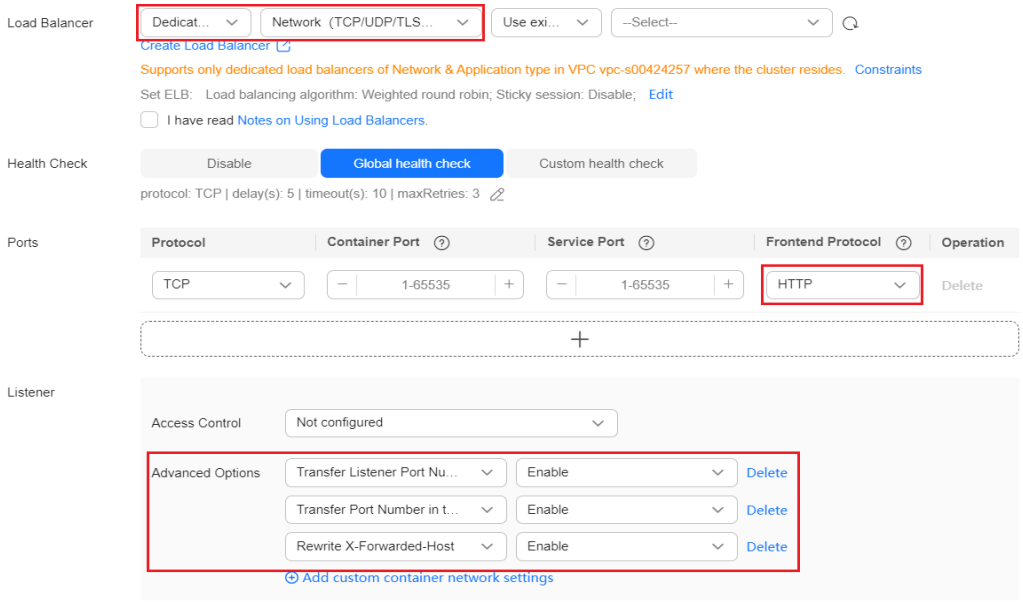
Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Using the Console**.

- **Service Name**: Specify a Service name, which can be the same as the workload name.

- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - In this example, only dedicated load balancers are supported, and the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP/TLS) & Application (HTTP/HTTPS)**. Otherwise, HTTP or HTTPS cannot be enabled on the listener port.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: In this example, select **HTTP** or **HTTPS** for the Service.
- **Listener**
  - **Advanced Options**: Select a proper option.

Configuration	Description	Restrictions
Transfer Listener Port Number	If this function is enabled, the listening port on the load balancer can be transferred to backend servers through the HTTP header of the packet.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.
Transfer Port Number in the Request	If this function is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.
Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.

Figure 7-46 Configuring HTTP/HTTPS headers



Step 4 Click **OK**.

----End

Using kubectl

This section uses an existing load balancer as an example. An example YAML file of a Service with HTTP/HTTPS header enabled is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance                    # Load balancer type, which can only be performance (dedicated load balancer)
    kubernetes.io/elb.id: 35cb350b-23e6-4551-ac77-10d5298f5204       # ID of an existing load balancer
    kubernetes.io/elb.protocol-port: http:80                # The HTTP port 80 is used.
    kubernetes.io/elb.x-forwarded-port: 'true'              # Obtain the listener port number.
    kubernetes.io/elb.x-forwarded-for-port: 'true'          # Obtain the client port number for requests.
    kubernetes.io/elb.x-forwarded-host: 'true'              # Rewrite X-Forwarded-Host.
spec:
  selector:
    app: nginx
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.**.**.**    # IP address of the load balancer
```


Table 7-59 Key parameters

Parameter	Type	Description
kubernetes.io/elb.x-forwarded-port	String	A load balancer can obtain the port number of a listener using **X-Forwarded-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a listener port number. • **false**: Disable the function of obtaining a listener port number.
kubernetes.io/elb.x-forwarded-for-port	String	A load balancer can obtain a client port number for requests using **X-Forwarded-For-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a client port number for requests. • **false**: Disable the function of obtaining a client port number for requests.
kubernetes.io/elb.x-forwarded-host	String	• **true**: Enable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header will be rewritten using the **Host** header of the client request and transmitted to backend servers. • **false**: Disable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header of the client will be transmitted to backend servers.

7.3.4.7 Configuring Timeout for a LoadBalancer Service

LoadBalancer Services allow you to configure timeout, which is the maximum duration for keeping a connection if no request is received from the client. If no request is received during this period, the load balancer closes the connection and establishes a new one with the client when the next request arrives.

NOTE

After timeout is configured, if you delete the timeout configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Notes and Constraints

The following table lists the scenarios where timeout can be configured for a Service.

Timeout Type	Load Balancer Type	Restrictions	Supported Cluster Version
Idle timeout	Dedicated	None	• v1.19: v1.19.16-r30 or later • v1.21: v1.21.10-r10 or later • v1.23: v1.23.8-r10 or later • v1.25: v1.25.3-r10 or later • Other clusters of later versions
Idle timeout	Shared	UDP is not supported.	• v1.23: v1.23.13-r0 or later • v1.25: v1.25.8-r0 or later • v1.27: v1.27.5-r0 or later • v1.28: v1.28.3-r0 or later • Other clusters of later versions
Request timeout	Dedicated and shared	Only HTTP and HTTPS are supported.	
Response timeout	Dedicated and shared	Only HTTP and HTTPS are supported.	

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure Service parameters. In this example, only mandatory parameters required for configuring timeout are listed. For details about how to configure other parameters, see **Using the Console**.

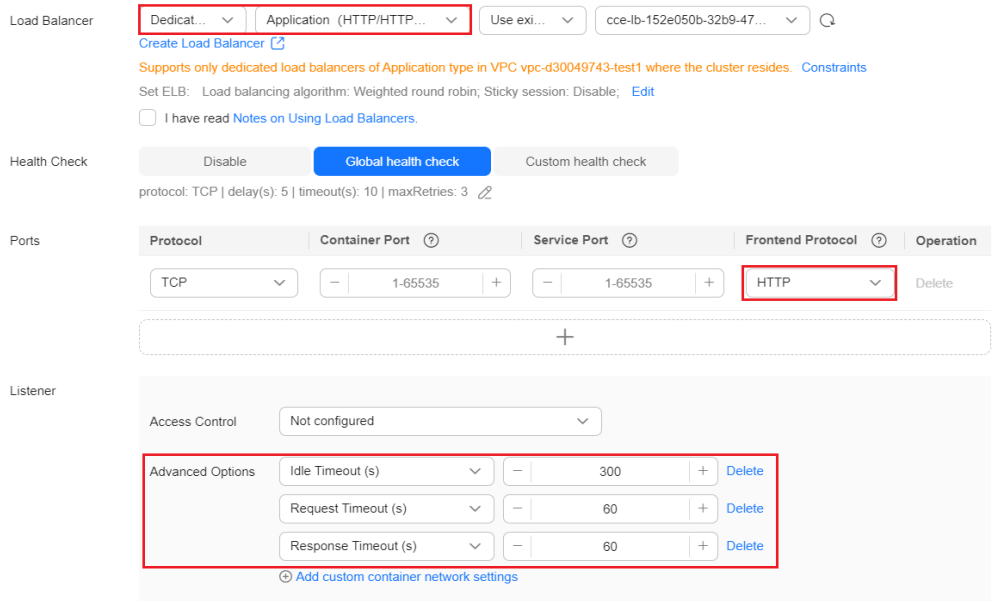
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - A load balancer can be dedicated or shared.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: protocol that the load balancer complies. Timeout cannot be configured for a UDP-compliant shared load balancer.

  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: Select a protocol for the listener. If HTTP/HTTPS is not enabled, only the idle timeout can be configured.

- **Listener**
  - **Advanced Options**: Select a proper option.

Configuration	Description	Restrictions
Idle Timeout	Timeout for an idle client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.	This configuration is not supported if the port of a shared load balancer uses UDP.
Request Timeout	Timeout for waiting for a request from a client. There are two cases: ■ If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. ■ If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.	This parameter is available only after **HTTP/HTTPS** is enabled on ports.
Response Timeout	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.	This parameter is available only after **HTTP/HTTPS** is enabled on ports.

Figure 7-47 Configuring timeout



Step 4 Click **OK**.

----End

Using kubectl

Use annotations to configure timeout. The following shows an example:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing dedicated load balancer is used. Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.protocol-port: http:80 # The HTTP port 80 is used.
    kubernetes.io/elb.keepalive_timeout: '300'  # Timeout setting for client connections
    kubernetes.io/elb.client_timeout: '60'     # Timeout for waiting for a request from a client
    kubernetes.io/elb.member_timeout: '60'     # Timeout for waiting for a response from a backend server
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```


Table 7-60 Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	String	Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request. Value: • For TCP listeners, the value ranges from **10** to **4000** (in seconds). The default value is **300**. • For HTTP, HTTPS, and TERMINATED_HTTPS listeners, the value ranges from **0** to **4000** (in seconds). The default value is **60**. • For UDP listeners, the value ranges from **10** to **4000** (in seconds). The default value is **300**.
kubernetes.io/elb.client_timeout	No	String	Timeout for waiting for a request from a client. There are two cases: • If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. • If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected. The value ranges from **1** to **300** (in seconds). The default value is **60**.
kubernetes.io/elb.member_timeout	No	String	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond within the duration specified by **member_timeout**, the load balancer will stop waiting and return HTTP 504 Gateway Timeout. The value ranges from **1** to **300** (in seconds). The default value is **60**.

7.3.4.8 Configuring TLS for a LoadBalancer Service

TLS can be used if ultra-high performance and large-scale TLS offloading are required. You can use TLS to forward encrypted TCP requests from clients for a Service.

NOTE

Service TLS relies on ELB. Before enabling TLS on a Service, check whether TLS is supported in the current region.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

When TLS is used, sticky session is not allowed.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure Service parameters. In this example, only mandatory parameters required for using TLS are listed. For details about how to configure other parameters, see **Using the Console**.

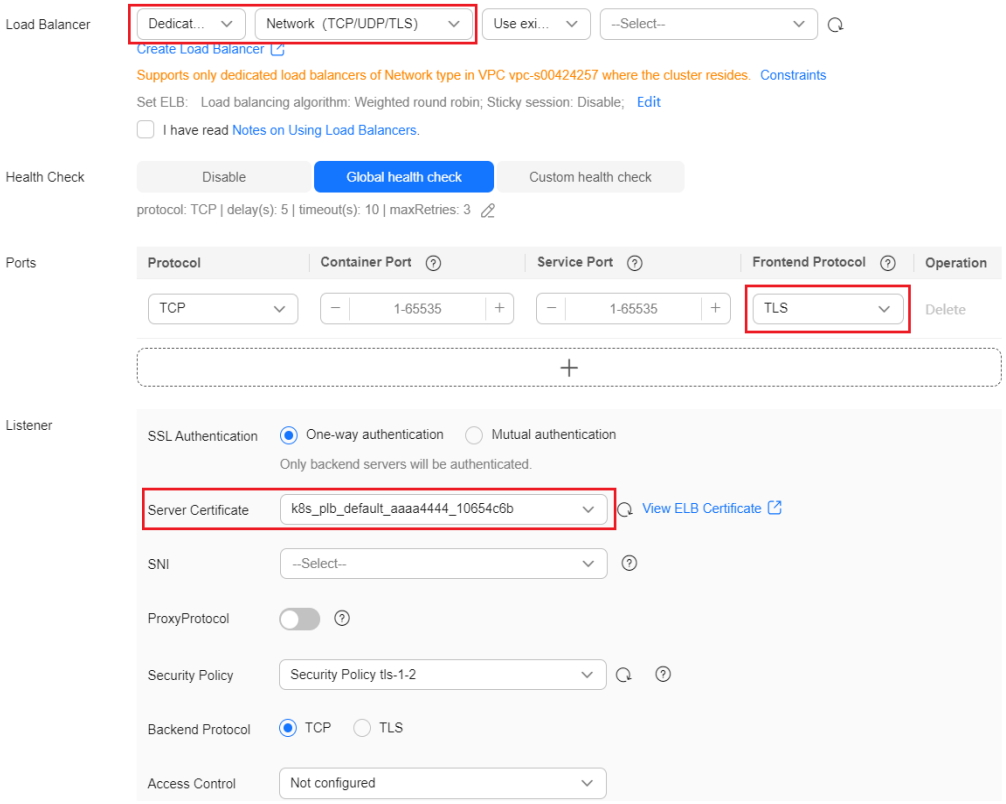
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - In this example, only dedicated load balancers are supported, and the type of the load balancer must be **Network (TCP/UDP/TLS)** or **Network (TCP/UDP/TLS) & Application (HTTP/HTTPS)**. Otherwise, TLS cannot be enabled on the listener port.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, TLS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.

  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: In this example, select **TLS** for the Service. For a **dedicated load balancer**, to use TLS, the type of the load balancer must be **Network (TCP/UDP/TLS)**.
- **Listener**
  - **SSL Authentication**: Select this option if **HTTPS/TLS** is enabled on the listener port. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
    - **One-way authentication**: Only the backend server is authenticated. If you also need to authenticate the identity of the client, select mutual authentication.
    - **Mutual authentication**: If you want the clients and the load balancer to authenticate each other, select this option. Only authenticated clients will be allowed to access the load balancer.
  - **CA Certificate**: If **SSL Authentication** is set to **Mutual authentication**, add a CA certificate to authenticate the client. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If mutual authentication is required, connections can be established only when the client provides a certificate issued by a specific CA.
  - **Server Certificate**: Select a server certificate. If no certificate is available, create one on the ELB console. For details, see **Adding a Certificate**.
  - **ProxyProtocol**: transfers the source IP addresses of clients to backend servers.

    NOTE

    Ensure the backend servers support ProxyProtocol. Otherwise, services may be interrupted.
  - **Security Policy**: If **TLS** is enabled on the listener port, you can select a security policy. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.
  - **Backend Protocol**: If **TLS** is enabled on the listener port, TCP or TLS can be used to access the backend server. The default value is **TCP**. This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Figure 7-48 Configuring TLS



Step 4 Click **OK**.

----End

Using kubectl

This section uses an existing load balancer as an example. An example YAML file of a TLS-compliant Service is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: test-tls
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance                                # Load balancer type, which can only be performance (dedicated load balancer) supported by a TLS listener
    kubernetes.io/elb.id: 35cb350b-23e6-4551-ac77-10d5298f5204          # ID of an existing load balancer
    kubernetes.io/elb.protocol-port: tls:443                            # Port on which TLS is to be enabled
    kubernetes.io/elb.cert-id: 98e91cb03dea418582a438a212b461d5         # TLS server certificate
    kubernetes.io/elb.tls-certificate-ids: e59934f5bc7044f58693de79f1cb4b6d  # TLS SNI certificate
    kubernetes.io/elb.client-ca-cert-id: 5b5178323a2f4eddbafed065945d9069    # Client CA certificate in TLS mutual authentication
    kubernetes.io/elb.proxy-protocol-enable: 'true'                     # ProxyProtocol for transferring the IP addresses of clients to backend servers
    kubernetes.io/elb.security-pool-protocol: 'on'                      # Backend security protocol. If this function is enabled, the backend protocol is TLS. Otherwise, the backend protocol is TCP.
    kubernetes.io/elb.security-policy-id: 175318e0-b6cb-44c5-80a2-0dc372f20df5  # ID of the custom security policy, whose priority is higher than that of the preset security policy
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2-fs                    # Preset security policy
spec:
```



```
selector:
  app: nginx
externalTrafficPolicy: Cluster
ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 443
    protocol: TCP
type: LoadBalancer
loadBalancerIP: **.**.**.**
```

Table 7-61 Key parameters

Parameter	Type	Description
kubernetes.io/elb.protocol-port	String	If a Service is TLS/HTTP/HTTPS-compliant, configure the protocol and port number in the format of "protocol:port". where, • **protocol**: specifies the protocol used by the listener port. The value can be **tls**, **http**, or **https**. • **port**: Service port specified by **spec.ports[].port**. For example, to create a TLS listener, the Service protocol must be **tls** and the Service port must be **443**. Therefore, the parameter value is **tls:443**.
kubernetes.io/elb.cert-id	String	ID of an ELB certificate, which is used as the TLS/HTTPS server certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.
kubernetes.io/elb.tls-certificate-ids	String	In ELB, the IDs of SNI certificates that must contain a domain name are separated by commas (,). To change an ID, remove the SNI certificate by specifying an empty string "". To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

Parameter	Type	Description
kubernetes.io/elb.client-ca-cert-id	String	ID of an ELB CA certificate. A CA certificate is issued by a certificate authority (CA) and used to verify the certificate issuer. If TLS/HTTPS mutual authentication is required, TLS/HTTPS connections can be established only when the client provides a certificate issued by a specific CA. To change an ID, remove the CA certificate by specifying an empty string "". To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, click **Certificates** in the navigation pane, and filter CA certificates. In the load balancer list, copy the ID under the target certificate name.
kubernetes.io/elb.security-pool-protocol	String	If the frontend protocol of a listener is TLS or HTTPS, you can enable the backend security protocol TLS or HTTPS. The backend security protocol of an existing listener cannot be changed. The modification takes effect only on new listeners that are created by changing the protocol or port. • **on/true**: enabled • **off/false**: disabled
kubernetes.io/elb.security-policy-id	String	ID of a custom security policy, whose priority is higher than that of a preset security policy. To change an ID, remove the policy by specifying an empty string "". To obtain a security policy, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance** > **TLS Security Policies**, and click the **Custom Security Policies** tab. In the security policy list, copy the ID under the target policy.
kubernetes.io/elb.tls-ciphers-policy	String	Preset security policy, which is **tls-1-0** by default. To obtain a security policy, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance** > **TLS Security Policies**, and click the **Default Security Policies** tab. In the security policy list, copy the name of the target policy.

Parameter	Type	Description
kubernetes.io/elb.proxy-protocol-enable	String	ProxyProtocol is enabled, which is available only when the frontend protocol is TLS. ProxyProtocol can be used to transfer the IP addresses of clients to backend servers. • **on/true**: enabled • **off/false**: disabled **NOTE** Ensure the backend servers support ProxyProtocol. Otherwise, services may be interrupted.

7.3.4.9 Configuring GZIP Data Compression for a LoadBalancer Service

LoadBalancer Services support data compression, which reduces the size of files to be transferred, improves file transfer efficiency, and reduces the bandwidth needed for the transmission.

NOTE

- This function depends on the ELB capability. Before using this function, check whether it is supported in the current region. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.
- After data compression is configured, if you delete the data compression configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

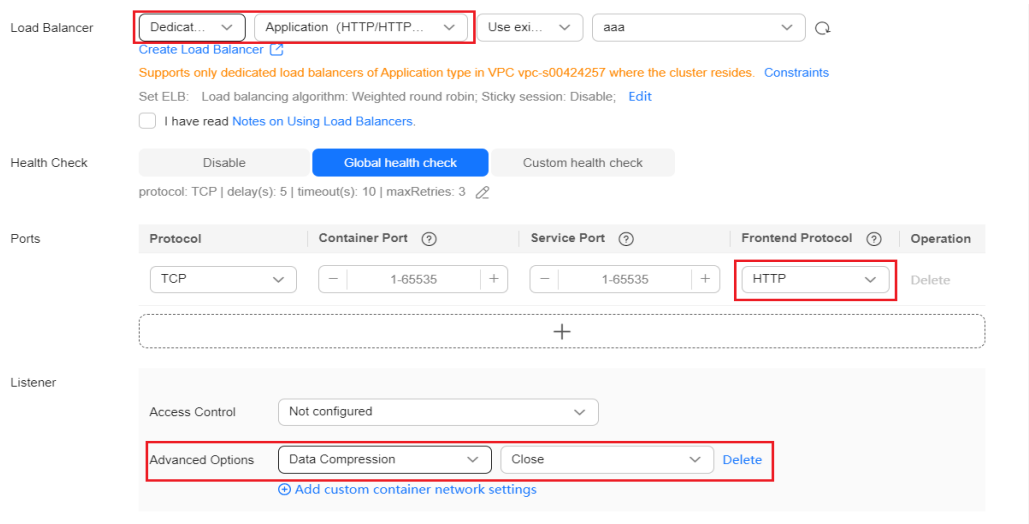
Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Using the Console**.

- **Service Name**: Specify a Service name, which can be the same as the workload name.

- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - In this example, only dedicated load balancers are supported, and the type of the load balancer must be **Application (HTTP/HTTPS)** or **Network (TCP/UDP/TLS) & Application (HTTP/HTTPS)**. Otherwise, HTTP or HTTPS cannot be enabled on the listener port.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP**. If you select **UDP**, HTTP and HTTPS will be unavailable.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: In this example, select **HTTP** or **HTTPS** for the Service.
- **Listener**
  - **Advanced Options**: Select a proper option.

Configuration	Description	Restrictions
Data Compression	If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. ▪ Brotli can compress all file formats. ▪ GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json	This parameter is available only after **HTTP/HTTPS** is enabled on the listener port of a dedicated load balancer.

Figure 7-49 Configuring data compression



Step 4 Click **OK**.

----End

Using kubectl

This section uses an existing load balancer as an example. An example YAML file of a Service with data compression enabled is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: nginx
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance                    # Load balancer type, which can only be performance (dedicated load balancer)
    kubernetes.io/elb.id: 35cb350b-23e6-4551-ac77-10d5298f5204       # ID of an existing load balancer
    kubernetes.io/elb.protocol-port: http:80                # The HTTP port 80 is used.
    kubernetes.io/elb.gzip-enabled: 'true'                  # Enable data compression.
spec:
  selector:
    app: nginx
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.**.**.**    # IP address of the load balancer
```


Table 7-62 Key parameters

Parameter	Type	Description
kubernetes.io/elb.gzip-enabled	String	• **true**: Data compression is enabled, and specific file types will be compressed. • **false**: Data compression is disabled, and no files will be compressed. By default, data compression is disabled. The files in the following format can be compressed: • Brotli can compress all file formats. • GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json This function is available only for HTTP/HTTPS listeners of dedicated load balancers.

7.3.4.10 Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Service

When using a LoadBalancer Service, you can configure a trustlist or blocklist to specify the IP addresses that are allowed or denied accessing a load balancer listener.

- Trustlist: Only the IP addresses in the list can access the listener.
- Blocklist: The IP addresses in the list are not allowed to access the listener.

NOTE

After the blocklist or trustlist access policy is configured, if you delete the blocklist or trustlist access policy configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.12-r0 or later
  - v1.25: v1.25.7-r0 or later
  - v1.27: v1.27.4-r0 or later
  - v1.28: v1.28.2-r0 or later
  - Other clusters of later versions
- An IP address group has been created on the ELB console. For details, see **Creating an IP Address Group**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Services** tab, and click **Create Service** in the upper right corner.

Step 3 Configure Service parameters.

- **Service Name**: Enter a service name, for example, **service-acl**.
- **Service Type**: Select **LoadBalancer**.
- **Service Affinity**: Select cluster level or node level as needed. For details about the differences, see **externalTrafficPolicy (Service Affinity)**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**

  Select a load balancer to be accessed. Only load balancers in the same VPC as the cluster are supported. If no load balancer is available, click **Create Load Balancer** to create one on the ELB console. Alternatively, select **Auto create** to create a load balancer. For details about parameter settings, see **Table 7-26**.
- **Health Check**: defaults to **Global health check**. You can configure this parameter as needed.
- **Ports**
  - **Protocol**: protocol used by the Service.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
- **Access Control**
  - **Inherit ELB Configurations**: CCE does not modify the existing access control configurations on the ELB console.
  - **Allow all IP addresses**: No access control is configured.
  - **Trustlist**: Only the selected IP address group can access the load balancer.
  - **Blocklist**: The selected IP address group cannot access the load balancer.

  NOTE

  For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can select a maximum of five IP address groups for access control at a time.

Step 4 Click **OK**.

----End

Using kubectl

An example YAML file of a Service created using an existing load balancer is as follows:


```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                 # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance                # Load balancer type
    kubernetes.io/elb.acl-id: <your_acl_id>             # ID of an IP address group for accessing a load
balancer
    kubernetes.io/elb.acl-status: 'on'                  # Enable access control.
    kubernetes.io/elb.acl-type: 'white'                 # Trustlist for access control
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Table 7-63 Annotations for ELB access control

Parameter	Type	Description
kubernetes.io/elb.acl-id	String	• If this parameter is not specified, CCE does not modify access control on the ELB. • If this parameter is left empty, all IP addresses are allowed to access the load balancer. • If this parameter is set to the IP address group ID of the load balancer, access control is enabled and you need to configure an IP address blocklist or trustlist for the load balancer. Additionally, you need to configure both **kubernetes.io/elb.acl-status** and **kubernetes.io/elb.acl-type**. **NOTE** For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can enter up to five IP address groups and separate them with commas (,). **How to obtain**: Log in to the console. In the **Service List**, choose **Networking** > **Elastic Load Balance**. On the Network Console, choose **Elastic Load Balance** > **IP Address Groups** and copy the **ID** of the target IP address group. For details, see **Creating an IP Address Group**.

Parameter	Type	Description
kubernetes.io/elb.acl-status	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **on**: Access control is enabled. • **off**: Access control is disabled.
kubernetes.io/elb.acl-type	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **black**: indicates a blocklist. The selected IP address group cannot access the load balancer. • **white**: indicates a trustlist. Only the selected IP address group can access the load balancer.

7.3.4.11 Configuring Health Check on Multiple Ports of a LoadBalancer Service

The annotation field related to the health check of the LoadBalancer Service is upgraded from **Kubernetes.io/elb.health-check-option** to **Kubernetes.io/elb.health-check-options**. Each Service port can be configured separately, and you can configure only some ports. If the port protocol does not need to be configured separately, the original annotation field is still available and does not need to be modified.

Notes and Constraints

- This feature is available in the following versions:
  - v1.19: v1.19.16-r5 or later
  - v1.21: v1.21.8-r0 or later
  - v1.23: v1.23.6-r0 or later
  - v1.25: v1.25.2-r0 or later
  - Versions later than v1.25
- Either **kubernetes.io/elb.health-check-option** or **kubernetes.io/elb.health-check-options** can be configured.
- The **target_service_port** field is mandatory and must be unique.
- For a TCP port, the health check protocol can only be TCP or HTTP. For a UDP port, the health check protocol must be UDP.

Procedure

The following is an example of using the **kubernetes.io/elb.health-check-options** annotation:

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
    version: v1
  annotations:
    kubernetes.io/elb.class: union          # Load balancer type
    kubernetes.io/elb.id: <your_elb_id>    # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.lb-algorithm: ROUND_ROBIN  # Load balancer algorithm
    kubernetes.io/elb.health-check-flag: 'on'    # Enable ELB health check.
    kubernetes.io/elb.health-check-options: '[
    {
        "protocol": "TCP",
        "delay": "5",
        "timeout": "10",
        "max_retries": "3",
        "target_service_port": "TCP:1",
        "monitor_port": "22"
    },
    {
        "protocol": "HTTP",
        "delay": "5",
        "timeout": "10",
        "max_retries": "3",
        "path": "/",
        "target_service_port": "TCP:2",
        "monitor_port": "22",
              "expected_codes": "200-399,401,404"
    }
    ]'
spec:
  selector:
    app: nginx
    version: v1
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 1
      nodePort: 0
      port: 1
      protocol: TCP
    - name: cce-service-1
      targetPort: 2
      nodePort: 0
      port: 2
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: **.**.**.**
```

Table 7-64 elb.health-check-options

Parameter	Mandatory	Type	Description
target_service_port	Yes	String	Port for health check specified by spec.ports. The value consists of the protocol and port number, for example, TCP:80.

Parameter	Mandatory	Type	Description
monitor_port	No	String	Re-specified port for health check. If this parameter is not specified, the service port is used by default. **NOTE** Ensure that the port is in the listening state on the node where the pod is located. Otherwise, the health check result will be affected.
delay	No	String	Health check interval (s) Value range: 1 to 50. Default value: **5**
timeout	No	String	Health check timeout, in seconds. Value range: 1 to 50. Default value: **10**
max_retries	No	String	Maximum number of health check retries. Value range: 1 to 10. Default value: **3**
protocol	No	String	Health check protocol. Default value: protocol of the associated Service Value options: TCP, UDP, or HTTP
path	No	String	Health check URL. This parameter needs to be configured when the protocol is **HTTP**. Default value: **/** Value range: 1-80 characters

7.3.4.12 Configuring Passthrough Networking for a LoadBalancer Service

Application Scenarios

kube-proxy, which is responsible for forwarding intra-cluster traffic, adds the IP addresses of load balancers associated with the LoadBalancer Services to nodes' local forwarding rules by default. When a client from within a cluster accesses the IP address of a load balancer, the traffic is directly forwarded to the destination instead of being forwarded by the load balancer.

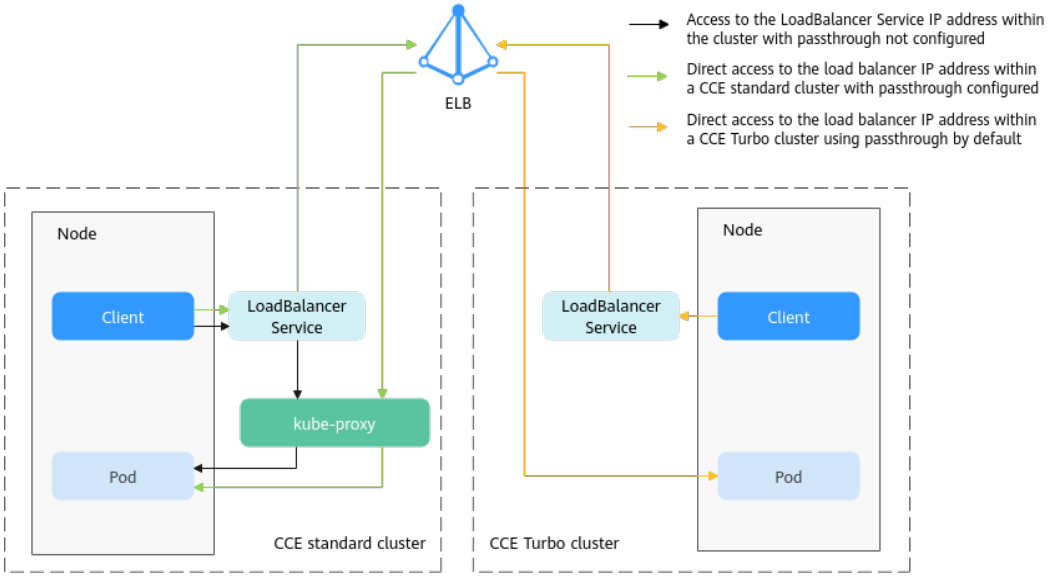
If node-level affinity is configured for a Service (with **externalTrafficPolicy** set to **Local**), the Service will forward traffic only to pods on the node that run these pods. When a node or pod accesses another pod in the same cluster, if the node where the client runs does not have the corresponding backend pod, the access may fail.

Solution

CCE supports passthrough networking. You can configure the **kubernetes.io/elb.pass-through** annotation for the LoadBalancer Service so that the load

balancer forwards the intra-cluster access to the IP address of the load balancer associated with the Service to backend pods.

Figure 7-50 Passthrough networking illustration



- CCE clusters

  When a LoadBalancer Service is accessed within the cluster, the access is forwarded to the backend pods using iptables/IPVS by default.

  When a LoadBalancer Service (configured with elb.pass-through) is accessed within the cluster, the access is first forwarded to the load balancer, then the nodes, and finally to the backend pods using iptables/IPVS.

- CCE Turbo clusters

  When a client accesses a LoadBalancer Service from within the cluster, passthrough is used by default. In this case, the client directly accesses the load balancer private network IP address and then access a container through the load balancer.

Notes and Constraints

- In a CCE standard cluster, after passthrough networking is configured for a dedicated load balancer, the private IP address of the load balancer cannot be accessed from the node where the workload pod resides or other containers on the same node as the workload.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the passthrough settings of Services connected to the same load balancer must be the same.
- If node-level (local) service affinity is used, **kubernetes.io/elb.pass-through** is automatically set to **onlyLocal** to enable pass-through.

Procedure

This section describes how to create a Deployment using an Nginx image and create a Service with passthrough networking enabled.

Step 1 Use the kubectl command line tool to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Use the Nginx image to create a Deployment.

Create an **nginx-deployment.yaml** file. The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```

Run the following command to deploy the workload:

```
kubectl create -f nginx-deployment.yaml
```

Step 3 Create a LoadBalancer Service and set **kubernetes.io/elb.pass-through** to **true**. For details about how to create LoadBalancer Service, see **LoadBalancer**.

The content of the **nginx-elb-svc.yaml** file is as follows. (In this example, a shared load balancer named **james** is automatically created.)

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.pass-through: "true"
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_ty
pe":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Step 4 Run the following command to create the Service:


```
kubectl create -f nginx-elb-svc.yaml
```

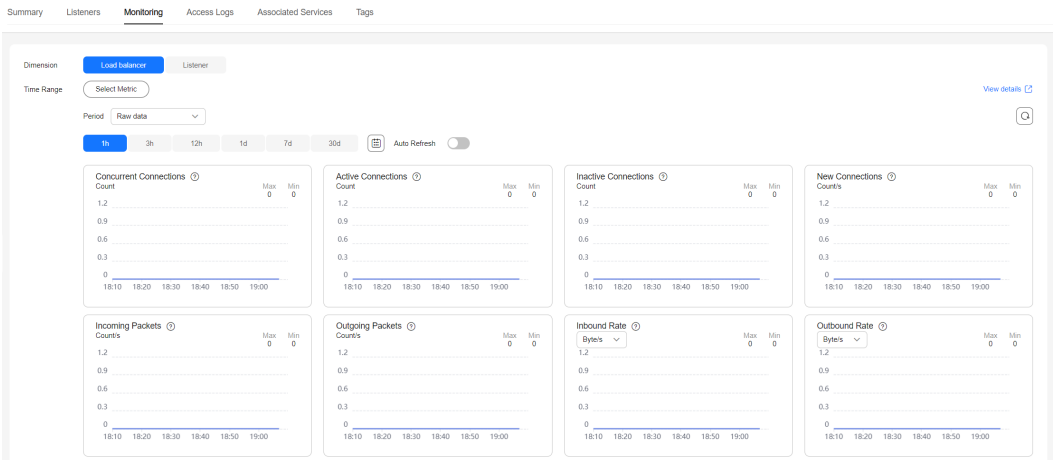
----End

Verification

Step 1 Log in to the ELB console and check the load balancer (named **james** in this example) associated with the Service.

Step 2 Click the load balancer name and click the **Monitoring** tab.

There is 0 connections to the load balancer.



Step 3 Log in to an Nginx container in the cluster using kubectl and access the IP address of the load balancer.

1. Obtain the Nginx containers in the cluster.
   ```
   kubectl get pod
   ```
   Information similar to the following is displayed:
   ```
   NAME                     READY   STATUS    RESTARTS   AGE
   nginx-7c4c5cc6b5-vpncx   1/1     Running   0          9m47s
   nginx-7c4c5cc6b5-xj5wl   1/1     Running   0          9m47s
   ```
2. Log in to an Nginx container container.
   ```
   kubectl exec -it nginx-7c4c5cc6b5-vpncx -- /bin/sh
   ```
3. Access the load balancer IP address.
   ```
   curl **.**.**.**
   ```

Step 4 Wait for a while and check the monitoring data on the ELB console.

If a new access connection is displayed, the access is forwarded by the load balancer as expected.

----End

7.3.4.13 Enabling a LoadBalancer Service to Obtain the Client IP Address

When creating a LoadBalancer Service using a shared load balancer, you can configure annotations for the load balancer listeners to obtain the client IP address.

NOTE

- If a dedicated load balancer is used, the function of obtaining the client IP address is enabled by default.
- After the function of obtaining the client IP address is enabled, if you delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.17-r0 or later
  - v1.25: v1.25.12-r0 or later
  - v1.27: v1.27.9-r0 or later
  - v1.28: v1.28.7-r0 or later
  - v1.29: v1.29.3-r0 or later
  - Other clusters of later versions
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

In CCE standard clusters, this setting takes effect only when Service affinity is set to the node level affinity (with **externalTrafficPolicy** set to **Local**).

In CCE Turbo clusters, this setting cannot take effect, because Service affinity cannot be set to the node level affinity and **externalTrafficPolicy** cannot be set to **Local**.

Using kubectl

An example YAML file of a Service created using an existing load balancer is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>            # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: union                 # Load balancer type
    kubernetes.io/elb.transparent-client-ip: 'true'  # Enable the function of obtaining the client source IP address.
spec:
  selector:
    app: nginx
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```


Table 7-65 Key parameters

Parameter	Type	Description
kubernetes.io/elb.transparent-client-ip	String	This parameter can be configured only for TCP/UDP Services. • **true**: Enabling the function of obtaining the client source IP address. • **false**: Disabling the function of obtaining the client source IP address.

7.3.4.14 Configuring a Custom EIP for a LoadBalancer Service

You can customize the EIP bound to a load balancer that is automatically created by CCE by adding the **kubernetes.io/elb.custom-eip-id** annotation to a Service.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.18-r0 or later
  - v1.25: v1.25.13-r0 or later
  - v1.27: v1.27.10-r0 or later
  - v1.28: v1.28.8-r0 or later
  - v1.29: v1.29.4-r0 or later
  - v1.30: v1.30.1-r0 or later
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- A custom EIP for a Service can be configured only when the Service is being updated, and the Service's annotation contains **kubernetes.io/elb.eip-id**.
- A custom EIP must be unbound to any resources.
- After you configure a custom EIP for a load balancer, if the existing EIP bound to the load balancer was automatically created by CCE during load balancer creation and is not being used by any other resources, the existing EIP will be deleted automatically when the associated Service is deleted. However, if the existing EIP was manually created, it will be unbound from the load balancer when you delete the Service, and you will need to manually delete the EIP.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Automatically create a load balancer with an EIP bound when creating a Service. For details, see **Using kubectl to Create a Service (Automatically Creating a Load Balancer)**.

An example YAML file of a Service created using a dedicated load balancer is as follows:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.autocreate:
'{"type":"public","bandwidth_name":"aaaaa","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"ban
dwidth_sharetype":"PER","eip_type":"5_g-vm","name":"xxx","available_zone":["xxx"],"elb_virsubnet_ids":
["fc0c61cd-c987-49c4-99a4-
b7d816b57581"],"l7_flavor_name":"","l4_flavor_name":"L4_flavor.elb.pro.max","vip_subnet_cidr_id":"cf35b03f
-c6ca-4f75-aa70-e2166cb1f800"}'
    kubernetes.io/elb.eip-id: 8560972c-2cc5-4699-94d6-e46f146eb73d     # ID of the EIP automatically
assigned during load balancer creation
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 0e78a84a-7deb-4747-aeb6-09b6a820b001
  labels:
    app: test-svc
    version: v1
  name: test-eip
  namespace: default
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 10.247.93.235
  clusterIPs:
  - 10.247.93.235
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  loadBalancerIP: *.*.*.*
  ports:
  - name: cce-service-0
    nodePort: 31354
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: test-svc
    version: v1
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: *.*.*.*
    - ip: 192.168.0.15
```

Step 3 Modify the Service configurations and add the **kubernetes.io/elb.custom-eip-id** annotation.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.autocreate:
'{"type":"public","bandwidth_name":"aaaaa","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"ban
dwidth_sharetype":"PER","eip_type":"5_g-vm","name":"xxx","available_zone":["xxx"],"elb_virsubnet_ids":
["fc0c61cd-c987-49c4-99a4-
b7d816b57581"],"l7_flavor_name":"","l4_flavor_name":"L4_flavor.elb.pro.max","vip_subnet_cidr_id":"cf35b03f
-c6ca-4f75-aa70-e2166cb1f800"}'
    kubernetes.io/elb.eip-id: 8560972c-2cc5-4699-94d6-e46f146eb73d     # ID of the EIP automatically
assigned during load balancer creation
    kubernetes.io/elb.custom-eip-id: 88c197a1-cb85-4b38-b672-1d60dc5d00db  # ID of the custom EIP
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 0e78a84a-7deb-4747-aeb6-09b6a820b001
  labels:
    app: test-svc
```



```
    version: v1
  name: test-eip
  namespace: default
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 10.247.93.235
  clusterIPs:
  - 10.247.93.235
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  loadBalancerIP: *.*.*.*
  ports:
  - name: cce-service-0
    nodePort: 31354
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: test-svc
    version: v1
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: *.*.*.*
    - ip: 192.168.0.15
```

Table 7-66 Key parameters

Parameter	Type	Description
kubernetes.io/elb.custom-eip-id	String	ID of the custom EIP, which can be seen on the EIP console The EIP must be bindable.

Step 4 After the Service is updated, check the Service again.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.autocreate:
'{"type":"public","bandwidth_name":"aaaaa","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_g-vm","name":"xxx","available_zone":["xxx"],"elb_virsubnet_ids":["fc0c61cd-c987-49c4-99a4-b7d816b57581"],"l7_flavor_name":"","l4_flavor_name":"L4_flavor.elb.pro.max","vip_subnet_cidr_id":"cf35b03f-c6ca-4f75-aa70-e2166cb1f800"}'
    kubernetes.io/elb.eip-id: 8560972c-2cc5-4699-94d6-e46f146eb73d     # ID of the EIP automatically assigned during load balancer creation
    kubernetes.io/elb.custom-eip-id: 88c197a1-cb85-4b38-b672-1d60dc5d00db  # ID of the custom EIP
    kubernetes.io/elb.custom-eip-status: '{"id":"88c197a1-cb85-4b38-b672-1d60dc5d00db","public_ip_address":"2.2.2.2"}' # After the custom EIP is configured, record the EIP's ID and IP address.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 0e78a84a-7deb-4747-aeb6-09b6a820b001
  labels:
    app: test-svc
    version: v1
  name: test-eip
  namespace: default
spec:
  allocateLoadBalancerNodePorts: true
```



```
  clusterIP: 10.247.93.235
  clusterIPs:
  - 10.247.93.235
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  loadBalancerIP: 2.2.2.2
  ports:
  - name: cce-service-0
    nodePort: 31354
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: test-svc
    version: v1
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 2.2.2.2
    - ip: 192.168.0.15
```

----End

7.3.4.15 Configuring a Range of Listening Ports for LoadBalancer Services

When creating a LoadBalancer Service, you can specify a port range for the ELB listener. This allows the listener to receive requests on ports within the specified range and forward them to the target backend servers.

NOTE

This function relies on ELB capabilities. Before using this function, check whether it is supported in the current region. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.

Prerequisites

- A Kubernetes cluster is available and the cluster version meets the following requirements:
  - v1.23: v1.23.18-r0 or later
  - v1.25: v1.25.13-r0 or later
  - v1.27: v1.27.10-r0 or later
  - v1.28: v1.28.8-r0 or later
  - v1.29: v1.29.4-r0 or later
  - v1.30: v1.30.1-r0 or later
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Precautions

You can configure a range of listening ports only when a dedicated load balancer is used and the TCP, UDP, or TLS protocol is selected.

Using the CCE Console

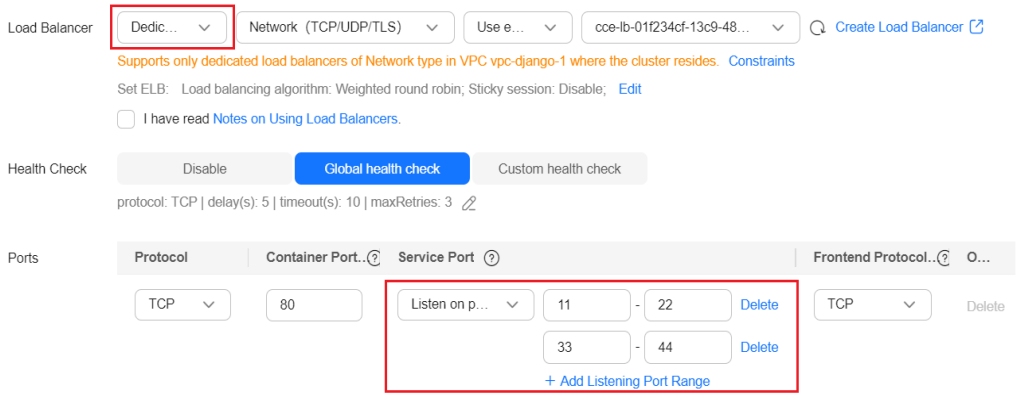
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Services** tab, and click **Create Service** in the upper right corner.

Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Using the Console**.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select a load balancer type and creation mode.
  - In this example, only dedicated load balancers are supported.
  - This section uses an existing load balancer as an example. For details about the parameters for automatically creating a load balancer, see **Table 7-26**.
- **Ports**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Container Port**: listener port of the workload. For example, Nginx uses port 80 by default.
  - **Service Port**: port used by the Service. In this example, select **Listen on ports** and configure a port range. All the requests received on the ports within this range will be forwarded to the backend server. Configure the port range from 1 to 65535. You can add a maximum of 10 non-overlapping ranges of listening ports.
  - **Frontend Protocol**: For a **dedicated load balancer**, to use TLS, the type of the load balancer must be **Network (TCP/UDP/TLS)**.

Figure 7-51 Configuring listening port ranges



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **service-test.yaml**. The file name can be customized.

```
vi service-test.yaml
```

An example YAML file of a Service associated with an existing load balancer is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: service-test
  labels:
    app: test
    version: v1
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance  # A dedicated load balancer is required.
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.port-ranges: '{"cce-service-0":["100,200", "300,400"], "cce-service-1":["500,600", "700,800"]}' # Configure ranges of listening ports.
spec:
  selector:
    app: test
    version: v1
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80          # Replace it with your container port.
      nodePort: 0
      port: 100       # If a port range is configured for listening, this parameter becomes invalid. However, it still needs to be assigned a unique value. By default, it is set to the starting port number of the range.
      protocol: TCP
    - name: cce-service-1
      targetPort: 81          # Replace it with your container port.
      nodePort: 0
      port: 500       # If a port range is configured for listening, this parameter becomes invalid. However, it still needs to be assigned a unique value.
      protocol: TCP
  type: LoadBalancer
  loadBalancerIP: <your_elb_ip> # Replace it with the private IP address of your existing load balancer.
```


Table 7-67 Parameters for listening to ports within a range

Parameter	Mandatory	Type	Description
kubernetes.io/elb.port-ranges	Yes	String	If a dedicated load balancer is used and the TCP, UDP, or TLS protocol is selected, you can create a listener that listens to ports within a certain range from 1 to 65535. You can add a maximum of 10 port ranges that do not overlap for each listener. The parameter value is in the following format, where **ports_name** and **port** must be unique: '{"*<ports_name_1>*":["*<port_1>*,*<port_2>*","*<port_3>*,*<port_4>*"],"*<ports_name_2>*":["*<port_5>*,*<port_6>*","*<port_7>*,*<port_8>*"]}' For example, the port names are **cce-service-0** and **cce-service-1**, and the listener listens to ports 100–200 and 300–400, and 500–600 and 700–800, respectively. '{"cce-service-0":["100,200", "300,400"], "cce-service-1":["500,600", "700,800"]}'

Step 3 Create the Service.

```
kubectl create -f service-test.yaml
```

If information similar to the following is displayed, the Service has been created:

```
service/service-test created
```

----End

7.3.4.16 Setting the Pod Ready Status Through the ELB Health Check

The ready status of the pod is associated with the ELB health check. After the health check is successful, the pod is ready. This association works with the **strategy.rollingUpdate.maxSurge** and **strategy.rollingUpdate.maxUnavailable** parameters of the pod to implement graceful rolling upgrade.

Notes and Constraints

- This feature is available in the following versions:
  - v1.19: v1.19.16-r5 or later
  - v1.21: v1.21.8-r0 or later
  - v1.23: v1.23.6-r0 or later
  - v1.25: v1.25.2-r0 or later
  - Versions later than v1.25
- This function applies only to passthrough scenarios, that is, scenarios where dedicated load balancers are used in CCE Turbo clusters.

- To use this function, configure the readinessGates field in the pod and specify the label **target-health.elb.k8s.cce/***{serviceName}*, where *{serviceName}* indicates the service name.
- The pod ready status takes effect only when the ELB backend is initially connected. The subsequent health check status does not affect the pod ready status.

Setting the Pod Ready Status Through the ELB Health Check

To use pod readiness gates, perform the following steps:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the upper right corner, click **Create from YAML**.

YAML content:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx
  namespace: default
  labels:
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      containers:
        - name: container-1
          image: nginx:latest
      imagePullSecrets:
        - name: default-secret
      readinessGates:
        - conditionType: target-health.elb.k8s.cce/specific-service-name  # Specifies the ServiceName.
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%          # Works with the following two parameters to control the number of ELB backends and implement graceful rolling upgrade.
      maxSurge: 25%
```

Step 3 Click **OK**. On the workload list, you can check the workload status and find that the pod is not ready.

An event is displayed as follows:

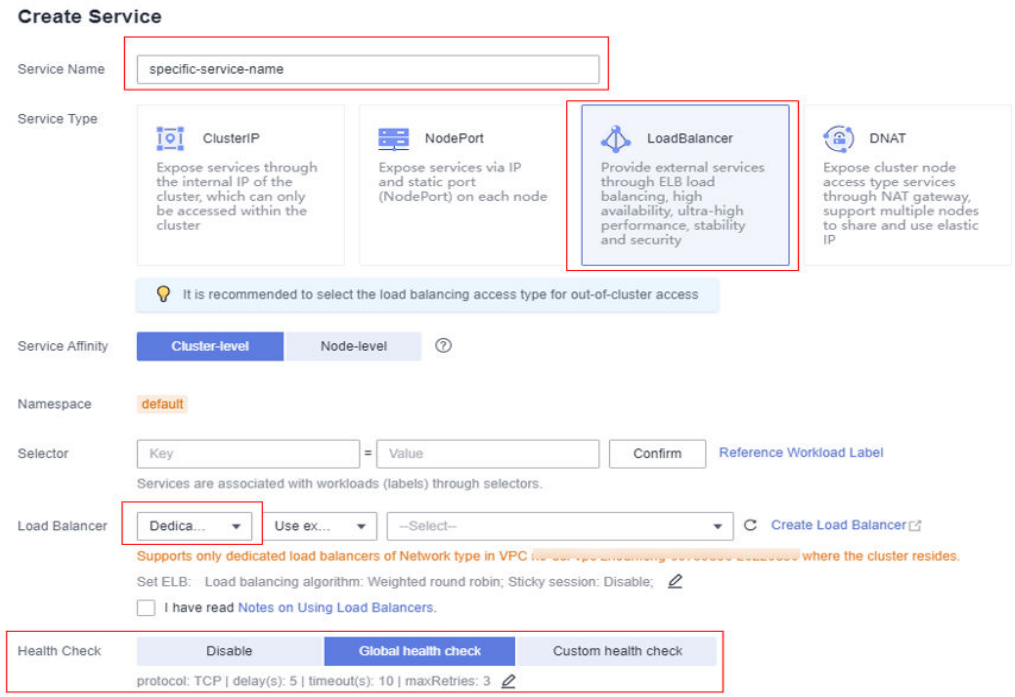
```
Pod not ready: correspondingcondition of pod readinessgate "target-health.elb.k8s.cce/specific-service-name" does not exist.
```

Step 4 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service** and configure the following parameters:

- **Service Name**: The value must be the same as the value of **readinessGates** in the pod.

- **Service Type**: Select **LoadBalancer**.
- **Selector**: Click **Reference Workload Label**, select the workload created in the previous step, and click **OK**.
- **Load Balancer**: Dedicated load balancers must be used. You can select an existing load balancer or automatically create a load balancer.
- **Health Check**: Whether to enable health check. (If it is not configured, the health check is enabled by default.)

Figure 7-52 Configuring a load balancer



Step 5 Go to the ELB console and check the backend server group. The health check status is normal.

Step 6 On the CCE console, the workload is in the **Running** status.

----End

7.3.4.17 Enabling ICMP Security Group Rules

Scenario

If a workload uses UDP for both load balancing and health check, enable ICMP security group rules for the backend servers. For details, see **How Does ELB Perform UDP Health Checks? What Are the Precautions for UDP Health Checks?**

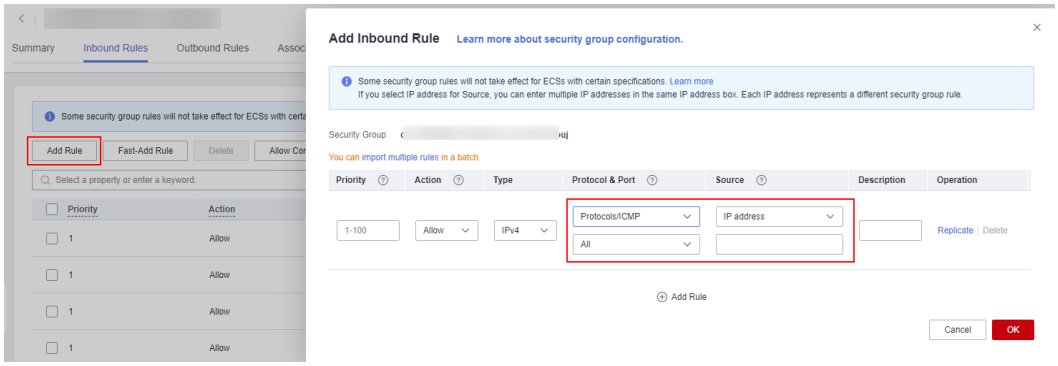
Procedure

Step 1 Log in to the CCE console and choose **Networking** > **Virtual Private Cloud** in the service list. In the navigation pane, choose **Access Control** > **Security Groups**.

Step 2 In the security group list, locate the security group of the cluster. Click the **Inbound Rules** tab page and then **Add Rule**. In the **Add Inbound Rule** dialog box, configure inbound parameters.

Cluster Type	ELB Type	Security Group	Protocol & Port	Allowed Source CIDR Block
CCE Standard	Shared	Node security group, which is named in the format of "{Cluster name}-cce-node-{Random ID}". If a custom node security group is bound to the cluster, select the target security group.	All ICMP ports	100.125.0.0/16 for the shared load balancer
	Dedicated	Node security group, which is named in the format of "{Cluster name}-cce-node-{Random ID}". If a custom node security group is bound to the cluster, select the target security group.	All ICMP ports	Backend subnet of the load balancer
CCE Turbo	Shared	Node security group, which is named in the format of "{Cluster name}-cce-node-{Random ID}". If a custom node security group is bound to the cluster, select the target security group.	All ICMP ports	100.125.0.0/16 for the shared load balancer
	Dedicated	ENI security group, which is named in the format of "{Cluster name}-cce-eni-{Random ID}". If a custom ENI security group is bound to the cluster, select the target security group.	All ICMP ports	Backend subnet of the load balancer

Figure 7-53 Adding a security group rule



Step 3 Click **OK**.

----End

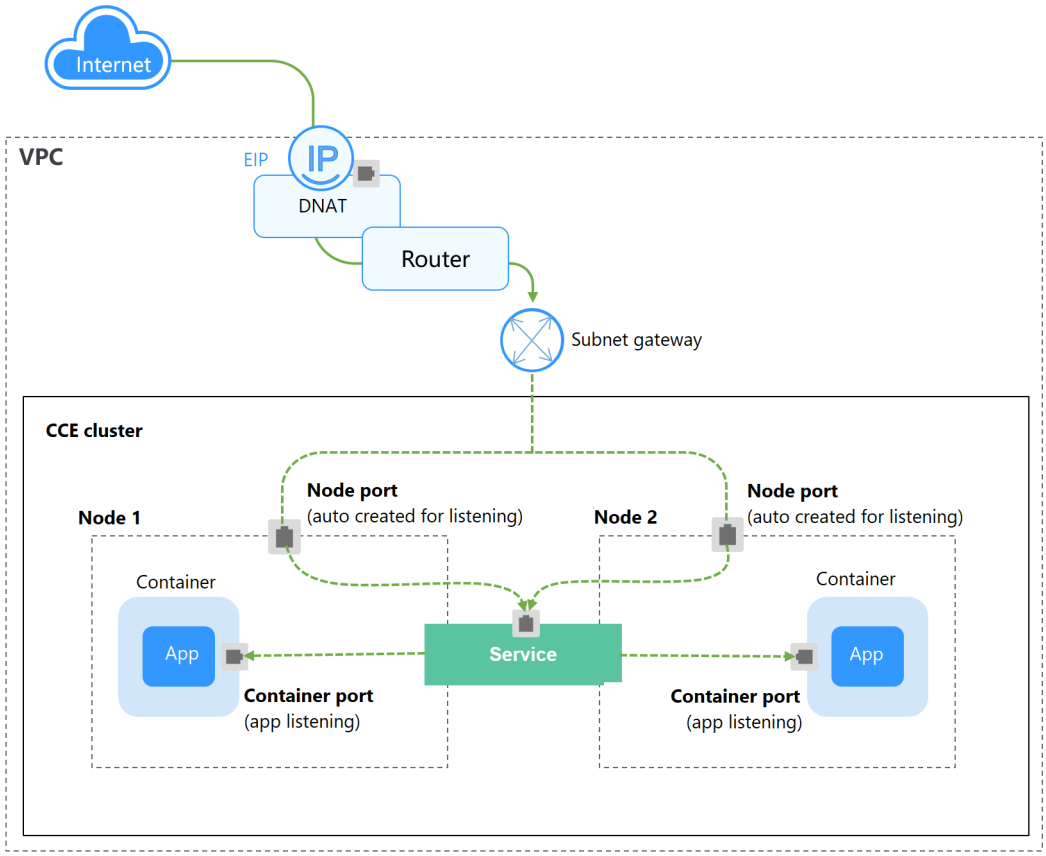
7.3.5 DNAT

Scenario

A **destination network address translation (DNAT) gateway** is situated between cluster nodes and public networks and assigned an EIP. After receiving inbound requests from public networks, the NAT gateway translates the EIP (destination address in the inbound requests) into a cluster-internal address. It appears to workload users as if all nodes running the workload share the same EIP.

DNAT provides higher reliability than EIP-based NodePort in which the EIP is bound to a single node and once the node is down, all inbound requests to the workload will not be distributed. The access address is in the format of <EIP>:<access port>, for example, 10.117.117.117:80.

Figure 7-54 DNAT

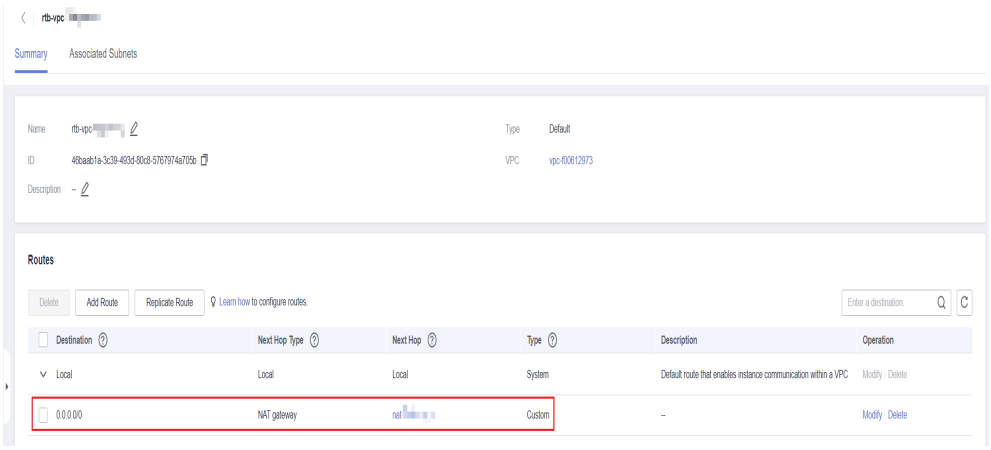


Notes and Constraints

Observe the following constraints when using the NAT Gateway service:

- DNAT rules do not support enterprise project authorization.
- Containers in the cluster cannot access the DNAT Service whose **externalTrafficPolicy** is **Local**.
- Multiple rules for one NAT gateway can use the same EIP, but the rules for different NAT gateways must use different EIPs.
- Each VPC can have only one NAT gateway.
- Users cannot manually add the default route in a VPC.
- Only one SNAT rule can be added to a subnet in a VPC.
- SNAT and DNAT rules are designed for different functions. If SNAT and DNAT rules use the same EIP, resource preemption will occur. An SNAT rule cannot share an EIP with a DNAT rule with **Port Type** set to **All ports**.
- DNAT rules do not support binding an EIP to a virtual IP address.
- When both the EIP and NAT Gateway services are configured for a server, data will be forwarded through the EIP.
- The custom CIDR block must be a subset of the VPC subnet CIDR blocks.
- The custom CIDR block must be a CIDR block of Direct Connect and cannot conflicts with VPC's existing subnet CIDR blocks.

- When you perform operations on underlying resources of an ECS, for example, changing its specifications, the configured NAT gateway rules become invalid. Delete the rules and reconfigure them.
- After a Service is created, if the affinity setting is switched from the cluster level to the node level, the connection tracing table will not be cleared. Do not modify the Service affinity setting after the Service is created. To modify it, create a Service again.
- If the node subnet is associated with a custom route table, add the NAT route to the custom route table when using the DNAT Service.



NOTE

For details about NAT gateway constraints, see **NAT Gateway Constraints**.

Creating a NAT Gateway and an Elastic IP Address

You have created a NAT gateway and an elastic IP address. The specific procedure is as follows:

Step 1 Log in to the management console, choose **Networking** > **NAT Gateway** from the service list, and click **Buy Public NAT Gateway** in the upper right corner. Configure parameters based on site requirements.

NOTE

When buying a NAT gateway, ensure that the NAT gateway belongs to the same VPC and subnet as the CCE cluster where the workload is running.

Step 2 Log in to the management console, choose **Networking** > **Elastic IP** from the service list, and click **Buy EIP** in the upper right corner. Configure parameters based on site requirements.

----End

Creating a DNAT Gateway Service

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. In the upper right corner, click **Create Service**.

Step 3 Configure intra-cluster access parameters.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **DNAT**.
- **Namespace**: Namespace to which the workload belongs.
- **Service Affinity**: For details, see **externalTrafficPolicy (Service Affinity)**.
  - **Cluster level**: The IP addresses and access ports of all nodes in a cluster can access the workload associated with the Service. Service access will cause performance loss due to route redirection, and the source IP address of the client cannot be obtained.
  - **Node level**: Only the IP address and access port of the node where the workload is located can access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **DNAT**: Select the DNAT gateway and EIP created in **Creating a NAT Gateway and an Elastic IP Address**.
- **Ports**
  - **Protocol**: protocol used by the Service.
  - **Container Port**: listener port of the workload. The Nginx workload listens on port 80.
  - **Service Port**: a port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

Step 4 Click **OK**.

----End

Setting the Access Type Using kubectl

You can configure Service access when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-nat-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-nat-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
```



```
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:latest
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

For descriptions of the preceding fields, see **Table 5-2**.

vi nginx-nat-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.class: dnat
    kubernetes.io/natgateway.id: e4a1cfcf-29df-4ab8-a4ea-c05dc860f554
spec:
  loadBalancerIP: 10.78.42.242
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Table 7-68 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	This parameter is set to **dnat** so CCE can work with a NAT gateway and DNAT rules can be added.
kubernetes.io/natgateway.id	Yes	String	ID of a NAT gateway.
loadBalancerIP	Yes	String	EIP ID.
port	Yes	Integer	Access port set on the console. The value ranges from 1 to 65535.
targetPort	Yes	String	Container port set on the console. The value ranges from 1 to 65535.
type	Yes	String	NAT gateway service type must be set to **LoadBalancer**.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.


```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                READY     STATUS            RESTARTS   AGE
nginx-2601814895-sf71t   1/1       Running           0          8s
```

Step 4 Create a Service.

kubectl create -f nginx-nat-svc.yaml

If information similar to the following is displayed, the Service has been created.

```
service "nginx-eip" created
```

kubectl get svc

If the following information is displayed, the Service has been set successfully, and the workload is accessible.

```
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP   10.247.0.1       <none>         443/TCP        3d
nginx-nat    LoadBalancer 10.247.226.2  10.154.74.98   80:30589/TCP   5s
```

Step 5 In the address bar of your browser, enter **10.154.74.98:80** and press **Enter**.

In this example, **10.154.74.98** is the elastic IP address and **80** is the port number obtained in the previous step.

----End

7.3.6 Headless Services

Services allow internal and external pod access, but not the following scenarios:

- Accessing all pods at the same time
- Pods in a Service accessing each other

This is where headless Service come into service. A headless Service does not create a cluster IP address, and the DNS records of all pods are returned during query. In this way, the IP addresses of all pods can be queried. **StatefulSets** use headless Services to support mutual access between pods.

```
apiVersion: v1
kind: Service       # Object type (Service)
metadata:
  name: nginx-headless
  labels:
    app: nginx
spec:
  ports:
    - name: nginx      #     - name: nginx     # Name of the port for communication between pods
      port: 80         # Port number for communication between pods
  selector:
    app: nginx         # Select the pod whose label is app:nginx.
  clusterIP: None      # Set this parameter to None, indicating that a headless Service is to be created.
```

Run the following command to create a headless Service:

```
# kubectl create -f headless.yaml
service/nginx-headless created
```

After the Service is created, you can query the Service.


```
# kubectl get svc
NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
nginx-headless    ClusterIP   None         <none>        80/TCP     5s
```

Create a pod to query the DNS. You can view the records of all pods. In this way, all pods can be accessed.

```
$ kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
If you do not see a command prompt, try pressing Enter.
/ # nslookup nginx-0.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-0.nginx.default.svc.cluster.local
Address: 172.16.0.31

/ # nslookup nginx-1.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-1.nginx.default.svc.cluster.local
Address: 172.16.0.18

/ # nslookup nginx-2.nginx
Server:         10.247.3.10
Address:        10.247.3.10#53
Name:   nginx-2.nginx.default.svc.cluster.local
Address: 172.16.0.19
```

7.4 Ingresses

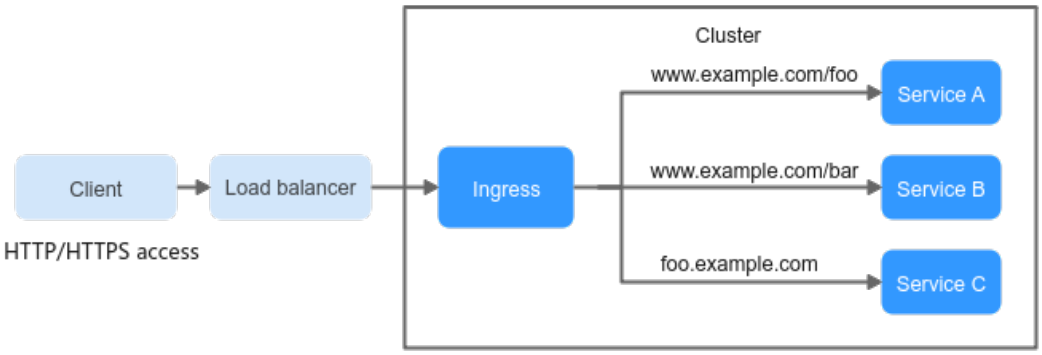
7.4.1 Overview

Why We Need Ingresses

A Service is generally used to forward access requests based on TCP and UDP and provide layer-4 load balancing for clusters. However, in actual scenarios, if there is a large number of HTTP/HTTPS access requests on the application layer, the Service cannot meet the forwarding requirements. Therefore, the Kubernetes cluster provides an HTTP-based access mode, ingress.

An ingress is an independent resource in the Kubernetes cluster and defines rules for forwarding external access traffic. As shown in **Figure 7-55**, you can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic.

Figure 7-55 Ingress diagram



Ingress Overview

Kubernetes uses ingress resources to define how incoming traffic should be handled, while the Ingress Controller is responsible for processing the actual traffic.

- **Ingress object**: a set of access rules that forward requests to specified Services based on domain names or paths. It can be added, deleted, modified, and queried by calling APIs.
- **Ingress Controller**: an executor for forwarding requests. It monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time, parses rules defined by ingresses, and forwards requests to the target backend Services.

  The way of implementing Ingress Controllers varies depending on their vendors. CCE supports LoadBalancer Ingress Controllers and NGINX Ingress Controllers.

  - LoadBalancer Ingress Controllers are deployed on master nodes and they forward traffic based on the ELB. All policy configurations and forwarding behaviors are handled on the ELB.
  - NGINX Ingress Controllers are deployed in clusters using charts and images maintained by the Kubernetes community. They provide external access through NodePort and forward external traffic to other services in the cluster through Nginx. All traffic forwarding behaviors and forwarding objects are within the cluster.

Ingress Feature Comparison

Table 7-69 Comparison between ingress features

Feature	ELB Ingress Controller	Nginx Ingress Controller
O&M	O&M-free	Self-installation, upgrade, and maintenance
Performance	One ingress supports only one load balancer.	Multiple ingresses support one load balancer.
	Enterprise-grade load balancers are used to provide high performance and high availability. Service forwarding is not affected in upgrade and failure scenarios.	Performance varies depending on the resource configuration of pods.

Feature	ELB Ingress Controller	Nginx Ingress Controller
	Dynamic loading is supported.	• Processes must be reloaded for non-backend endpoint changes. This causes loss to persistent connections. • Lua supports hot updates of endpoint changes. • Processes must be reloaded for a Lua modification.
Component deployment	Deployed on the master node	Deployed on worker nodes, and operations costs required for the Nginx component
Route redirection	Supported	Supported
SSL configuration	Supported	Supported
Using ingress as a proxy for backend services	Supported	Supported, which can be implemented through backend-protocol: HTTPS annotations.

The LoadBalancer ingress is essentially different from the open source Nginx Ingress. Therefore, their supported Service types are different. For details, see **Services Supported by LoadBalancer Ingresses**.

LoadBalancer Ingress Controllers are deployed on master nodes. All policy configurations and forwarding behaviors are handled on the ELB. Load balancers outside the cluster can connect to nodes in the cluster only through the IP address of the VPC in non-passthrough networking scenarios. Therefore, LoadBalancer ingresses support only NodePort Services. However, in the passthrough networking scenario where a dedicated load balancer is used in a CCE Turbo cluster, ELB can directly forward traffic to pods in the cluster. In this case, the ingress can only interconnect with ClusterIP Services.

NGINX Ingress Controller runs in a cluster and is exposed as a Service through NodePort. Traffic is forwarded to other Services in the cluster through Nginx ingresses. The traffic forwarding behavior and forwarding object are in the cluster. Therefore, both ClusterIP and NodePort Services are supported.

In conclusion, LoadBalancer ingresses use enterprise-grade load balancers to forward traffic and delivers high performance and stability. NGINX Ingress Controller is deployed on cluster nodes, which consumes cluster resources but has better configurability.

Working Rules of LoadBalancer Ingress Controller

LoadBalancer Ingress Controller developed by CCE implements layer-7 network access for the internet and intranet (in the same VPC) based on ELB and distributes access traffic to the target Services using different paths.

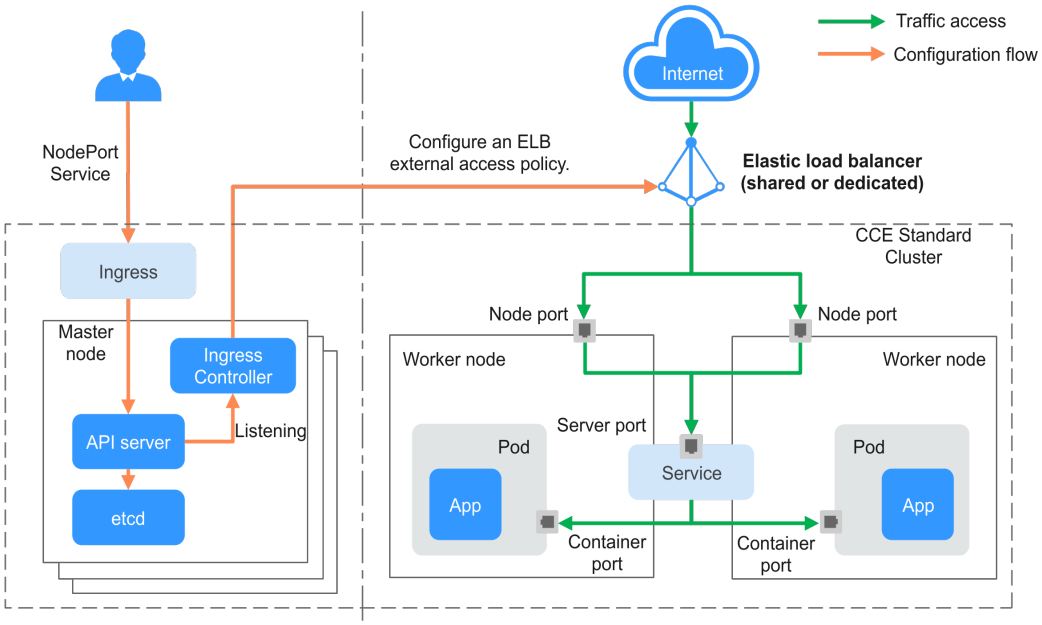
LoadBalancer Ingress Controller is deployed on the master node and bound to the load balancer in the VPC where the cluster resides. Different domain names, ports, and forwarding policies can be configured for the same load balancer (with the same IP address). The working rules of LoadBalancer Ingress Controller are as follows:

1. A user creates an ingress and configures a traffic access rule in the ingress, including the load balancer, access path, SSL, and backend Service port.
2. When Ingress Controller detects that the ingress changes, it reconfigures the listener and backend server route on the ELB according to the traffic access rule.
3. When a user attempts to access a workload, the ELB forwards the traffic to the target workload according to the configured forwarding rule.

The way LoadBalancer Ingress Controller works depends on the type of cluster and ELB being used. The following section describes the configuration process and network flow in various scenarios.

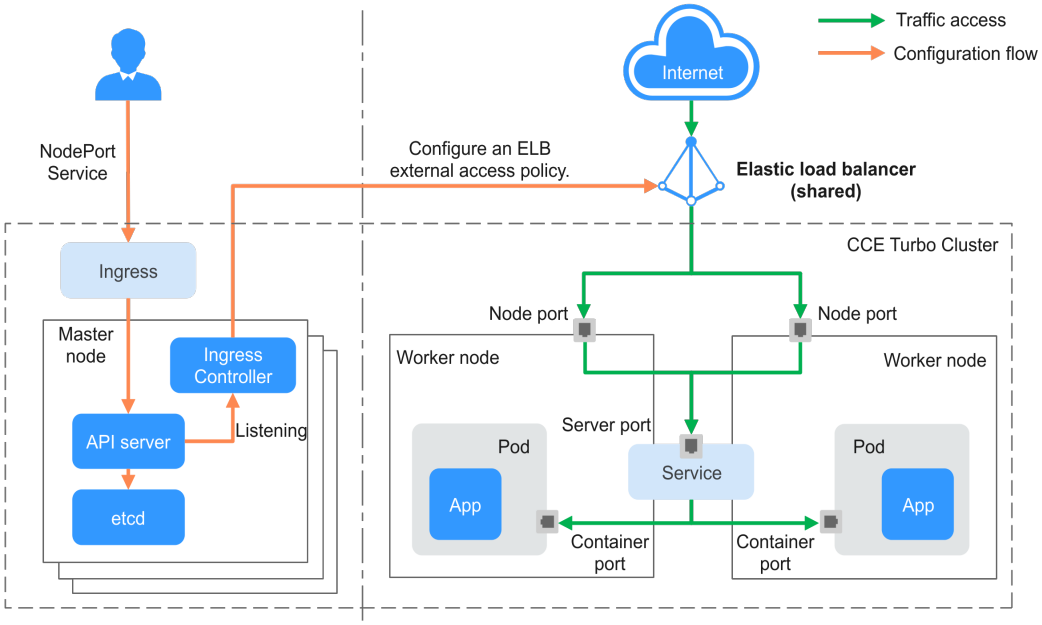
CCE Standard Clusters

Figure 7-56 Working flow of a LoadBalancer ingress in a CCE standard cluster



CCE Turbo Clusters Where a Shared Load Balancer Is Used

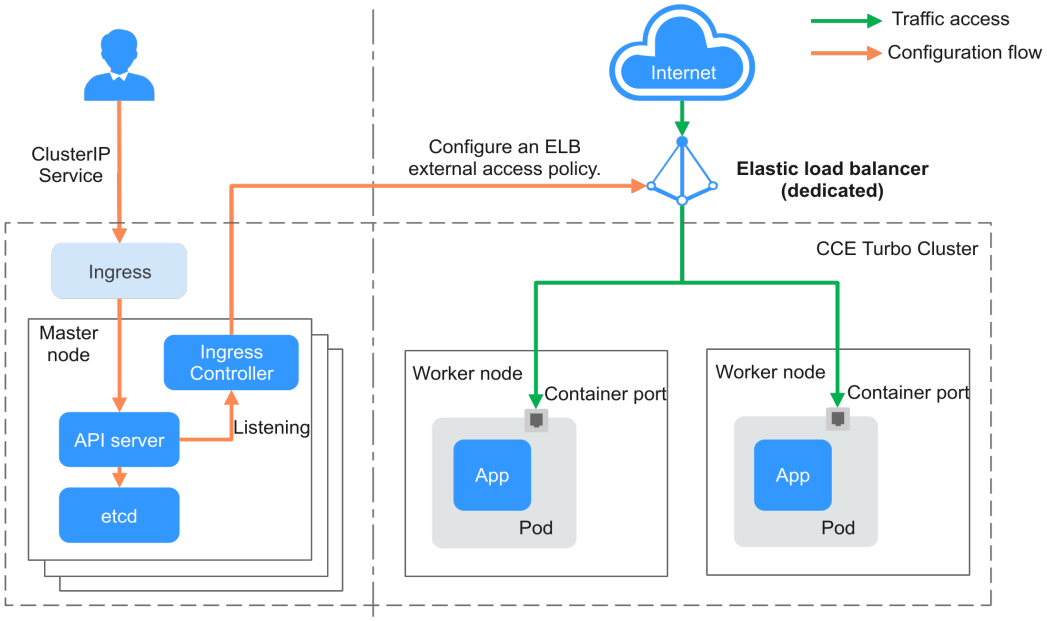
Figure 7-57 Working flow of a LoadBalancer ingress in a CCE Turbo cluster where a shared load balancer is used



CCE Turbo Clusters Where a Dedicated Load Balancer Is Used

When a **CCE Turbo cluster** is used, pod IP addresses are directly allocated from the VPC. **Dedicated load balancers** enable passthrough networking to pods. When creating an ingress for external cluster access, you can use ELB to access a ClusterIP Service and use pods as the backend server of the ELB listener. In this way, external traffic can directly access the pods in the cluster without being forwarded by node ports.

Figure 7-58 Working flow of a LoadBalancer ingress in a CCE Turbo cluster where a dedicated load balancer is used



Working Rules of NGINX Ingress Controller

Nginx Ingress uses ELB as the traffic ingress. The **NGINX Ingress Controller** add-on is deployed in a cluster to balance traffic and control access.

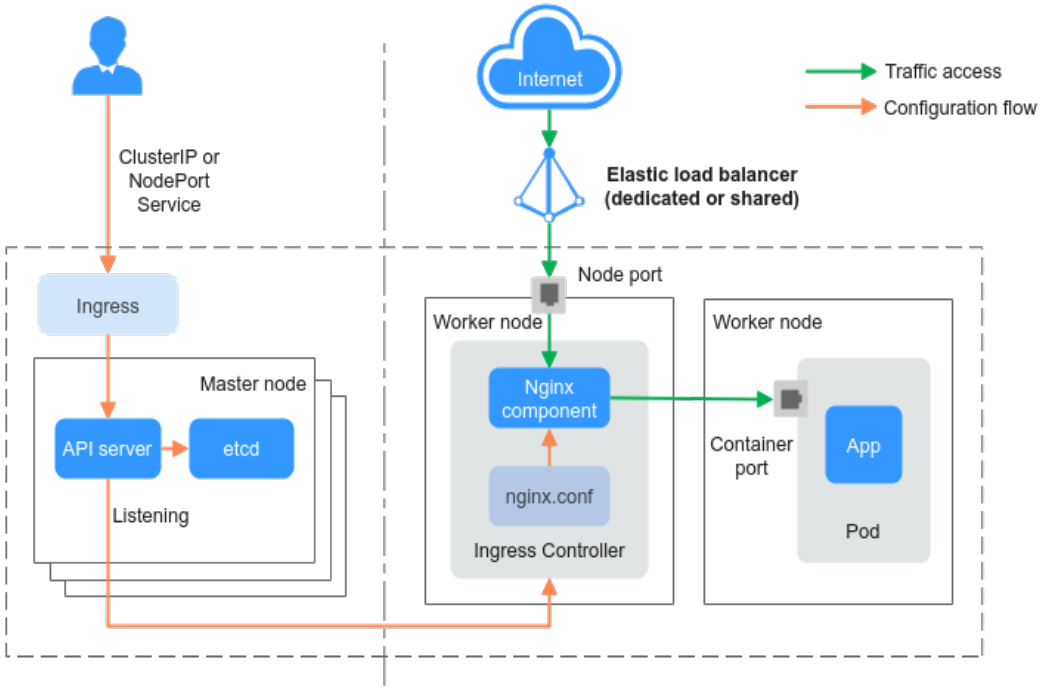
NOTE

NGINX Ingress Controller uses the charts and images provided by the **open-source community**, and issues may occur during usage. CCE periodically synchronizes the community version to fix known vulnerabilities. Check whether your service requirements can be met.

NGINX Ingress Controller is deployed on worker nodes through pods, which will result in O&M costs and Nginx component running overheads. **Figure 7-59** shows the working rules of NGINX Ingress Controller.

1. After you update ingress resources, NGINX Ingress Controller writes a forwarding rule defined in the ingress resources into the **nginx.conf** configuration file of Nginx.
2. The built-in Nginx component reloads the updated configuration file to modify and update the Nginx forwarding rule.
3. When traffic accesses a cluster, the traffic is first forwarded by the created load balancer to the Nginx component in the cluster. Then, the Nginx component forwards the traffic to each workload based on the forwarding rule.

Figure 7-59 Working rules of NGINX Ingress Controller



Services Supported by Ingresses

LoadBalancer and the open-source Nginx ingresses support different Services due to their implementation principles.

Services Supported by LoadBalancer Ingresses

Table 7-70 Services supported by LoadBalancer ingresses

Cluster Type	ELB Type	ClusterIP	NodePort
CCE standard cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Not supported	Supported
CCE Turbo cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Supported	Not supported **NOTE** ENIs are separately bound to pods in a CCE Turbo cluster, and ELB directly connects to pods. Therefore, NodePort access is not available.

Services Supported by Nginx Ingresses

Table 7-71 Services supported by Nginx ingress

Cluster Type	ELB Type	ClusterIP	NodePort
CCE standard cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported
CCE Turbo cluster	Shared load balancer	Supported	Supported
	Dedicated load balancer	Supported	Supported

7.4.2 Comparison Between ELB Ingress and Nginx Ingress

In CCE, clusters can use Nginx Ingress and ELB Ingress to enable layer-7 network access for applications.

- Nginx Ingress, enhanced by CCE using the NGINX Ingress Controller from the community, regularly updates community features and bug fixes. Nginx Ingress provides various configuration options, catering to users with advanced gateway customization requirements.
- ELB Ingress, backed by ELB, offers fully managed and O&M-free services. It can handle tens of millions of concurrent connections and millions of new connections. ELB Ingress supports the interconnection with both shared and dedicated load balancers.

This section describes the differences between Nginx Ingress and ELB Ingress.

Typical Application Scenarios

Type	Feature
Nginx Ingress	• Standard configurations • Extensive gateway customization • Canary release and blue-green deployment of cloud native applications
ELB Ingress	• Hosted gateway that is highly available and O&M-free • Layer 7 high-performance auto scaling of cloud native applications • Canary release and blue-green deployment of cloud native applications • Isolated resources for dedicated use. A load balancer deployed in a single AZ can handle up to 20 million concurrent connections, making it ideal for managing a large volume of requests.

Functions

Item	Nginx Ingress	ELB Ingress
Positioning	Layer 7 traffic governance offers various advanced routing functions.	• Layer 7 traffic governance offers various advanced routing functions. It seamlessly incorporates cloud-native technologies to deliver fully managed load balancing services that are O&M-free, highly available, high-performance, ultra-secure, and support multiple protocols. • Computing resources can be scaled to handle traffic surges. • ELB can handle tens of millions of concurrent connections and millions of new connections.
Basic routing	• Routing can be based on content and source IP addresses. • HTTP header modification, redirection, rewriting, rate limiting, cross-region routing, and sticky sessions are available. • Forwarding rules can be configured for both requests and responses, and the rules for responses can be configured using extended Snippet. • Forwarding rules are matched based on the longest path. If multiple paths are matched, the longest forwarding path is prioritized.	• Routing can be based on content and source IP addresses. • HTTP header modification, redirection, rewriting, rate limiting, and sticky sessions are available. • Forwarding rules can be configured for both requests and responses. • Forwarding rules are prioritized in descending order. If multiple paths are matched, a lower value indicates a higher priority.
Protocol	• HTTP and HTTPS • WebSocket, WSS, and gRPC	• HTTP and HTTPS • gRPC

Item	Nginx Ingress	ELB Ingress
Configuration modification	• Processes must be reloaded for non-backend endpoint changes. This causes loss to persistent connections. • Lua supports hot updates of endpoint changes. • Processes must be reloaded for a Lua modification.	The declarative OpenAPI between cloud services enables the dynamic loading of modified configurations to ELB.
Authentication	• Basic authentication • OAuth	TLS authentication
Performance	• Both system and Nginx parameters require manual optimization for performance tuning. • To ensure proper system running, you must configure a proper number of replicas and resource limits. For more information, see **Creating an Nginx Ingress on the Console**.	ELB can handle tens of millions of concurrent connections and millions of new connections.
Observability	• Log collection through Access Log • Monitoring and alarm configuration through Prometheus	• Log access for cloud services through interconnected LTS • Auditing key operations • Metrics-backed monitoring through interconnected Cloud Eye • Alarm rules configurable through interconnected Cloud Eye
O&M	• Bring-your-own component maintenance and periodic version synchronization from the community • Scaling through HPA • Proactive configuration for optimization	• Fully managed and O&M-free • Configuration-free automatic scaling for ultra-large capacity • Auto scaling based on service traffic

Item	Nginx Ingress	ELB Ingress
Security	• HTTPS • Blocklists and trustlists	• SSL-integrated HTTPS for full-link HTTPS, SNI multi-certificate, RSA, ECC dual-certification, TLS 1.3, and TLS algorithm suites • WAF • Anti-DDoS • Blocklists and trustlists • Custom security policies
Service governance	• Kubernetes-backed service discovery • Canary release • Traffic limit for service HA	• Kubernetes-backed service discovery • Canary release

7.4.3 LoadBalancer Ingresses

7.4.3.1 Creating a LoadBalancer Ingress on the Console

In Kubernetes, an ingress is a resource object that controls how Services within a cluster can be accessed from outside the cluster. You can use ingresses to configure different forwarding rules to access pods in a cluster. The following uses an **Nginx workload** as an example to describe how to create a LoadBalancer ingress on the console.

Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- It is recommended that other resources not use the load balancer automatically created by an ingress. Otherwise, the load balancer will be occupied when the ingress is deleted, resulting in residual resources.
- After an ingress is created, upgrade and maintain the configuration of the selected load balancers on the CCE console. Do not modify the configuration on the ELB console. Otherwise, the ingress service may be abnormal.
- The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.

- If you have an IPVS-backed cluster and use the same ELB load balancer for both the ingress and Service in the same cluster or for the Service ports in different clusters, you will not be able to access the ingress from the nodes or containers in the cluster, or the Service ports in other clusters. This is because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge, which intercepts the load balancer traffic. Use separate load balancers for these ingresses and Services.
- Do not connect an ingress and **a Service that uses HTTP** to the same listener of the same load balancer. Otherwise, a port conflict occurs.
- A dedicated load balancer must be of the application type (HTTP/HTTPS) and support private networks (with a private IP address).
- If multiple ingresses access the same ELB port in a cluster, the listener configuration items (such as the certificate associated with the listener and the HTTP/2 attribute of the listener) are subject to the configuration of the first ingress.

Adding a LoadBalancer Ingress

This section uses an Nginx workload as an example to describe how to add a LoadBalancer ingress.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

- **Name**: Customize the name of an ingress, for example, **ingress-demo**.
- **Interconnect with Nginx**: This option is displayed only after the **NGINX Ingress Controller** add-on is installed. If this option is available, the NGINX Ingress Controller add-on has been installed. Enabling this option will create an Nginx ingress. Disable it if you want to create a LoadBalancer ingress. For details, see **Creating an Nginx Ingress on the Console**.
- **Load Balancer**: Select a load balancer type and creation mode.

  A load balancer can be dedicated or shared. A dedicated load balancer must be of the application type (HTTP/HTTPS) and support private networks.

  You can select **Use existing** or **Auto create** to obtain a load balancer. For details about the configuration of different creation modes, see **Table 7-72**.

Table 7-72 Load balancer configurations

How to Create	Configuration
Use existing	Only the load balancers in the same VPC as the cluster can be selected. If no load balancer is available, click **Create Load Balancer** to create one on the ELB console.

How to Create	Configuration
Auto create	– **Instance Name**: Enter a load balancer name. – **Enterprise Project**: This parameter is available only for enterprise users who have enabled an enterprise project. Enterprise projects facilitate project-level management and grouping of cloud resources and users. – **AZ**: available only to dedicated load balancers. You can create load balancers in multiple AZs to improve service availability. You can deploy a load balancer in multiple AZs for high availability. – **Frontend Subnet**: available only to dedicated load balancers. It is used to allocate IP addresses for load balancers to provide services externally. – **Backend Subnet**: available only to dedicated load balancers. It is used to allocate IP addresses for load balancers to access the backend service. – **Network Specifications**, **Application-oriented Specifications**, or **Specifications** (available only to dedicated load balancers) ▪ **Fixed**: applies to stable traffic, billed based on specifications. – **EIP**: If you select **Auto create**, you can configure the billing mode and size of the public network bandwidth. – **Resource Tag**: You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency.

- **Listener**: An ingress configures a listener for the load balancer, which listens to requests from the load balancer and distributes traffic. After the configuration is complete, a listener is created on the load balancer. The default listener name is *k8s_<Protocol type>_<Port number>*, for example, *k8s_HTTP_80*.
  - **External Protocol**: **HTTP** and **HTTPS** are available.
  - **External Port**: port number that is open to the ELB service address. The port number is configurable.
  - **Access Control**
    - **Inherit ELB Configurations**: CCE does not modify the existing access control configurations on the ELB console.
    - **Allow all IP addresses**: No access control is configured.
    - **Trustlist**: Only the selected IP address group can access the load balancer.

- **Blocklist**: The selected IP address group cannot access the load balancer.

  NOTE

  For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can select a maximum of five IP address groups for access control at a time.

– **Certificate Source**: TLS secret and ELB server certificates are supported.

  - **TLS secret**: For details about how to create a secret certificate, see **Creating a Secret**.
  - **ELB server certificate**: Use a certificate created on ELB.

– **Server Certificate**: When an HTTPS listener is created for a load balancer, bind a certificate to the load balancer to support encrypted authentication for HTTPS data transmission.

  NOTE

  If there is already an HTTPS ingress for the chosen port on the load balancer, the certificate of the new HTTPS ingress must be the same as the certificate of the existing ingress. This means that a listener has only one certificate. If two certificates, each with a different ingress, are added to the same listener of the same load balancer, only the certificate added earliest takes effect on the load balancer.

– **SNI**: stands for Server Name Indication (SNI), which is an extended protocol of TLS. SNI allows multiple TLS-compliant domain names for external access using the same IP address and port number, and different domain names can use different security certificates. After SNI is enabled, the client is allowed to submit the requested domain name when initiating a TLS handshake request. After receiving the TLS request, the load balancer searches for the certificate based on the domain name in the request. If the certificate corresponding to the domain name is found, the load balancer returns the certificate for authorization. Otherwise, the default certificate (server certificate) is returned for authorization.

  NOTE

  - The **SNI** option is available only when **HTTPS** is used.
  - This function is supported only in clusters of v1.15.11 or later.
  - Only one domain name can be specified for each SNI certificate. Wildcard-domain certificates are supported.
  - For ingresses connected to the same ELB port, do not configure SNIs with the same domain name but different certificates. Otherwise, the SNIs will be overwritten.

– **Security Policy**: combinations of different TLS versions and supported cipher suites available to HTTPS listeners.

  For details about security policies, see **TLS Security Policy**.

NOTE

- **Security Policy** is available only when **HTTPS** is selected.
- This function is supported only in clusters of v1.17.9 or later.

– **Backend Protocol**:

When the **listener** is HTTP-compliant, only **HTTP** can be selected.

If it is an **HTTPS listener**, this parameter can be set to **HTTP**, **gRPC**, or **HTTPS**. Only dedicated load balancers support gRPC. After HTTP/2 is enabled, CCE will automatically add the **kubernetes.io/elb.http2-enable:true** annotation. gRPC is available only in certain regions. For details, see the CCE console.

– **Advanced Options**

Configuration	Description	Restrictions
Transfer Listener Port Number	If this function is enabled, the listening port on the load balancer can be transferred to backend servers through the HTTP header of the packet.	This function is available only for dedicated load balancers.
Transfer Port Number in the Request	If this function is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet.	This function is available only for dedicated load balancers.
Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.	This function is available only for dedicated load balancers.
Data Compression	If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. ▪ Brotli can compress all file formats. ▪ GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json	This function is available only for dedicated load balancers.

Configuration	Description	Restrictions
Idle Timeout	Timeout for an idle client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request.	This function is unavailable for the UDP listener of a shared load balancer.
Request Timeout	Timeout for waiting for a request from a client. There are two cases: ■ If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. ■ If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected.	This function is only for HTTP and HTTPS listeners.
Response Timeout	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.	This function is available only for HTTP and HTTPS listeners.
HTTP2	Whether to use HTTP/2 for a client to communicate with a load balancer. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server.	This function is available only when the **listener** is HTTPS-compliant.

– **Redirect to HTTPS**: When **HTTP** is selected for **External Protocol**, traffic can be redirected to HTTPS. After this function is enabled, you can click

Modify to configure the HTTPS port. For details, see the HTTPS-compliant **listener configuration**.

- **Gray release**: After an ingress is created, you can create a grayscale release policy in the **Operation** column of the ingress. For details, see **Configuring Grayscale Release for a LoadBalancer Ingress**.
- **Forwarding Policy**: When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL, for example, 10.117.117.117:80/helloworld), the request is forwarded to the corresponding target Service for processing. You can click + to add multiple forwarding policies.
  - **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
  - **Path Matching Rule**:
    - **Prefix match**: If the URL is set to **/healthz**, the URL that meets the prefix can be accessed, for example, **/healthz/v1** and **/healthz/v2**.
    - **Exact match**: The URL can be accessed only when it is fully matched. For example, if the URL is set to **/healthz**, only /healthz can be accessed.
    - **RegEX match**: The URL is matched based on the regular expression. For example, if the regular expression is **/[A-Za-z0-9_.-]+/test**, all URLs that comply with this rule can be accessed, for example, **/abcA9/test** and **/v1-Ab/test**. Two regular expression standards are supported: POSIX and Perl.
  - **Path**: access path, for example, **/healthz**

    NOTE

    The access path added here must exist in the backend application. Otherwise, the forwarding fails.

    For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.
  - **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. For details, see **Services Supported by Ingresses**.
  - **Destination Service Port**: Select the access port of the destination Service.
  - **Set ELB**:
    - **Algorithm**: Three algorithms are available: weighted round robin, weighted least connections algorithm, or source IP hash.

NOTE

- **Weighted round robin**: Requests are forwarded to different servers based on their weights, which indicate server processing performance. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. This algorithm is often used for short connections, such as HTTP services.
- **Weighted least connections**: In addition to the weight assigned to each server, the number of connections processed by each backend server is considered. Requests are forwarded to the server with the lowest connections-to-weight ratio. Building on **least connections**, the **weighted least connections** algorithm assigns a weight to each server based on their processing capability. This algorithm is often used for persistent connections, such as database connections.
- **Source IP hash**: The source IP address of each request is calculated using the hash algorithm to obtain a unique hash key, and all backend servers are numbered. The generated key allocates the client to a particular server. This enables requests from different clients to be distributed in load balancing mode and ensures that requests from the same client are forwarded to the same server. This algorithm applies to TCP connections without cookies.

- **Sticky Session**: This function is disabled by default. Options are as follows:
  - **Load balancer cookie**: Enter the **Stickiness Duration** , which ranges from 1 to 1440 minutes.
  - **Application cookie**: This parameter is available only for shared load balancers. In addition, enter **Cookie Name**, which ranges from 1 to 64 characters.

  NOTE

  - When the **distribution policy** uses the source IP hash, sticky session cannot be set.
  - Dedicated load balancers in the clusters of a version earlier than v1.21 do not support sticky sessions. If sticky sessions are required, use shared load balancers.

- **Health Check**: Set the health check configuration of the load balancer. If this function is enabled, the following configurations are supported:

Parameter	Description
Protocol	When the protocol of the target Service port is TCP, more protocols including gRPC and HTTP are supported. Only dedicated load balancers support gRPC. gRPC is available only in certain regions. For details, see the CCE console. ○ **Check Path** (supported only by HTTP or gRPC for health check): specifies the health check URL. The check path must start with a slash (/) and contain 1 to 80 characters.

Parameter	Description
Port	By default, the service port (NodePort or container port of the Service) is used for health check. You can also specify another port for health check. After the port is specified, a service port named **cce-healthz** will be added for the Service. ○ **Node Port**: If a shared load balancer is used or no ENI instance is associated, the node port is used as the health check port. If this parameter is not specified, a random port is used. The value ranges from 30000 to 32767. ○ **Container Port**: When a dedicated load balancer is associated with an ENI instance, the container port is used for health check. The value ranges from 1 to 65535.
Check Period (s)	Specifies the maximum interval between health checks. The value ranges from 1 to 50.
Timeout (s)	Specifies the maximum timeout duration for each health check. The value ranges from 1 to 50.
Max. Retries	Specifies the maximum number of health check retries. The value ranges from 1 to 10.

  - **Operation**: Click **Delete** to delete the configuration.
  - **Actions**: **Redirect to URL** and **Rewrite URL** are available only for dedicated load balancers.
    - **Redirect to URL**: When an access request meets the forwarding policy, the request will be redirected to a specified URL, and a specific status code will be returned.
    - **Rewrite URL**: When an access request meets the forwarding policy, the URL will be rewritten based on the matching rule. You are allowed to configure a regular expression for the patch matching rule, and the result obtained using the regular expression can be used for rewriting the URL. For example, the regular expression configured in the forwarding rule is **/first/(.*)/(.*)/end**, and the rewrite URL is set to **/${1}/${2}**. When the client URL for sending requests is **/first/aaa/bbb/end**, the forwarding rule matches **/first/(.*)/(.*)/end**. Then, **${1}** in the rewrite URL will be replaced with **aaa** and **${2}** replaced with **bbb**, the request path received by the backend server will be **/aaa/bbb**.

- **Annotation**: Ingresses provide some advanced CCE functions, which are implemented by annotations. When you use kubectl to create a container, annotations will be used. For details, see **Automatically Creating a Load Balancer While Creating an Ingress** or **Associating an Existing Load Balancer to an Ingress While Creating the Ingress**.

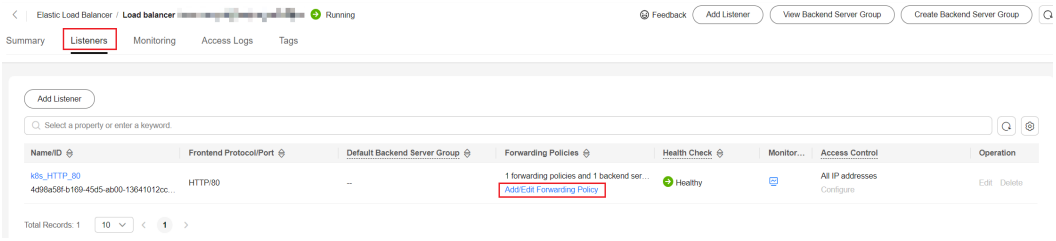
Step 4 Click **OK**. After the ingress is created, it is displayed in the ingress list.

On the ELB console, you can check the load balancer automatically created through CCE. The default name is **cce-lb-<ingress.UID>**. Click the load balancer name to go to the details page. On the **Listeners** tab page, check the listener and forwarding policy of the target ingress.

NOTICE

After an ingress is created, upgrade and maintain the selected load balancer on the CCE console. Do not modify the configuration on the ELB console. Otherwise, the ingress service may be abnormal.

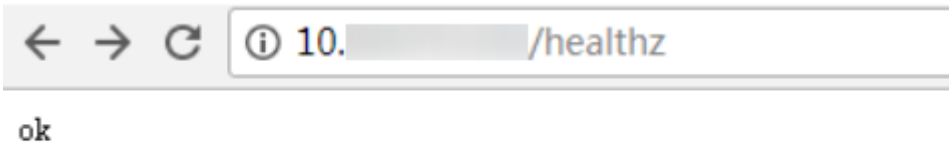
Figure 7-60 LoadBalancer ingress configuration



Step 5 Access the /healthz interface of the workload, for example, workload **defaultbackend**.

1. Obtain the access address of the **/healthz** interface of the workload. The access address consists of the load balancer IP address, external port, and mapping URL, for example, 10.**.**.**:80/healthz.
2. Enter the URL of the /healthz interface, for example, http://10.**.**.**:80/healthz, in the address box of the browser to access the workload, as shown in **Figure 7-61**.

Figure 7-61 Accessing the /healthz interface of defaultbackend



----End

Related Operations

The Kubernetes ingress structure does not contain the **property** field. Therefore, the ingress created by the API called by client-go does not contain the **property** field. CCE provides a solution to ensure compatibility with the Kubernetes client-go. For details about the solution, see **How Can I Achieve Compatibility Between Ingress's property and Kubernetes client-go?**

7.4.3.2 Creating a LoadBalancer Ingress Using kubectl

This section uses an **Nginx workload** as an example to describe how to create a LoadBalancer ingress using kubectl.

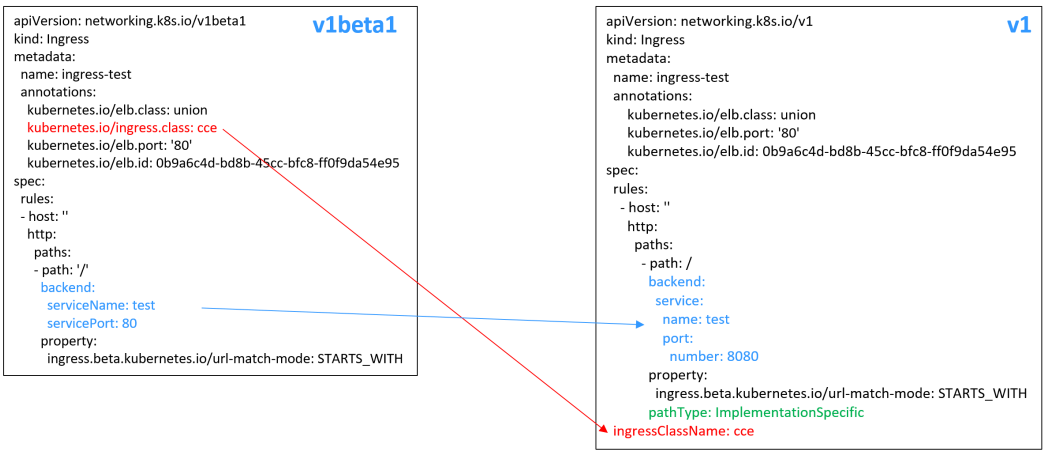
- If no load balancer is available in the same VPC, CCE can automatically create a load balancer when creating an ingress. For details, see **Automatically Creating a Load Balancer While Creating an Ingress**.
- If a load balancer is available in the same VPC, perform the operation by referring to **Associating an Existing Load Balancer to an Ingress While Creating the Ingress**.

Ingress API Version Upgrade in CCE Clusters v1.23

In CCE clusters of v1.23 or later, the ingress version is switched to **networking.k8s.io/v1**.

Compared with v1beta1, v1 has the following differences in parameters:

- The ingress type is specified by **spec.ingressClassName** instead of **kubernetes.io/ingress.class** in **annotations**.
- The format of **backend** has changed.
- The **pathType** parameter must be specified for each path. The options are as follows:
  - **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE, which is the same as v1beta1.
  - **Exact**: exact matching of the URL, which is case-sensitive.
  - **Prefix**: matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by one. A path element refers to a list of labels in the path separated by a slash (/).



Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.

- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- A dedicated load balancer must be of the application type (HTTP/HTTPS) and support private networks (with a private IP address).

Creating an Ingress Using kubectl

You can determine whether to automatically create a load balancer or use an existing one when creating an ingress.

Automatically Creating a Load Balancer While Creating an Ingress

The following describes how to run the kubectl command to automatically create a load balancer when creating an ingress.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

NOTE

Starting from cluster v1.23, the ingress version is switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see **Ingress API Version Upgrade in CCE Clusters v1.23**.

Example of a shared load balancer (public network access) for clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type":"public",
          "bandwidth_name":"cce-bandwidth-******",
          "bandwidth_chargemode":"bandwidth",
          "bandwidth_size":5,
          "bandwidth_sharetype":"PER",
          "vip_subnet_cidr_id":  "******",
          "vip_address":  "**.**.**.**",
          "eip_type":"5_bgp"
      }'
    kubernetes.io/elb.tags: key1=value1,key2=value2          # ELB resource tags
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: <your_service_port>  # Replace it with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```



```
      pathType: ImplementationSpecific
  ingressClassName: cce    # A LoadBalancer ingress is used.
```

Example of a shared load balancer (public network access) for clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/ingress.class: cce    # A LoadBalancer ingress is used.
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type":"public",
          "bandwidth_name":"cce-bandwidth-******",
          "bandwidth_chargemode":"bandwidth",
          "bandwidth_size":5,
          "bandwidth_sharetype":"PER",
          "eip_type":"5_bgp"
        }'
    kubernetes.io/elb.tags: key1=value1,key2=value2          # ELB resource tags
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: <your_service_port>  # Replace it with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Example of a dedicated load balancer (public network access) for clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "vip_subnet_cidr_id": "******",
          "vip_address": "**.**.**.**",
          "elb_virsubnet_ids":["******"],
          "available_zone": [
              "ap-southeast-1a"
          ],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tags: key1=value1,key2=value2          # ELB resource tags
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
```



```
        service:
          name: <your_service_name>  # Replace it with the name of your target Service.
          port:
            number: <your_service_port>  # Replace it with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Example of a dedicated load balancer (public network access) for clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "elb_virsubnet_ids":["******"],
          "available_zone": [
              "ap-southeast-1a"
          ],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tags: key1=value1,key2=value2          # ELB resource tags
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: <your_service_port>  # Replace it with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-73 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. Options: • **union**: shared load balancer • **performance**: dedicated load balancer. For details, see **Differences Between Shared and Dedicated Load Balancers.** Default: **union**

Parameter	Mandatory	Type	Description
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	**cce**: A proprietary LoadBalancer ingress is used. This parameter is mandatory when an ingress is created by calling the API.
ingressClassName	Yes (only for clusters of v1.23 or later)	String	**cce**: A proprietary LoadBalancer ingress is used. This parameter is mandatory when an ingress is created by calling the API.
kubernetes.io/elb.port	Yes	String	This parameter indicates the external port registered with the address of the LoadBalancer Service. The value ranges from 1 to 65535. **NOTE** Some ports are high-risk ports and are blocked by default, for example, port 21.
kubernetes.io/elb.subnet-id	None	String	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters later than v1.11.7-r0. It is left blank by default. For details about how to obtain the value, see **What Is the Difference Between the VPC Subnet API and the OpenStack Neutron Subnet API?**

Parameter	Mandatory	Type	Description
kubernetes.io/elb.enterpriseID	No	String	**Kubernetes clusters of v1.15 and later versions support this field. In Kubernetes clusters earlier than v1.15, load balancers are created in the default project by default.** ID of the enterprise project in which the load balancer will be created. The value contains 1 to 100 characters. **How to obtain**: Log in to the management console and choose **Enterprise** > **Project Management** on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.
kubernetes.io/elb.autocreate	Yes	elb.autocreate object	Whether to automatically create a load balancer associated with an ingress. For details about the field description, see **Table 7-74**. **Example** • Automatically created shared load balancer with an EIP bound: '{"type":"public","bandwidth_name":"cce-bandwidth-******","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • Automatically created shared load balancer: {"type":"inner","name":"A-location-d-test"}

Parameter	Mandatory	Type	Description
kubernetes.io/elb.tags	No	String	Whether to add resource tags to a load balancer. This function is available only when the load balancer is automatically created, and the cluster is of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later. A tag is in the format of "key=value". Use commas (,) to separate multiple tags.
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
path	Yes	String	User-defined route path. All external access requests must match **host** and **path**. **NOTE** The access path added here must exist in the backend application. Otherwise, the forwarding fails. For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.
ingress.beta.kubernetes.io/url-match-mode	No	String	Route matching policy. Default: **STARTS_WITH** (prefix match) Options: • **EQUAL_TO**: exact match • **STARTS_WITH**: prefix match • **REGEX**: regular expression match

Parameter	Mandatory	Type	Description
pathType	Yes	String	Path type. This field is supported only by clusters of v1.23 or later. • **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE. • **Exact**: exact matching of the URL, which is case-sensitive. • **Prefix**: prefix matching, which is case-sensitive. With this method, the URL path is separated into multiple elements by slashes (/) and the elements are matched one by one. If each element in the URL matches the path, the subpaths of the URL can be routed normally. **NOTE** – During prefix matching, each element must be exactly matched. If the last element of the URL is the substring of the last element in the request path, no matching is performed. For example, **/foo/bar** matches **/foo/bar/baz** but does not match **/foo/barbaz**. – When elements are separated by slashes (/), if the URL or request path ends with a slash (/), the slash (/) at the end is ignored. For example, **/foo/bar** matches **/foo/bar/**. See **examples** of ingress path matching.

Table 7-74 elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: **cce-lb+service.UID**
type	No	String	Network type of the load balancer. ● **public**: public network load balancer ● **inner**: private network load balancer Default: **inner**
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is **cce-bandwidth-** ******. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
bandwidth_chargemode	No	String	Bandwidth billing mode. ● **bandwidth**: billed by bandwidth ● **traffic**: billed by traffic Default: **bandwidth**
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. ● The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. ● The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. ● The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.

Parameter	Mandatory	Type	Description
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. • **PER**: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. • **5_telcom**: China Telecom • **5_union**: China Unicom • **5_bgp**: dynamic BGP • **5_sbgp**: static BGP The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet. This field can be specified only for clusters of v1.21 or later.
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block. This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.
available_zone	Yes	Array of strings	AZ where the load balancer is located. You can obtain all supported AZs by **getting the AZ list**. This parameter is available only for dedicated load balancers.
l4_flavor_name	Yes	String	Flavor name of the layer-4 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers.

Parameter	Mandatory	Type	Description
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of **l4_flavor_name**, that is, both are elastic specifications or fixed specifications.
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Do not use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`
ipv6_vip_virsubnet_id	No	String	ID of the IPv6 subnet where the load balancer resides. IPv6 must be enabled for the corresponding subnet. This parameter is mandatory only when the dual-stack clusters are used. This parameter is available only for dedicated load balancers.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

Step 5 Enter **http://*121.**.**.***:80** in the address box of the browser to access the workload (for example, **Nginx workload**).

121.**.**.** indicates the IP address of the unified load balancer.

----End

Associating an Existing Load Balancer to an Ingress While Creating the Ingress

CCE allows you to connect to an existing load balancer when creating an ingress.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

NOTE

- Starting from cluster v1.23, the ingress version is switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see **Ingress API Version Upgrade in CCE Clusters v1.23**.
- An existing dedicated load balancer must be of the application type (HTTP/HTTPS) and support private networks (with a private IP address).

If the cluster version is 1.23 or later, the YAML file configuration is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 8080            # Replace 8080 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

If the cluster version is 1.21 or earlier, the YAML file configuration is as follows:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.port: '80'
    kubernetes.io/ingress.class: cce
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
```



```
    servicePort: 80
  property:
    ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-75 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	ID of a load balancer. The value can contain 1 to 100 characters. **How to obtain**: On the management console, click **Service List**, and choose **Networking** > **Elastic Load Balance**. Click the name of the target load balancer. On the **Summary** tab page, find and copy the ID.
kubernetes.io/elb.ip	No	String	Service address of a load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.
kubernetes.io/elb.class	Yes	String	Load balancer type. • **union**: shared load balancer • **performance**: dedicated load balancer, which can be used only in clusters of v1.17 and later. For details, see **Differences Between Shared and Dedicated Load Balancers**. **NOTE** If a LoadBalancer ingress accesses an existing dedicated load balancer, the dedicated load balancer must be of the application load balancing (HTTP/HTTPS) type.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

Step 5 Enter **http://121.**.**.**:80** in the address box of the browser to access the workload (for example, **Nginx workload**).

121..**.**** indicates the IP address of the unified load balancer.

----End

7.4.3.3 Annotations for Configuring LoadBalancer Ingresses

You can add annotations to a YAML file for more advanced ingress functions. This section describes the annotations that can be used when you create a LoadBalancer ingress.

Indexes

Category	Ingress Annotation
Load balancer configuration	• **Basic Configurations for Interconnecting with ELB** • **Adding Resource Tags**
Port or protocol configuration	• **Configuring ELB Certificates** • **Using HTTP/2** • **Interconnecting with HTTPS/GRPC Backend Services** • **Configuring a Range of Listening Ports**
Advanced features of ELB listeners	• **Configuring Timeout for an Ingress** • **Configuring a Slow Start** • **Blocklist/Trustlist** • **Configuring an HTTP/HTTPS Header** • **Enabling GZIP** • **Configuring a Custom EIP**
Forwarding policy	• **Configuring Grayscale Release** • **Configuring URL Redirection** • **Configuring URL Rewriting** • **Redirecting HTTP to HTTPS** • **Configuring the Priorities of Forwarding Rules** • **Configuring a Custom Header Forwarding Policy** • **Configuring Cross-Origin Access** • **Configuring Advanced Forwarding Rules** • **Configuring Advanced Forwarding Actions**

NOTICE

In a cluster, multiple ingresses can share a listener, allowing them to use the same port on a single load balancer. If two ingresses have different listener configurations, the listener configuration of the earlier ingress (known as the first route) will be used. For details about how to check the first route, see **How Can I Determine Which Ingress the Listener Settings Have Been Applied To?**

This case involves the following scenarios:

- **Configuring ELB Certificates**
- **Using HTTP/2**
- **Configuring Timeout for an Ingress**
- **Blocklist/Trustlist**
- **Configuring an HTTP/HTTPS Header**
- **Enabling GZIP**

Ensure that the configurations of different listeners for various ingresses are synchronized.

Basic Configurations for Interconnecting with ELB

Application scenarios and use cases:

- Associate an existing load balancer. For details, see **Associating an Existing Load Balancer to an Ingress While Creating the Ingress**.
- Automatically create a load balancer. For details, see **Automatically Creating a Load Balancer While Creating an Ingress**.

Table 7-76 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. Options: • **union**: shared load balancer • **performance**: dedicated load balancer, which can be used only in clusters of v1.17 and later. For details, see **Differences Between Shared and Dedicated Load Balancers**.	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/ingress.class	String	• **cce**: A proprietary LoadBalancer ingress is used. • **nginx**: Nginx Ingress is used. This parameter is mandatory when an ingress is created by calling the API. For clusters of v1.23 or later, use the parameter **ingressClassName**. For details, see **Creating a LoadBalancer Ingress Using kubectl**.	Only clusters of v1.21 or earlier
kubernetes.io/elb.port	String	This parameter indicates the external port registered with the address of the LoadBalancer Service. The value ranges from 1 to 65535. **NOTE** Some ports are high-risk ports and are blocked by default, for example, port 21.	v1.9 or later
kubernetes.io/elb.id	String	Mandatory **when an existing load balancer is to be associated**. ID of a load balancer. **How to obtain**: On the management console, click **Service List**, and choose **Networking** > **Elastic Load Balance**. Click the name of the target load balancer. On the **Summary** tab page, find and copy the ID.	v1.9 or later
kubernetes.io/elb.ip	String	Mandatory **when an existing load balancer is to be associated**. Service address of a load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.	v1.9 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.autocreate	Table 7-97 object	Mandatory **when load balancers are automatically created**. **Example** • Automatically created shared load balancer with an EIP bound: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • Automatically created shared load balancer with no EIP bound: {"type":"inner","name":"A-location-d-test"}	v1.9 or later
kubernetes.io/elb.enterpriseID	String	Optional **when load balancers are automatically created**. **Clusters of v1.15 and later versions support this field. In clusters earlier than v1.15, load balancers are created in the default project by default.** This parameter indicates the ID of the enterprise project in which the ELB load balancer will be created. If this parameter is not specified or is set to **0**, resources will be bound to the default enterprise project. **How to obtain**: Log in to the management console and choose **Enterprise** > **Project Management** on the top menu bar. In the list displayed, click the name of the target enterprise project, and copy the ID on the enterprise project details page.	v1.15 or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.subnet-id	String	Optional **when load balancers are automatically created**. ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. • Mandatory when a cluster of v1.11.7-r0 or earlier is to be automatically created. • Optional for clusters of a version later than v1.11.7-r0.	Mandatory for clusters earlier than v1.11.7-r0 Discarded in clusters of a version later than v1.11.7-r0

Configuring ELB Certificates

For details about application scenarios and use cases, see **Configuring an HTTPS Certificate for a LoadBalancer Ingress**.

Table 7-77 ELB certificate annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tls-certificate-ids	String	ELB certificate IDs, which are separated by comma (,). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.	v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tls-ciphers-policy	String	The default value is **tls-1-2**, which is the default security policy used by the listener and takes effect only when HTTPS is used. Options: • **tls-1-0** • **tls-1-1** • **tls-1-2** • **tls-1-2-strict** • **tls-1-0-with-1-3** (dedicated load balancer) • **tls-1-2-fs** (dedicated load balancer) • **tls-1-2-fs-with-1-3** (dedicated load balancer) For details of cipher suites for each security policy, see **Table 7-100**.	Clusters of v1.17.17 or later
kubernetes.io/elb.security_policy_id	String	The ID of the custom security group policy on ELB. Obtain it on the ELB console. This field takes effect only when HTTPS is used and has a higher priority than the default security policy. For details about how to create and update a custom security policy, see **TLS Security Policy**.	Clusters of v1.17.17 or later

Adding Resource Tags

For details about application scenarios and use cases, see **Automatically Creating a Load Balancer While Creating an Ingress**.

Table 7-78 Annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.tags	String	Add resource tags to a load balancer. This parameter can be configured only when a load balancer is automatically created. A tag is in the format of "key=value". Use commas (,) to separate multiple tags.	v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later

Using HTTP/2

For details about application scenarios and use cases, see **Configuring HTTP/2 for a LoadBalancer Ingress**.

Table 7-79 Annotations for using HTTP/2

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.http2-enable	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Options: • **true**: enabled • **false**: disabled (default value) Note: **HTTP/2 can be enabled or disabled only when the listener uses HTTPS.** This parameter is invalid and defaults to **false** when the listener protocol is HTTP.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later

Interconnecting with HTTPS/GRPC Backend Services

For details about application scenarios and use cases, see **Configuring HTTPS Backend Services for a LoadBalancer Ingress** and **Configuring gRPC Backend Services for a LoadBalancer Ingress**.

Table 7-80 Annotations for interconnecting with HTTPS backend services

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.pool-protocol	String	To interconnect with HTTPS backend services, set this parameter to **https**.	v1.23.8, v1.25.3, or later
		To interconnect with GRPC backend services, set this parameter to **grpc**.	v1.23.10-r20, v1.25.5-r20, v1.27.2-r20, v1.28.1-r0, or later

Configuring Timeout for an Ingress

For details about application scenarios and use cases, see **Configuring Timeout for a LoadBalancer Ingress**.

Table 7-81 Annotations for configuring timeout of an ingress

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.keepalive_timeout	String	Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request. Value: • For TCP listeners, the value ranges from **10** to **4000** (in seconds). The default value is **300**. • For HTTP or HTTPS listeners, the value ranges from **0** to **4000** (in seconds). The default value is **60**. For UDP listeners, this parameter does not take effect.	Dedicated load balancers: v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, or later Shared load balancers: v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later
kubernetes.io/elb.client_timeout	String	Timeout for waiting for a request from a client. There are two cases: • If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. • If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected. The value ranges from **1** to **300** (in seconds). The default value is **60**. This parameter is available only for HTTP and HTTPS listeners. Minimum value: **1** Maximum value: **300** Default value: **60**	

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.member_timeout	String	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond within the duration specified by **member_timeout**, the load balancer will stop waiting and return HTTP 504 Gateway Timeout. The value ranges from **1** to **300** (in seconds). The default value is **60**. This parameter is available only for HTTP and HTTPS listeners. Minimum value: **1** Maximum value: **300** Default value: **60**	

Configuring a Slow Start

For details about application scenarios and use cases, see **Configuring a Slow Start for a LoadBalancer Ingress**.

Table 7-82 Annotations for configuring a slow start

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.slowstart	String	Duration of slow start, in seconds. The slow start duration ranges from 30 to 1200. • This configuration applies only to dedicated load balancers. • This parameter is valid only when the allocation policy of the target Service is weighted round robin (WRR) and sticky session is disabled. **NOTE** The load balancer linearly increases the proportion of requests to backend servers in slow start mode. When the configured slow start duration elapses, the load balancer sends full share of requests to backend servers and exits the slow start mode.	v1.23 or later

Configuring Grayscale Release

For details about application scenarios and use cases, see **Configuring Grayscale Release for a LoadBalancer Ingress**.

Table 7-83 Annotations for configuring grayscale release

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.canary	String	Grayscale release status for an ingress. If this parameter is set to **true**, the implementation of grayscale release varies depending on annotation configurations. Option: **true** • This configuration applies only to dedicated load balancers. • If this parameter is set to **true**, the configuration cannot be deleted or modified.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.canary-weight	String	Weight of a grayscale release. After this parameter is configured, the ingress will be grayscale released based on the weight. • The value is a positive integer ranging from 0 to 100. It is a percentage of traffic for routing. • This parameter is mandatory when an ingress is grayscale released by weight. • Do not configure this parameter with other grayscale release functions.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.session-affinity-mode	String	After weight-based grayscale release is enabled, configure sticky session. This parameter can only be set to **HTTP_COOKIE** for grayscale release.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.session-affinity-option	String	Sticky session timeout after sticky session is enabled for weight-based grayscale release. The parameter value is a JSON string in the following format: `{"persistence_timeout": "1440"}` Parameters: • Timeout range: 1 to 1440 • Default value: **1440**	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.canary-by-header	String	Key of an ingress header used for grayscale release, indicating the name of a request header. This parameter must be used with **kubernetes.io/elb.canary-by-header-value**. Parameters: Enter 1 to 40 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.canary-by-header-value	String	Value of an ingress header used for grayscale release. This parameter must be used with **kubernetes.io/elb.canary-by-header**. The parameter value is an array in JSON format, for example: `'{"values":["a","b"]}'` Parameters: ● Array length: At least one value must be configured. – If domain names and paths are configured for an ingress forwarding policy, a maximum of eight values can be configured. – If only paths are configured for an ingress forwarding policy, a maximum of nine values can be configured. ● Enter 1 to 128 characters. Asterisks (*) and question marks (?) are allowed, but spaces and double quotation marks are not allowed. An asterisk can match zero or more characters, and a question mark can match one character.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.canary-by-cookie	String	Key of cookie-based grayscale release, indicating the name of a request cookie. This parameter must be used with **kubernetes.io/elb.canary-by-cookie-value**. Parameters: Enter 1 to 100 characters, including letters, digits, and other characters (!%'"()*+,./:=?@^\\-_`~).	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.canary-by-cookie-value	String	Value of cookie-based grayscale release. This parameter must be used with **kubernetes.io/elb.canary-by-cookie**. The parameter value is an array in JSON format, for example: `'{"values":["a","b"]}'` Parameters: ● Array length: At least one value must be configured. – If domain names and paths are configured for an ingress forwarding policy, a maximum of eight values can be configured. – If only paths are configured for an ingress forwarding policy, a maximum of nine values can be configured. ● Enter 1 to 100 characters, including letters, digits, and other characters (!%'"()*+,./:=?@^\\-_`~). Spaces are not allowed.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.canary-related-ingress-uid	String	UID of the original ingress associated with the grayscale release ingress, which is used to display the association between the original ingress and the grayscale release ingress. ● Format: character string ● Value: **metadata.uid** of the original ingress	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Blocklist/Trustlist

For details about application scenarios and use cases, see **Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Ingress**.

Table 7-84 Annotations for ELB access control

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.acl-id	String	• If this parameter is not specified, CCE does not modify access control on the ELB. • If this parameter is left empty, all IP addresses are allowed to access the load balancer. • If this parameter is set to the IP address group ID of the load balancer, access control is enabled and you need to configure an IP address blocklist or trustlist for the load balancer. Additionally, you need to configure both **kubernetes.io/elb.acl-status** and **kubernetes.io/elb.acl-type**. **NOTE** For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can enter up to five IP address groups and separate them with commas (,). **How to obtain**: Log in to the console. In the **Service List**, choose **Networking** > **Elastic Load Balance**. On the Network Console, choose **Elastic Load Balance** > **IP Address Groups** and copy the **ID** of the target IP address group. For details, see **Creating an IP Address Group**.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later
kubernetes.io/elb.acl-status	String	Access control status. This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **on**: Access control is enabled. • **off**: Access control is disabled.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later
kubernetes.io/elb.acl-type	String	IP address list type. This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **black**: indicates a blocklist. The selected IP address group cannot access the load balancer. • **white**: indicates a trustlist. Only the selected IP address group can access the load balancer.	v1.23.12-r0, v1.25.7-r0, v1.27.4-r0, v1.28.2-r0, or later

Configuring a Range of Listening Ports

A custom listening port can be configured for an ingress. In this way, both ports 80 and 443 can be exposed.

For details about application scenarios and use cases, see **Configuring a Range of Listening Ports for a LoadBalancer Ingress**.

Table 7-85 Annotations for a custom listening port

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.listen-ports	String	Create multiple listening ports for an ingress. The port number ranges from 1 to 65535. The following is an example for JSON characters: `kubernetes.io/elb.listen-ports: '[{"HTTP":80},{"HTTPS":443}]'` • Only the listening ports that comply with both HTTP and HTTPS are allowed. • This function is available only for newly created ingresses in clusters of a version earlier than v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, or v1.30.4-r0. Additionally, after you configure multiple listening ports, the annotations cannot be modified or deleted. In clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later, the annotations can be modified or deleted. • If both **kubernetes.io/elb.listen-ports** and **kubernetes.io/elb.port** are configured, **kubernetes.io/elb.listen-ports** takes a higher priority. • Ingress configuration items such as the blocklist, trustlist, and timeout concurrently take effect on multiple listening ports. When HTTP/2 is enabled for an ingress, HTTP/2 takes effect only on the HTTPS port.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Configuring an HTTP/HTTPS Header

For details about application scenarios and use cases, see **Configuring an HTTP/HTTPS Header for a LoadBalancer Ingress**.

Table 7-86 Annotations for configuring an HTTP/HTTPS header

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.x-forwarded-port	String	A load balancer can obtain the port number of a listener using **X-Forwarded-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a listener port number. • **false**: Disable the function of obtaining a listener port number.	v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later
kubernetes.io/elb.x-forwarded-for-port	String	A load balancer can obtain a client port number for requests using **X-Forwarded-For-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a client port number for requests. • **false**: Disable the function of obtaining a client port number for requests.	
kubernetes.io/elb.x-forwarded-host	String	• **true**: Enable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header will be rewritten using the **Host** header of the client request and transmitted to backend servers. • **false**: Disable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header of the client will be transmitted to backend servers.	

Enabling GZIP

For details about application scenarios and use cases, see **Configuring GZIP Data Compression for a LoadBalancer Ingress**.

Table 7-87 Annotations for enabling GZIP

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.gzip-enabled	String	LoadBalancer ingresses support data compression, which reduces the size of files to be transferred, improves file transfer efficiency, and reduces the bandwidth needed for the transmission. If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. By default, data compression is disabled. The files in the following format can be compressed: ● Brotli can compress all file formats. ● GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json This function is available only for HTTP/HTTPS listeners of dedicated load balancers. If the advanced configuration for enabling data compression or the target annotation is deleted, the ELB configuration will not be modified.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Configuring URL Redirection

For details about application scenarios and use cases, see **Configuring URL Redirection for a LoadBalancer Ingress**.

Table 7-88 Annotations for configuring URL redirection

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.redirect-url	String	URL for redirection. Format: A valid URL must start with **http://** or **https://**, for example, **https://example.com/**. Parameter: This configuration takes effect on all forwarding rules of a single ingress. After the configuration is deleted, the target URL redirection rule will be automatically cleared. Either this annotation or the annotation of a grayscale release can be configured.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later
kubernetes.io/elb.redirect-url-code	String	Code returned after an ingress is redirected to a URL. Format: The return code can be 301, 302, 303, 307, or 308. Parameter: The default value is **301**.	

Configuring URL Rewriting

For details about application scenarios and use cases, see **Configuring URL Rewriting for a LoadBalancer Ingress**.

Table 7-89 Annotations for configuring URL rewriting

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.rewrite-target	String	Information about the rewritten path. Format: A proper rule matching a regular expression must start with a slash (/). Parameter: This configuration takes effect on the URL of a single ingress matching the regular expression. After the configuration is deleted, the target URL rewriting rule will be automatically cleared. Either this annotation or the annotation of a grayscale release can be configured.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Redirecting HTTP to HTTPS

For details about application scenarios and use cases, see **Redirecting HTTP to HTTPS for a LoadBalancer Ingress**.

Table 7-90 Annotations for redirecting HTTP to HTTPS

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.ssl-redirect	String	Whether to enable redirection from HTTP to HTTPS. Format: The value can be **true** or **false**. Parameter: **true** indicates that redirection is enabled. If the value is **false** or the parameter is unavailable, redirection is disabled. **NOTE** Either this annotation or the annotation of a grayscale release can be configured.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

Configuring the Priorities of Forwarding Rules

When ingresses use the same load balancer listener, forwarding rules can be prioritized based on the following rules:

- Forwarding rules of different ingresses: The rules are sorted based on the priorities (ranging from 1 to 1000) of the **kubernetes.io/elb.ingress-order** annotation. A smaller value indicates a higher priority.
- Forwarding rules of an ingress: If the **kubernetes.io/elb.rule-priority-enabled** annotation is set to **true**, the forwarding rules are sorted based on the sequence in which they are added during ingress creation. A forwarding rule added earlier indicates a higher priority. If the **kubernetes.io/elb.rule-priority-enabled** annotation is not configured, the default sorting of the forwarding rules on the load balancer will be used.

If the preceding annotations are not configured, the default sorting of the forwarding rules on the load balancer will be used, regardless of whether the forwarding rules are of the same ingress or different ingresses under the same load balancer listener.

For details about application scenarios and use cases, see **Configuring the Priorities of Forwarding Rules for LoadBalancer Ingresses**.

Table 7-91 Annotations for configuring the priorities of forwarding rules

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.ingress-order	String	Specifies the sequence of forwarding rules of different ingresses. The value ranges from 1 to 1000. A smaller value indicates a higher priority. The priority of a forwarding rule must be unique under the same load balancer listener. This parameter is available only for dedicated load balancers. **NOTE** When this annotation is configured, the **kubernetes.io/elb.rule-priority-enabled** annotation is enabled by default. The forwarding rules of each ingress will be sorted.	v1.23.15-r0, v1.25.10-r0, v1.27.7-r0, v1.28.5-r0, v1.29.1-r10, or later
kubernetes.io/elb.rule-priority-enabled	String	This parameter can only be set to **true**, indicating to sort the forwarding rules of an ingress. The priorities of the forwarding rules are determined based on the sequence in which they are added during ingress creation. A forwarding rule added earlier indicates a higher priority. If this parameter is not configured, the default sorting of the forwarding rules on the load balancer will be used. After this parameter is enabled, it cannot be disabled. This parameter is available only for dedicated load balancers.	

Configuring a Custom Header Forwarding Policy

For details about application scenarios and use cases, see **Configuring a Custom Header Forwarding Policy for a LoadBalancer Ingress**.

Table 7-92 Annotations for configuring a custom header forwarding policy

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.headers.*${svc_name}*	String	Custom header of the Service associated with an ingress. *${svc_name}* is the Service name. Format: a JSON string, for example, {"key": "test", "values": ["value1", "value2"]} • **key/value** indicates the key-value pair of the custom header. A maximum of eight values can be configured. Enter 1 to 40 characters for a key. Only letters, digits, hyphens (-), and underscores (_) are allowed. Enter 1 to 128 characters for a value. Asterisks (*) and question marks (?) are allowed, but spaces and double quotation marks are not allowed. An asterisk can match zero or more characters, and a question mark can match one character. • Either a custom header or grayscale release can be configured. • Enter 1 to 51 characters for *${svc_name}*.	v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later

Configuring a Custom EIP

For details about application scenarios and use cases, see **Configuring a Custom EIP for a LoadBalancer Ingress**.

Table 7-93 Annotations of custom EIP configurations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.custom-eip-id	String	ID of the custom EIP, which can be seen on the EIP console The EIP must be bindable.	v1.23.18-r0, v1.25.13-r0, v1.27.10-r0, v1.28.8-r0, v1.29.4-r0, v1.30.1-r0, or later

Configuring Cross-Origin Access

For details about application scenarios and use cases, see **Configuring Cross-Origin Access for LoadBalancer Ingresses**.

Table 7-94 Annotations for configuring cross-origin access

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.cors-allow-origin	Array[string]	Specify **Access-Control-Allow-Origin** for the origin that can be accessed. Options: • Wildcard (*): indicates that all domain names can be accessed. • Domain names: start with **http://** or **https://**. Level-1 wildcard domain names are supported. The format is **http(s)://example.com** or **http(s)://example.com:port**, where the port number ranges from 1 to 65535. You can enter multiple values by separating them with commas (,).	v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later
kubernetes.io/elb.cors-allow-headers	Array[string]	Specify **Access-Control-Allow-Headers** for allowed request headers. You can enter multiple values by separating them with commas (,).	
kubernetes.io/elb.cors-expose-headers	Array[string]	Specify **Access-Control-Expose-Headers** for custom response headers that can be accessed by a cross-origin request. This allows you to retrieve non-standard response headers using client-side JavaScript code. You can enter multiple values by separating them with commas (,).	
kubernetes.io/elb.cors-allow-methods	Array[string]	Specify **Access-Control-Allow-Methods** for allowed HTTP request methods. You can enter multiple values by separating them with commas (,).	
kubernetes.io/elb.cors-allow-credentials	String	Specify **Access-Control-Allow-Credentials** to control the sending of credentials (such as cookies). Options: • **true**: Credentials can be sent. • **false**: Credentials cannot be sent. Once configured, the settings cannot be deleted. To remove the configuration, use **kubernetes.io/elb.cors-disabled** to delete all cross-origin settings.	

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.cors-max-age	String	Specify **Access-Control-Max-Age** for the cache duration of a CORS pre-check request. Unit: second. Value range: -1 to 172800 Properly configure this parameter based on service requirements. If the value is too low, pre-check requests may happen frequently. If the value is too high, the CORS policy may not take effect immediately.	
kubernetes.io/elb.cors-disabled	String	Specify whether to disable all cross-origin settings. Options: • **true**: disables all cross-origin settings. The parameter values in the YAML file will not be deleted, but all cross-origin settings will not take effect. • **false**: default value, indicating that cross-origin settings will be implemented based on your configuration.	

Configuring Advanced Forwarding Rules

For details about application scenarios and use cases, see **Configuring Advanced Forwarding Rules for a LoadBalancer Ingress**.

Table 7-95 Annotations for advanced forwarding rules

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.conditions.*${svc_name}*	String	Configure an advanced forwarding rule. *${svc_name}* indicates the Service name, which can contain a maximum of 51 characters. If the annotation value is set to *[]*, the advanced forwarding rule will be deleted. The annotation value is a JSON array. For details, see **Table 7-135**. **NOTICE** • Due to ELB API restrictions, a **kubernetes.io/elb.conditions.***{svcName}* can contain a maximum of 10 key-value pairs. • The rules in a condition array are connected by an AND relationship, while the values in the same rule block are connected by an OR relationship. For example, if both Method and QueryString are configured, the target traffic can be distributed only when both rules are met. However, if the Method value is GET or POST, the target traffic can be distributed only when both rules are met and the Method value must be GET or POST.	v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later

Configuring Advanced Forwarding Actions

For details about application scenarios and use cases, see **Configuring Advanced Forwarding Actions for a LoadBalancer Ingress**.

Table 7-96 Annotations for configuring advanced forwarding actions

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.actions.*${svc_name}*	String	Advanced forwarding action of the Service associated with an ingress. *${svc_name}* indicates the Service name, which can contain a maximum of 51 characters. The annotation value is a JSON array. If it is set to *[]*, the advanced forwarding action will be deleted. The following advanced forwarding actions are supported: • **Write or delete a header** • **Return a fixed response** • **Limit rate**	The forwarding actions supported by clusters vary depending on the cluster version. For details, see **Table 7-136**.

Parameters for Automatically Creating a Load Balancer

Table 7-97 elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: **cce-lb+service.UID**
type	No	String	Network type of the load balancer. • **public**: public network load balancer • **inner**: private network load balancer Default: **inner**
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is **cce-bandwidth-** ******. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.

Parameter	Mandatory	Type	Description
bandwidth_chargemode	No	String	Bandwidth billing mode. • **bandwidth**: billed by bandwidth • **traffic**: billed by traffic Default: **bandwidth**
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The value ranges from 1 Mbit/s to 2000 Mbit/s by default. Configure this parameter based on the bandwidth range allowed in your region. The minimum increment for bandwidth adjustment varies depending on the bandwidth range. • The minimum increment is 1 Mbit/s if the allowed bandwidth does not exceed 300 Mbit/s. • The minimum increment is 50 Mbit/s if the allowed bandwidth ranges from 300 Mbit/s to 1000 Mbit/s. • The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1000 Mbit/s.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. • **PER**: dedicated bandwidth
eip_type	Yes for public network load balancers	String	EIP type. • **5_telcom**: China Telecom • **5_union**: China Unicom • **5_bgp**: dynamic BGP • **5_sbgp**: static BGP The specific type varies with regions. For details, see the EIP console.
vip_subnet_cidr_id	No	String	Subnet where a load balancer is located. The subnet must belong to the VPC where the cluster resides. If this parameter is not specified, the ELB load balancer and the cluster are in the same subnet. This field can be specified only for clusters of v1.21 or later.

Parameter	Mandatory	Type	Description
vip_address	No	String	Private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the ELB CIDR block. If this parameter is not specified, an IP address will be automatically assigned from the ELB CIDR block. This parameter is available only in clusters of v1.23.11-r0, v1.25.6-r0, v1.27.3-r0, or later versions.
available_zone	Yes	Array of strings	AZ where the load balancer is located. You can obtain all supported AZs by **getting the AZ list**. This parameter is available only for dedicated load balancers.
l4_flavor_name	Yes	String	Flavor name of the layer-4 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers.
l7_flavor_name	No	String	Flavor name of the layer-7 load balancer. You can obtain all supported types by **getting the flavor list**. This parameter is available only for dedicated load balancers. The value of this parameter must be the same as that of **l4_flavor_name**, that is, both are elastic specifications or fixed specifications.
elb_virsubnet_ids	No	Array of strings	Subnet where the backend server of the load balancer is located. If this parameter is left blank, the default cluster subnet is used. Load balancers occupy different number of subnet IP addresses based on their specifications. Do not use the subnet CIDR blocks of other resources (such as clusters and nodes) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

Parameter	Mandatory	Type	Description
ipv6_vip_virsubnet_id	No	String	ID of the IPv6 subnet where the load balancer resides. IPv6 must be enabled for the corresponding subnet. This parameter is mandatory only when the dual-stack clusters are used. This parameter is available only for dedicated load balancers.

7.4.3.4 Advanced Setting Examples of LoadBalancer Ingresses

7.4.3.4.1 Configuring an HTTPS Certificate for a LoadBalancer Ingress

Ingresses support SSL or TLS certificates, allowing you to secure your Services with HTTPS.

You are allowed to use either of the following ways to configure an ingress certificate in a cluster:

- **Using a TLS Certificate**: You need to first import a certificate to a secret. CCE will then automatically handle the certificate configurations on the ELB console and give a name to the certificate (started with **k8s_plb_default**). This certificate, which is generated by CCE, **cannot be modified or deleted from the ELB console**. To modify the certificate, update the secret where the certificate is imported to on CCE.
- **Using a Certificate Created on the ELB Console**: You are allowed to directly use certificates created on the ELB console. There is no need to manually configure the cluster Secrets, and you can modify the certificates on the ELB console.

NOTE

If multiple ingresses share the same external port on a load balancer, you are advised to use the same certificate and security policy for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring a Same Port for Multiple Ingresses**.

Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider. For details, see **Purchasing an SSL Certificate**.

Using a TLS Certificate

You can configure a TLS certificate using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

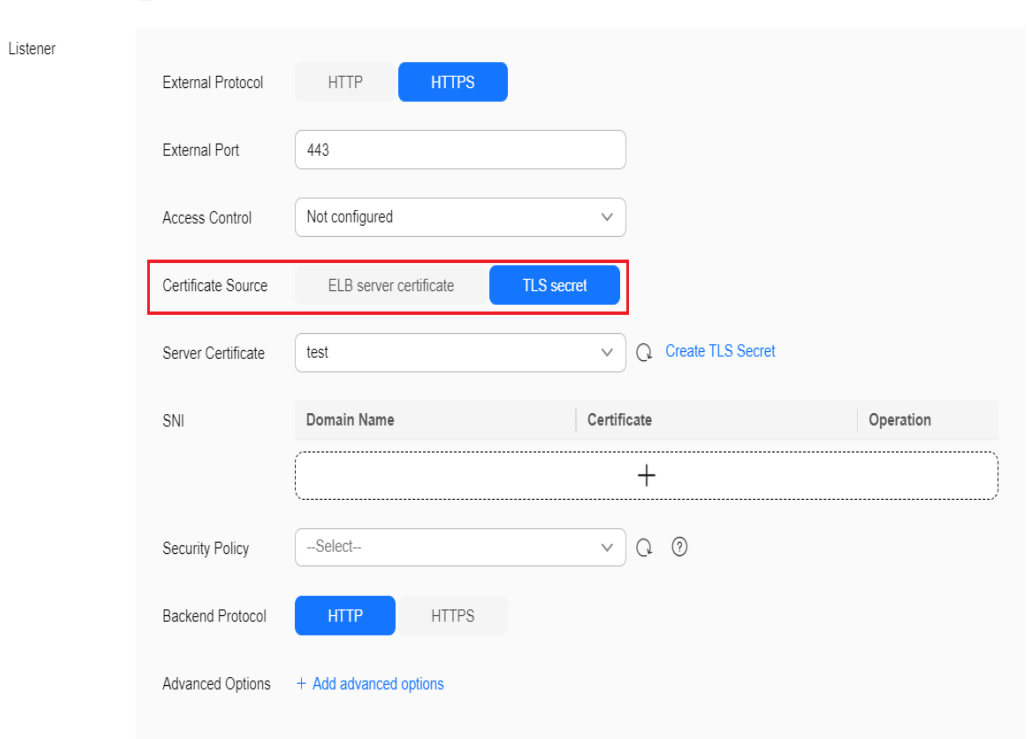
This example explains only key parameters for configuring HTTPS certificates. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-98 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: Select **HTTPS** when configuring a certificate for an ingress. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **TLS secret**. • **Server Certificate**: **kubernetes.io/tls** and **IngressTLS** are supported. If no certificate is available, you can create a TLS certificate. For details about the configuration parameters, see **Creating a Secret**.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: TLS secret • Server Certificate: test

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-62 Selecting a TLS certificate



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Ingress supports two TLS secret types: kubernetes.io/tls and IngressTLS. IngressTLS is used as an example. For details, see **Creating a Secret**. For details about examples of the kubernetes.io/tls secret and its description, see **TLS secrets**.

Create a YAML file named **ingress-test-secret.yaml**. The file name can be customized.

```
vi ingress-test-secret.yaml
```

The YAML file is configured as follows:

```
apiVersion: v1
data:
  tls.crt: LS0******tLS0tCg==
  tls.key: LS0tL******0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
  name: ingress-test-secret
  namespace: default
type: IngressTLS
```

NOTE

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files. The values of **tls.crt** and **tls.key** are Base64-encoded.

Step 3 Create a secret.

```
kubectl create -f ingress-test-secret.yaml
```

If information similar to the following is displayed, the secret has been created:

```
secret/ingress-test-secret created
```

Step 4 Check the created secret.

```
kubectl get secrets
```

If information similar to the following is displayed, the secret has been created:

```
NAME                  TYPE                                  DATA   AGE
ingress-test-secret   IngressTLS                            2      13s
```

Step 5 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

NOTE

Default security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.17 or later.

Custom security policy (kubernetes.io/elb.security_policy_id) is supported only in clusters of v1.17.17 or later.

An example YAML file of an ingress associated with an automatically created load balancer is as follows:

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.security_policy_id: 99bec42b-0dd4-4583-98e9-b05ce628d157 # The priority of the custom security policy is higher than that of the default security policy.
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.security_policy_id: 99bec42b-0dd4-4583-98e9-b05ce628d157 # The priority of the custom security policy is higher than that of the default security policy.
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
```



```
- host: ''
  http:
    paths:
    - path: '/'
      backend:
        service:
          name: <your_service_name>  # Replace it with the name of your target Service.
          port:
            number: 80               # Replace 80 with the port number of your target Service.
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      pathType: ImplementationSpecific
ingressClassName: cce
```

Table 7-99 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.security_policy_id	No	String	The ID of the custom security group policy on ELB. Obtain it on the ELB console. This field takes effect only when HTTPS is used and has a higher priority than the default security policy. For details about how to create and update a custom security policy, see **TLS Security Policy**.
kubernetes.io/elb.tls-ciphers-policy	No	String	The default value is **tls-1-2**, which is the default security policy used by the listener and takes effect only when HTTPS is used. Options: • **tls-1-0** • **tls-1-1** • **tls-1-2** • tls-1-2-strict • tls-1-0-with-1-3 (dedicated load balancer) • tls-1-2-fs (dedicated load balancer) • tls-1-2-fs-with-1-3 (dedicated load balancer) For details of cipher suites for each security policy, see **Table 7-100**.

Parameter	Mandatory	Type	Description
tls	No	Array of strings	When HTTPS is used, this parameter must be added to specify the secret certificate. Multiple independent domain names and certificates can be added. For details, see **Configuring SNI for a LoadBalancer Ingress**.
secretName	No	String	This parameter is mandatory if HTTPS is used. Set this parameter to the name of the created secret.

Table 7-100 tls_ciphers_policy parameters

Security Policy	TLS Version	Cipher Suite
tls-1-0	TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA
tls-1-1	TLS 1.2 TLS 1.1	
tls-1-2	TLS 1.2	
tls-1-2-strict	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384

Security Policy	TLS Version	Cipher Suite
tls-1-0-with-1-3 (dedicated load balancer)	TLS 1.3 TLS 1.2 TLS 1.1 TLS 1.0	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:AES128-SHA256:AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-SHA:AES256-SHA:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_8_SHA256:TLS_AES_128_CCM_SHA256
tls-1-2-fs (dedicated load balancer)	TLS 1.3 TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384
tls-1-2-fs-with-1-3 (dedicated load balancer)	TLS 1.3 TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_8_SHA256:TLS_AES_128_CCM_SHA256

Step 6 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 7 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce     *        121.**.**.**    80,443  10s
```

Step 8 Enter **https://*121.**.**.***:443** in the address box of the browser to access the workload (for example, **Nginx workload**).

*121.**.**.** indicates the IP address of the unified load balancer.

----End

Using a Certificate Created on the ELB Console

You can configure a certificate created on the ELB console for an ingress through either the CCE console or kubectl.

NOTE

- If both an ELB certificate and a TLS certificate are specified for the same ingress, the ingress will use the ELB certificate.
- CCE does not check whether an ELB certificate is valid. It only checks whether the certificate is present.
- Only ingresses in clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support ELB certificates.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

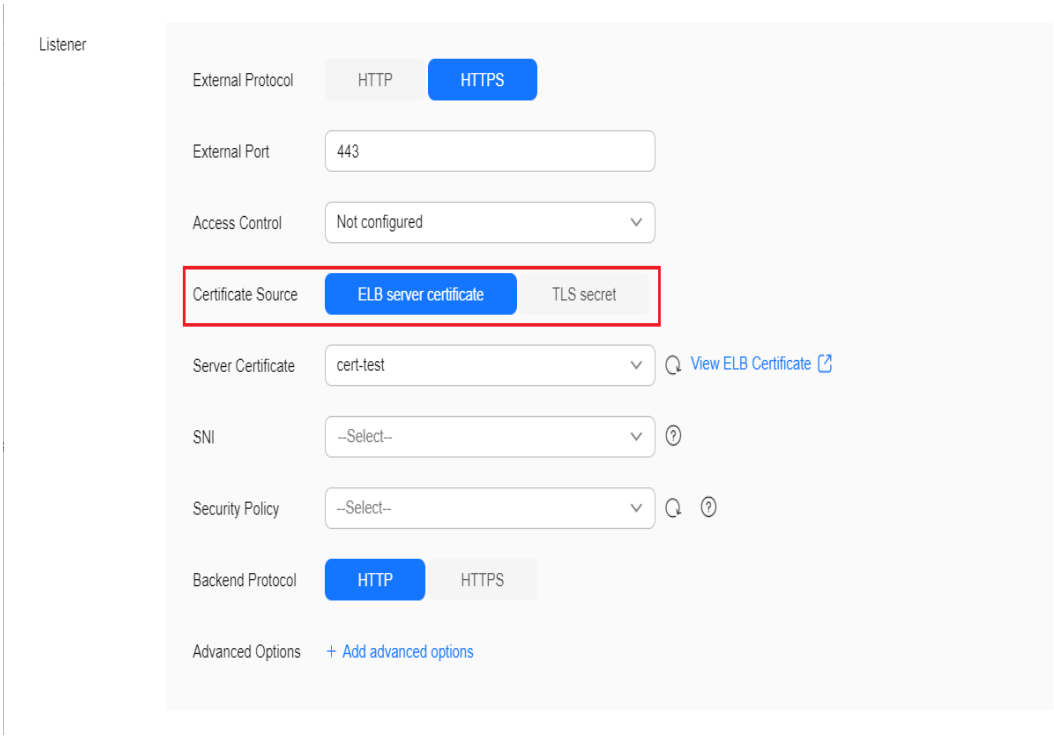
This example explains only key parameters for configuring HTTPS certificates. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-101 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: Select **HTTPS**. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **ELB server certificate**. • **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: ELB server certificate • Server Certificate: cert-test

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-63 Selecting a certificate from the ELB service



Step 4 Click **OK**.

----End

Using kubectl

To use an ELB certificate, you can specify the **kubernetes.io/elb.tls-certificate-ids** annotation.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: 80               # Replace 80 with the port number of your target Service.
```



```
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
ingressClassName: cce
```

Table 7-102 Key parameters

Parameter	Type	Description
kubernetes.io/elb.tls-certificate-ids	String	ELB certificate IDs, which are separated by comma (,). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained. If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80,443  10s
```

----End

Configuring a Same Port for Multiple Ingresses

In a cluster, multiple ingresses can share a listener, allowing them to use the same port on a single load balancer. When two ingresses are set up with HTTPS certificates, the server certificate that is used will be based on the configuration of the earliest ingress. For details, see **How Can I Determine Which Ingress the Listener Settings Have Been Applied To?**

Starting from v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, v1.29.1-r0, and later, the YAML file of ingresses includes **annotation: kubernetes.io/elb.listener-master-ingress**. This annotation specifies the server certificate configured and used by ingresses.

For example, the configurations of two ingresses are as follows:

Ingress Name	ingress1	ingress2
Namespace	default	default
Creation Time	2024-04-01	2024-04-02
Protocol	HTTPS	HTTPS
Load Balancer	elb1	elb1
Port	443	443
kubernetes.io/elb.listener-master-ingress	default/ingress1	default/ingress1

In the configurations of the two ingresses, **annotation: kubernetes.io/elb.listener-master-ingress** is present and the value is **default/ingress1**, indicating that the server certificates that take effect for the two ingresses are the server certificates configured for **ingress1** in the **default** namespace.

NOTE

If ingresses in separate namespaces use the same listener and TLS certificates, the secrets associated with the TLS certificates may not display normal for the later-created ingress due to namespace isolation. For details, see **How Can I Synchronize Certificates When Multiple Ingresses in Different Namespaces Share a Listener?**

To update a server certificate, modify the certificate in the **ingress1** configuration within the **default** namespace. Once the modification is made, the certificates used by both **ingress1** and **ingress2** will be updated accordingly.

To find the ingress with the active certificate configuration, run the following command:

```
kubectl get ingress -n ${namespace} ${ingress_name}  -oyaml | grep kubernetes.io/elb.listener-master-ingress
```

7.4.3.4.2 Updating the HTTPS Certificate for a LoadBalancer Ingress

If the HTTPS certificate for a LoadBalancer ingress is about to expire or has expired, follow the operations provided in this section to update it.

Application Scenarios

Scenario	Description
LoadBalancer Service certificate	To update an HTTPS certificate, create a LoadBalancer Service certificate and select it when modifying the ingress. Alternatively, modify the certificate on the ELB certificate management page.

Scenario	Description
TLS certificate	To update an HTTPS certificate, update the target secret in the cluster. CCE will then automatically configure the certificate (starting with **k8s_plb_default**) on the ELB. The certificate automatically created by CCE **cannot be modified or deleted on the ELB**.

Procedure for TLS Secret Certificates

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose ConfigMaps and Secrets. In the right pane, click the **Secrets** tab, find the secret used by the ingress, and click **Update**.

Step 3 Modify the certificate file in the secret and click **OK** to update the secret. CCE will automatically synchronize the certificate to the ELB.

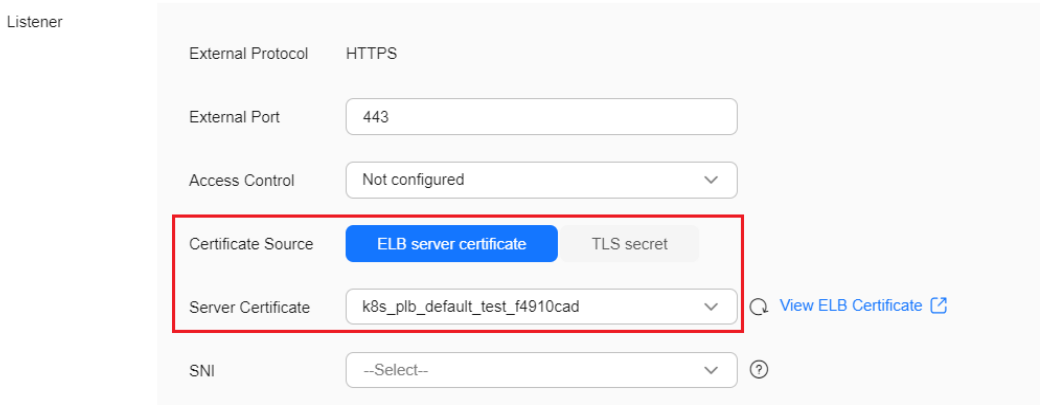
----End

Procedure for LoadBalancer Service Certificates

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. Then, click the **Ingresses** tab, locate the row containing the target ingress, and choose **More** > **Update** in the **Operation** column.

Step 3 Set **Certificate Source** to **ELB server certificate**, select a new server certificate, and click **OK** to update the ingress.



----End

7.4.3.4.3 Configuring SNI for a LoadBalancer Ingress

An SNI certificate is an extended server certificate that allows the same IP address and port number to provide multiple access domain names for external systems. Different security certificates can be used based on the domain names requested by clients to ensure HTTPS communication security.

When configuring SNI, you need to add a certificate associated with a domain name. The client submits the requested domain name information when initiating an SSL handshake request. After receiving the SSL request, the load balancer searches for the certificate based on the domain name. If the certificate is found, the load balancer will return it to the client. If the certificate is not found, the load balancer will return the default server certificate.

You are allowed to use either of the following ways to configure an ingress certificate in a cluster:

- **Configuring SNI Using a TLS Certificate**: You need to first import a certificate to a Secret. CCE will then automatically handle the certificate configurations on the ELB console and give a name to the certificate (started with **k8s_plb_default**). This certificate, which is generated by CCE, cannot be modified or deleted from the ELB console.
- **Configuring SNI Using an ELB Certificate**: You are allowed to directly use certificates created on the ELB console. There is no need to manually configure the cluster Secrets, and you can modify the certificates on the ELB console.

Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider. For details, see **Purchasing an SSL Certificate**.

Configuring SNI Using a TLS Certificate

You can configure SNI through a TLS certificate using either the CCE console or kubectl.

NOTE

- The **SNI** option is available only when HTTPS is used.
- Only one domain name can be specified for each SNI certificate. Wildcard-domain certificates are supported.
- Security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.11 or later.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

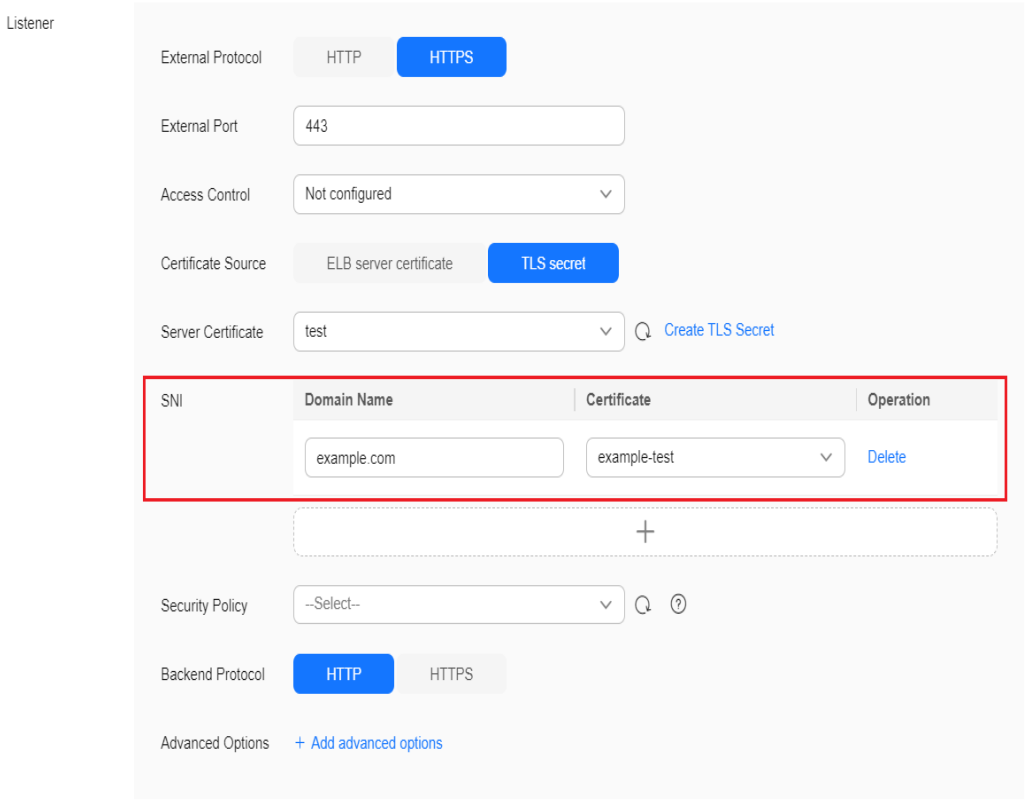
This example explains only key parameters for configuring SNI certificates. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-103 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: Select **HTTPS** when configuring a certificate for an ingress. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **TLS secret**. • **Server Certificate**: **kubernetes.io/tls** and **IngressTLS** are supported. If no certificate is available, you can create a TLS certificate. For details about the configuration parameters, see **Creating a Secret**. • **SNI**: Enter a domain name and select a certificate. The SNI certificate must contain the domain name information. SNI certificates support two TLS secret types: kubernetes.io/tls and IngressTLS.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: TLS secret • Server Certificate: test • SNI: – Domain Name: example.com – Certificate: example-test

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: example.com • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-64 Configuring SNI using a TLS certificate



Step 4 Click **OK**.

----End

Using kubectl

In this example, the **sni-test-secret** SNI certificate is used as an example. The specified domain name must be the same as that of the SNI certificate.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an automatically created load balancer is as follows:

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
    secretName: sni-test-secret   #SNI certificate
  rules:
  - host: example.com   #The domain name must be the same as the value of hosts in the tls field.
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.class: performance
```



```
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.autocreate:
      '{
          "type": "public",
          "bandwidth_name": "cce-bandwidth-******",
          "bandwidth_chargemode": "bandwidth",
          "bandwidth_size": 5,
          "bandwidth_sharetype": "PER",
          "eip_type": "5_bgp",
          "available_zone": [
              "ap-southeast-1a"
          ],
          "elb_virsubnet_ids":["b4bf8152-6c36-4c3b-9f74-2229f8e640c9"],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
  - secretName: ingress-test-secret
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
    secretName: sni-test-secret # SNI certificate
  rules:
  - host: example.com   # The domain name must be the same as the value of hosts in the tls field.
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80            # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS  HOSTS         ADDRESS         PORTS   AGE
ingress-test  cce    example.com   121.**.**.**    80,443  10s
```

Step 5 Use HTTPS to access the ingress. *${ELB_IP}* specifies the IP address accessed by the target ingress.

```
curl -H "Host:example.com" -k https://${ELB_IP}:443
```

If the ingress can be accessed, the certificate is configured.

----End

Configuring SNI Using an ELB Certificate

You can configure SNI using an ELB certificate for an ingress through either the CCE console or kubectl.

NOTE

- If both an ELB certificate and a TLS certificate are specified for the same ingress, the ingress will use the ELB certificate.
- CCE does not check whether an ELB certificate is valid. It only checks whether the certificate is present.
- Only ingresses in clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support ELB certificates.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

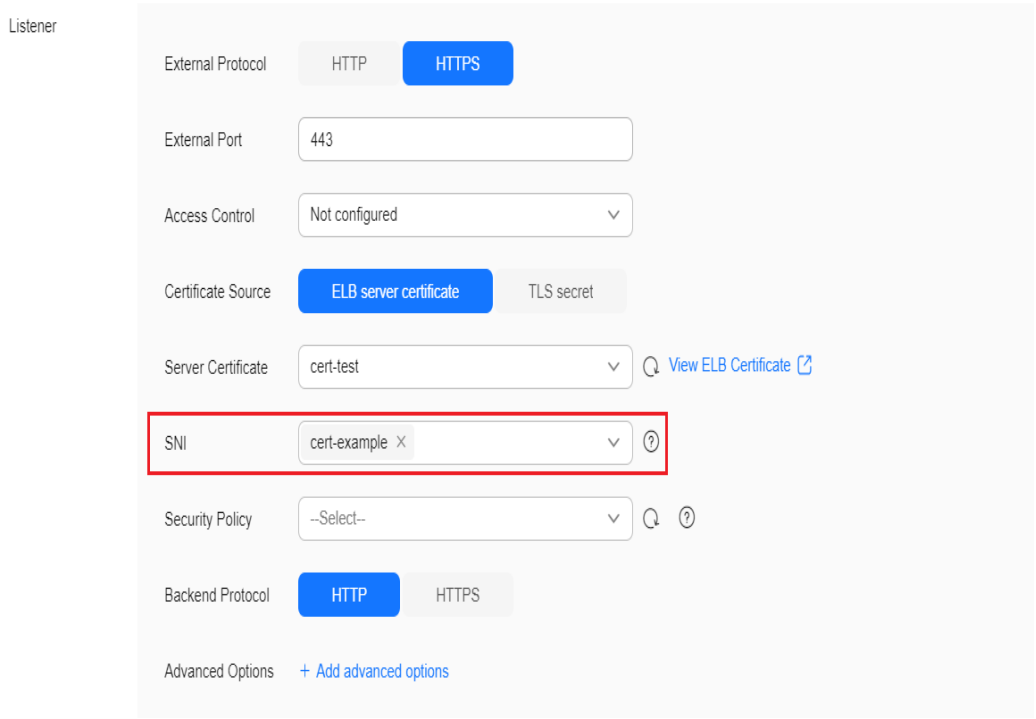
This example explains only key parameters for configuring SNI certificates. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-104 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: Select **HTTPS**. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **ELB server certificate**. • **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**. • **SNI**: Select the corresponding SNI certificate, which must contain the domain name information. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: ELB server certificate • Server Certificate: cert-test • SNI: cert-example

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-65 Configuring SNI using an ELB certificate



Step 4 Click **OK**.

----End

Using kubectl

To use an ELB certificate for an ingress, you can specify the **kubernetes.io/elb.tls-certificate-ids** annotation.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: 0b9a6c4d-bd8b-45cc-bfc8-ff0f9da54e95
    kubernetes.io/elb.class: union
    kubernetes.io/elb.tls-certificate-ids:
058cc023690d48a3867ad69dbe9cd6e5,b98382b1f01c473286653afd1ed9ab63
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
              name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: 80            # Replace 80 with the port number of your target Service.
```



```
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      pathType: ImplementationSpecific
ingressClassName: cce
```

Table 7-105 Key parameters

Parameter	Type	Description
kubernetes.io/elb.tls-certificate-ids	String	ELB certificate IDs, which are separated by comma (,). The list length is greater than or equal to 1. The first ID in the list is the server certificate, and the other IDs are SNI certificates in which a domain name must be contained. If an SNI certificate cannot be found based on the domain name requested by the client, the server certificate will be returned by default. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS          PORTS   AGE
ingress-test  cce      *         121.**.**.**     80,443  10s
```

Step 5 Use HTTPS to access the ingress. *${ELB_IP}* specifies the IP address accessed by the target ingress.

```
curl -H "Host:example.com" -k https://${ELB_IP}:443
```

If the ingress can be accessed, the certificate is configured.

----End

7.4.3.4.4 Configuring Multiple Forwarding Policies for a LoadBalancer Ingress

An ingress can route requests to multiple backend Services based on different matching policies. For example, requests can be routed to three different backend Services separately by accessing **www.example.com/foo**, **www.example.com/bar**, and **foo.example.com/**.

NOTICE

The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.

For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.

Prerequisites

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Routing an Ingress to Multiple Services

You can an ingress route to multiple services using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

This example explains only key parameters for configuring forwarding policies. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

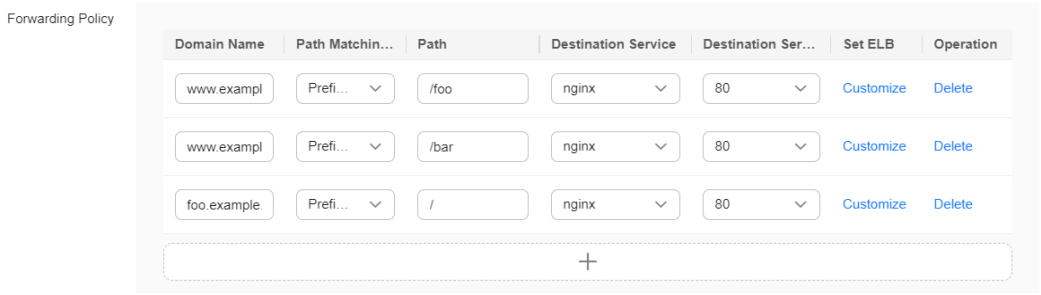
Table 7-106 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener.	• External Protocol: HTTP • External Port: 80

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	Forwarding rule 1: • Domain Name: www.example.com • Path Matching Rule: Prefix match • Path: /foo • Destination Service: nginx • Destination Service Port: 80 Forwarding rule 2: • Domain Name: www.example.com • Path Matching Rule: Prefix match • Path: /bar • Destination Service: nginx • Destination Service Port: 80 Forwarding rule 3: • Domain Name: foo.example.com • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx

Parameter	Description	Example
		• Destination Service Port: 80

Figure 7-66 Routing an ingress to multiple Services



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  #Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80            # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
      - path: '/bar'
        backend:
```



```
        service:
          name: <your_service_name>  # Replace it with the name of your target Service.
          port:
            number: 80            # Replace 80 with the port number of your target Service.
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      pathType: ImplementationSpecific
  - host: 'foo.example.com'
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80            # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: cce
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id>  #Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance  # Load balancer type
spec:
  rules:
  - host: 'www.example.com'
    http:
      paths:
      - path: '/foo'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
      - path: '/bar'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
  - host: 'foo.example.com'
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:


```
NAME          CLASS  HOSTS                          ADDRESS         PORTS   AGE
ingress-test  cce    www.example.com,foo.example.com  121.**.**.**    80      10s
```

----End

7.4.3.4.5 Configuring HTTP/2 for a LoadBalancer Ingress

Ingresses can use HTTP/2 to expose Services. Connections from the load balancer to your application use HTTP/1.x by default. If your application is capable of receiving HTTP/2 requests, enable the use of HTTP/2 by referring to **Using the CCE Console** or **Using kubectl**.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider. For details, see **Purchasing an SSL Certificate**.

Notes and Constraints

- Only an HTTPS-compliant load balancer supports HTTP/2.
- After HTTP/2 is configured, if you delete the advanced configuration for enabling HTTP/2 on the CCE console or delete the target annotation from the YAML file, HTTP/2 will be disabled on the ELB.
- If multiple ingresses share the same external port on a load balancer, you are advised to use the same HTTP/2 configuration for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring Multiple Ingresses to Use the Same External ELB Port**.

Configuring HTTP/2 for an Ingress

You can configure HTTP/2 for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

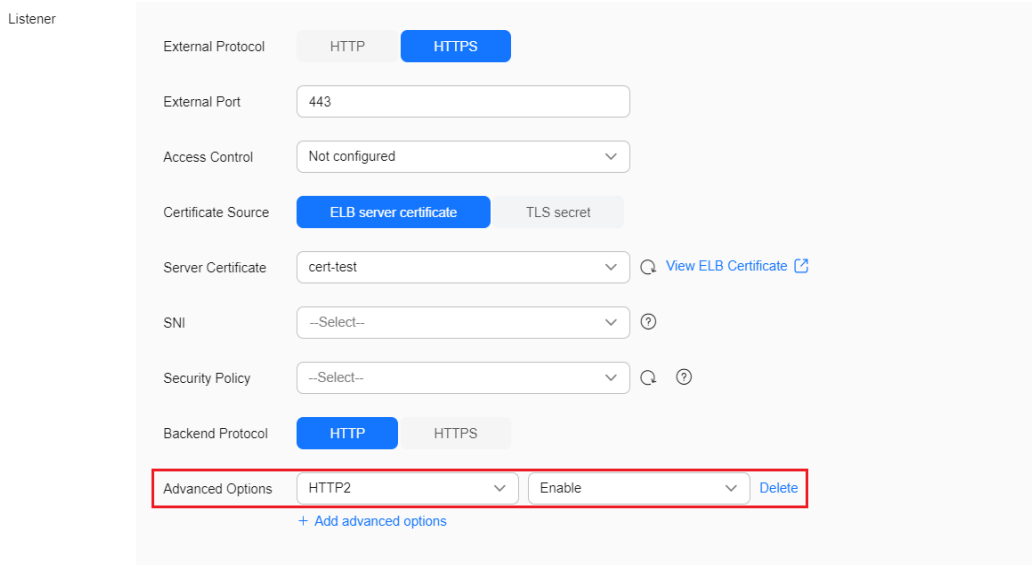
This example explains only key parameters for configuring HTTP/2. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-107 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer.	Shared
Listener	• **External Protocol**: Select **HTTPS**. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **ELB server certificate**. • **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**. • **Advanced Options**: Add advanced configurations, select **HTTP2**, and set its status to **Enable**.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: ELB server certificate • Server Certificate: cert-test • Advanced Options: HTTP2 enabled

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-67 Configuring HTTP/2 for an ingress



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your existing load balancer.
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/2.
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80            # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-108 HTTP/2 parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.http2-enable	No	String	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Options: • **true**: enabled • **false**: disabled (default value) Note: **HTTP/2 can be enabled or disabled only when the listener uses HTTPS.** This parameter is invalid and defaults to **false** when the listener protocol is HTTP.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```


If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce     *         121.**.**.**    80,443  10s
```

----End

7.4.3.4.6 Configuring HTTPS Backend Services for a LoadBalancer Ingress

Ingresses can interconnect with backend services of different protocols. By default, the backend proxy channel of an ingress is HTTP-compliant. To create an HTTPS channel, add the following configuration to the **annotations** field:

```
kubernetes.io/elb.pool-protocol: https
```

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.8-r0 or later
  - v1.25: v1.25.3-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider. For details, see **Purchasing an SSL Certificate**.

Notes and Constraints

- Ingresses can interconnect with HTTPS backend services only when dedicated load balancers are used.
- When an ingress interconnects with an HTTPS backend service, the ingress protocol must be HTTPS.

Configuring an HTTPS Backend Service

You can configure an HTTPS backend Service for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

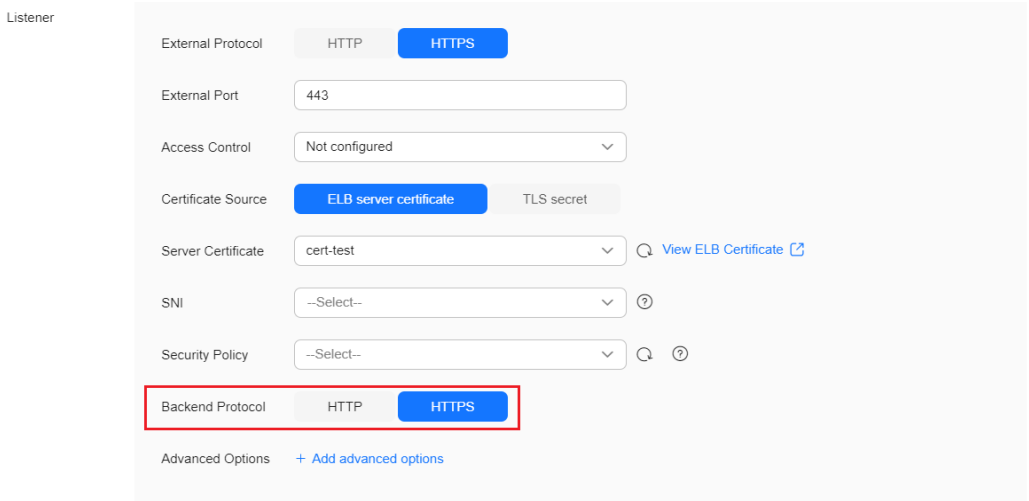
This example explains only key parameters for configuring HTTPS backend Services. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-109 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: Select **HTTPS** for an HTTPS backend Service for an ingress. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **ELB server certificate**. • **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**. • **Backend Protocol**: Select **HTTPS**.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: ELB server certificate • Server Certificate: cert-test • Backend Protocol: HTTPS

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-68 Configuring an HTTPS backend Service



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing dedicated load balancer is used.
Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.pool-protocol: https  # Interconnected HTTPS backend service
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
    - secretName: ingress-test-secret
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: 80
              property:
                ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
              pathType: ImplementationSpecific
  ingressClassName: cce
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS          PORTS   AGE
ingress-test  cce      *         121.**.**.**     80,443  10s
```

----End

7.4.3.4.7 Configuring gRPC Backend Services for a LoadBalancer Ingress

Ingresses can interconnect with backend services of different protocols. By default, the backend proxy channel of an ingress is HTTP-compliant. To create a gRPC channel, add the following configuration to the **annotations** field:

```
kubernetes.io/elb.pool-protocol: grpc
```

NOTE

This function depends on ELB listeners and is available only in certain regions. Obtain these regions on the CCE console. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.10-r20 or later
  - v1.25: v1.25.5-r20 or later
  - v1.27: v1.27.2-r20 or later
  - v1.28: v1.28.1-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.
- You can obtain a trusted certificate from a certificate provider. For details, see **Purchasing an SSL Certificate**.

Notes and Constraints

- Ingresses can interconnect with gRPC backend services only when dedicated load balancers are used.
- When an ingress interconnects with a gRPC backend service, the ingress protocol must be HTTPS and HTTP/2 must be enabled.

Configuring a gRPC Backend Service

You can configure a gRPC backend Service for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

This example explains only key parameters for configuring gRPC backend Services. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-110 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test

Parameter	Description	Example
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: Select **HTTPS** for a gRPC backend Service for an ingress. • **External Port**: specifies the port of the load balancer listener. The default HTTPS port is 443. • **Certificate Source**: Select **ELB server certificate**. • **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**. • **Backend Protocol**: Select **GRPC**. **NOTE** Only dedicated load balancers support gRPC, and HTTP/2 must be enabled. After HTTP/2 is enabled, CCE will automatically add the **kubernetes.io/elb.http2-enable:true** annotation. gRPC is available only in certain regions. Obtain these regions on the CCE console.	• External Protocol: HTTPS • External Port: 443 • Certificate Source: ELB server certificate • Server Certificate: cert-test • Backend Protocol: GRPC

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service. • (Optional) **Set ELB**: Set the health check protocol to gRPC. Click **Customize**, enable health check, and select **GRPC**.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-69 Configuring a gRPC backend Service

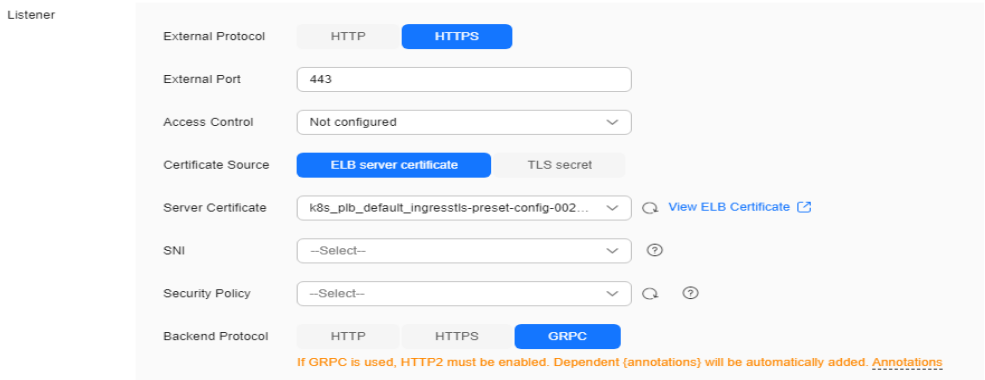
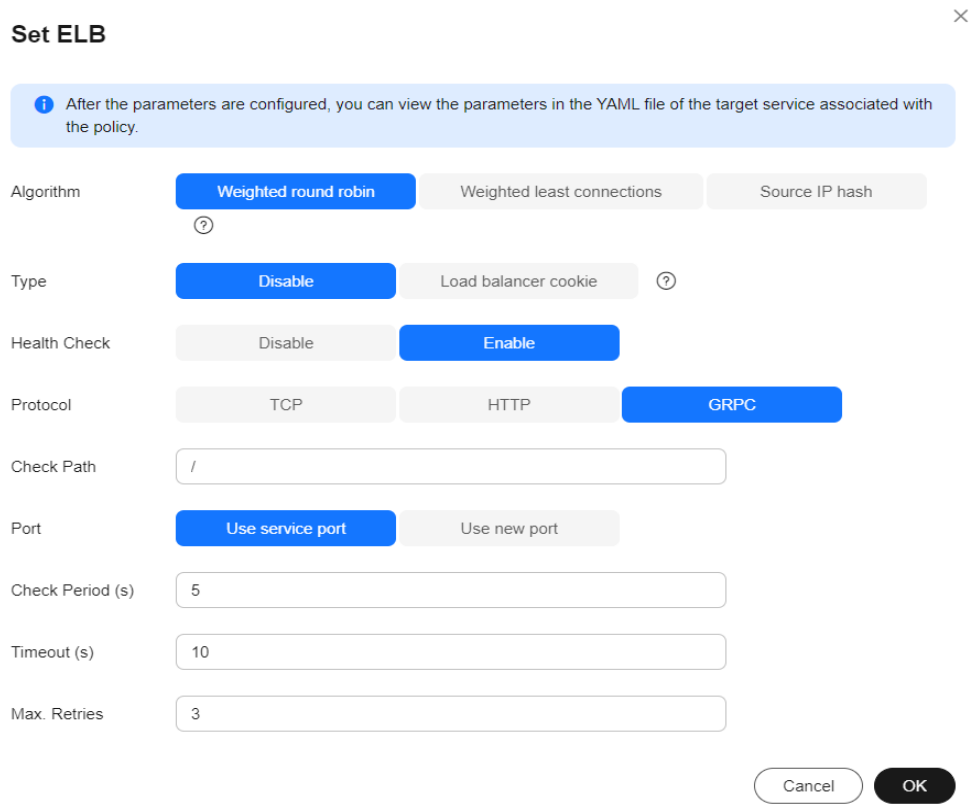


Figure 7-70 Setting the health check protocol to gRPC



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '443'
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing dedicated load balancer is used. Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.pool-protocol: grpc  # Interconnected gRPC backend service
    kubernetes.io/elb.http2-enable: 'true' # Enable HTTP/2.
    kubernetes.io/elb.tls-ciphers-policy: tls-1-2
spec:
  tls:
    - secretName: ingress-test-secret
  rules:
```



```
  - host: ''
    http:
      paths:
        - path: '/'
          backend:
            service:
              name: <your_service_name>  # Replace it with the name of your target Service.
              port:
                number: 80
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
  ingressClassName: cce
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

----End

7.4.3.4.8 Configuring Timeout for a LoadBalancer Ingress

LoadBalancer ingresses support the following timeout settings:

- Idle timeout setting for client connections: maximum duration for keeping a connection when no client request is received. If no request is received during this period, the load balancer closes the connection and establishes a new one with the client when the next request arrives.
- Timeout for waiting for a request from a client: If the client fails to send a request header to the load balancer during the timeout duration or the interval for sending body data exceeds a specified period, the load balancer will release the connection.
- Timeout setting for waiting for a response from a backend server: If the backend server fails to respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout to the client.

Prerequisites

- A CCE standard or Turbo cluster is available. The table below shows which cluster versions support the timeout configuration.

Timeout Type	Load Balancer Type	Supported Cluster Version
Idle Timeout	Dedicated	• v1.19: v1.19.16-r30 or later • v1.21: v1.21.10-r10 or later • v1.23: v1.23.8-r10 or later • v1.25: v1.25.3-r10 or later • Other clusters of later versions
Request Timeout	Dedicated	
Response Timeout	Dedicated	
Idle Timeout	Shared	• v1.23: v1.23.13-r0 or later • v1.25: v1.25.8-r0 or later • v1.27: v1.27.5-r0 or later • v1.28: v1.28.3-r0 or later • Other clusters of later versions
Request Timeout	Shared	
Response Timeout	Shared	

- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- If you delete the timeout configuration when updating an ingress, the timeout configuration of the existing listener will be retained.
- If multiple ingresses share the same external port on a load balancer, you are advised to use the same timeout configuration for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring Multiple Ingresses to Use the Same External ELB Port**.

Configuring Timeout

You can configure timeout for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

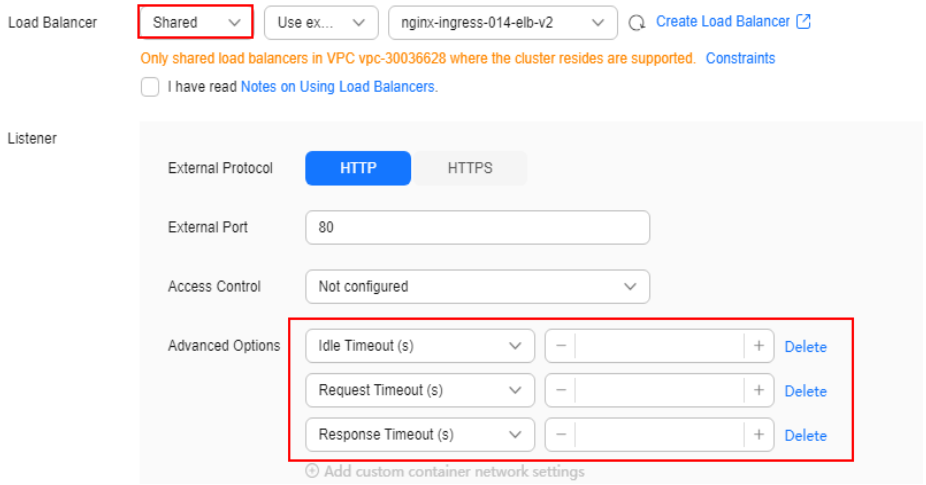
This example explains only key parameters for configuring timeout. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-111 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. A load balancer can be dedicated or shared.	Dedicated
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener. • **Advanced Options** – **Idle Timeout (s)**: specifies the idle timeout of a client connection. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request. – **Request Timeout (s)**: specifies the timeout duration for waiting for a client request. Specifically: If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected. – **Response Timeout (s)**: specifies the timeout duration for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond during the timeout duration, the load balancer will stop waiting and return HTTP 504 Gateway Timeout.	• External Protocol: HTTP • External Port: 80 • Advanced Options – Idle Timeout (s): 60 – Request Timeout (s): 60 – Response Timeout (s): 60

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-71 Configuring timeout



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id>    # In this example, an existing dedicated load balancer is used. Replace its ID with the ID of your dedicated load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.keepalive_timeout: '300'  # Timeout setting for client connections
    kubernetes.io/elb.client_timeout: '60'      # Timeout duration for waiting for a request from a client
    kubernetes.io/elb.member_timeout: '60'      # Timeout for waiting for a response from a backend server
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: test
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-112 Key annotation parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.keepalive_timeout	No	String	Timeout for client connections. If there are no requests reaching the load balancer during the timeout duration, the load balancer will disconnect the connection from the client and establish a new connection when there is a new request. The value ranges from **0** to **4000** (in seconds). The default value is **60**.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.client_timeout	No	String	Timeout for waiting for a request from a client. There are two cases: • If the client fails to send a request header to the load balancer during the timeout duration, the request will be interrupted. • If the interval between two consecutive request bodies reaching the load balancer is greater than the timeout duration, the connection will be disconnected. The value ranges from **1** to **300** (in seconds). The default value is **60**.
kubernetes.io/elb.member_timeout	No	String	Timeout for waiting for a response from a backend server. After a request is forwarded to the backend server, if the backend server does not respond within the duration specified by **member_timeout**, the load balancer will stop waiting and return HTTP 504 Gateway Timeout. The value ranges from **1** to **300** (in seconds). The default value is **60**.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce     *        121.**.**.**    80      10s
```

----End

7.4.3.4.9 Configuring a Slow Start for a LoadBalancer Ingress

With slow start configured, a load balancer linearly increases the proportion of requests to backend server pods. When the slow start duration elapses, the load balancer sends full share of requests to backend pods and exits the slow start mode. Slow start gives applications time to warm up and respond to requests with optimal performance.

NOTE

After a slow start is configured, if you delete the target annotation from the YAML file, slow start will not be enabled.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version is v1.23 or later.
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- Only dedicated load balancers support slow start for HTTP and HTTPS backend server groups.
- Slow start takes effect only when the weighted round robin algorithm is used.
- Slow start takes effect only for new backend server pods.
- After the slow start duration elapses, backend servers will not enter the slow start mode again.
- Slow start takes effect when health check is enabled and backend server pods are running properly.
- If health check is disabled, slow start takes effect immediately.
- With slow start configured for an ingress, all forwarding policies of the ingress take effect.

Procedure

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: <your_elb_id> # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.slowstart: '30' #Configure a slow start.
spec:
  rules:
    - host: ''
      http:
        paths:
```



```
      - path: /
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80               # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
ingressClassName: cce
```

Table 7-113 Parameters for configuring a slow start

Parameter	Mandatory	Type	Description
kubernetes.io/elb.slowstart	No	String	Specifies whether to enable slow start. After you enable it, the load balancer linearly increases the proportion of requests to backend server pods in this mode. When the slow start duration elapses, the load balancer sends full share of requests to backend server pods and exits the slow start mode. **Clusters later than v1.23 support this field.** The slow start duration ranges from **30** to **1200**. Duration of slow start, in seconds. ● Grayscale release applies only to dedicated load balancers. ● This parameter is valid only when the allocation policy of the target Service is weighted round robin (WRR) and sticky session is disabled.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce      *         121.**.**.**    80      10s
```

----End

7.4.3.4.10 Configuring Grayscale Release for a LoadBalancer Ingress

Dedicated LoadBalancer ingresses support grayscale release by:

- Proportion
- HTTP request header

- Cookie

NOTE

ELB is necessary for grayscale release of ingresses. To use grayscale release, submit a service ticket to request the enabling of ELB grayscale release.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions

Notes and Constraints

- After a grayscale ingress is created, do not delete the original ingress.
- When configuring multiple grayscale release policies for ingresses associated with a load balancer listener, the policy based on HTTP request headers takes the highest priority. The cookie-based policy has the second highest priority. The proportion-based policy has the lowest priority.
- This feature relies on the ELB advanced forwarding policies. Once enabled, the priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Deploy the original service.

1. In the navigation pane, choose **Workloads**. Then, click **Create Workload** in the upper right corner and deploy a workload named **origin-server**. For details about parameter settings, see **Creating a Deployment**.
2. In the navigation pane, choose **Services & Ingresses**. On the **Services** tab page, click **Create Service** in the upper right corner. Create a Service named **origin-service** and associate it with the created **origin-server** workload. For a CCE standard cluster, create a NodePort Service, and for a CCE Turbo cluster, create a ClusterIP Service.
3. In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab page and then **Create Ingress** in the upper right corner. Create an ingress named **origin-ingress** and associate it with the created **origin-service** Service. For details about parameter settings, see **Creating a LoadBalancer Ingress on the Console**.

Step 3 Grayscale release a new version for the Service.

1. In the navigation pane, choose **Services & Ingresses**, switch to the **Ingresses** tab, locate the row containing the ingress for which you want to configure

grayscale release, and choose **More** > **Create Grayscale Release** in the **Operation** column.

2. Configure grayscale release parameters.

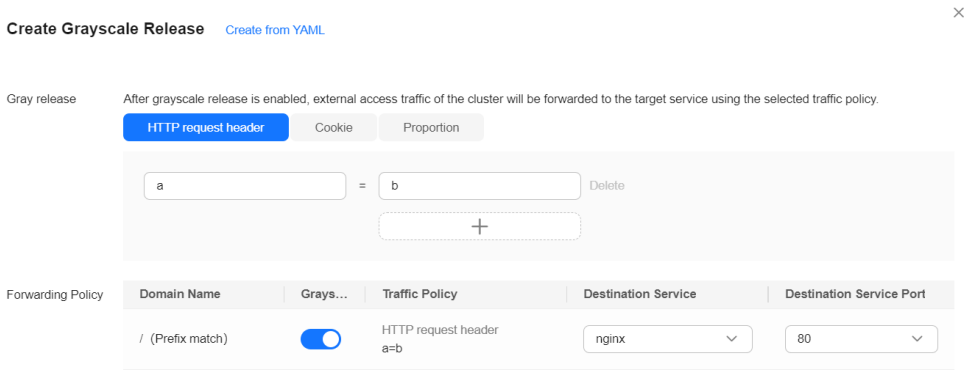
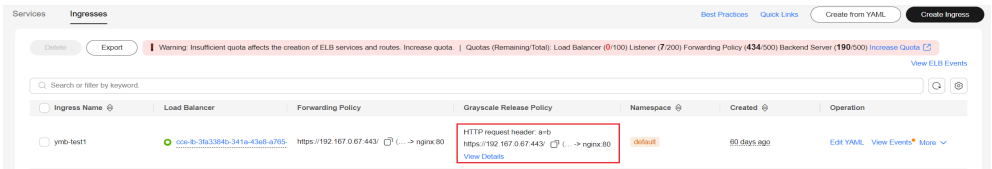


Table 7-114 Key parameters

Parameter	Description	Example
Gray release	– **HTTP request header**: When the header key-value pair in an HTTP request is matched, the grayscale release service can be accessed. ▪ If domain names and paths are configured for an ingress forwarding policy, a maximum of eight values can be configured. ▪ If only paths are configured for an ingress forwarding policy, a maximum of nine values can be configured. – **Cookie**: When the cookie key-value pair in an HTTP request is matched, the grayscale release service can be accessed. – **Proportion**: percentage of the requests for accessing the grayscale release service to the total number of requests	HTTP request header – Key: **a** – Value: **b**

Parameter	Description	Example
Forwarding Policy	– **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. – **Grayscale Release**: After grayscale release is enabled for a domain name, external access traffic of the cluster will be forwarded to the target Service using the selected traffic policy. – **Traffic Policy**: Select an option from **HTTP request header**, **Cookie**, and **Proportion** as needed. (The percentage of a grayscale request is 0%.) – **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. – **Destination Service Port**: Select the access port of the destination Service.	– Domain Name: You do not need to configure this parameter. – Grayscale Release: enabled – Traffic Policy: HTTP request header – Destination Service: nginx – Destination Service Port: 80

3. Click **OK**.
4. Check whether the new version is grayscale released.



Step 4 Terminate the old-version Service.

After the Service of the new version runs stably, terminate the Service of the old version. During the Service termination, change the original Service to the new-version Service so that all traffic will be routed to the new-version Service. Then, delete the grayscale released ingress.

----End

Using kubectl

Step 1 Deploy the original service.

1. Deploy a workload named **origin-server**.
2. Deploy a Service named **origin-service** and associate it with the created **origin-server** workload. (Create a NodePort Service in a CCE standard cluster or a ClusterIP Service in a CCE Turbo cluster.)

3. Deploy an ingress named **origin-ingress** and associate it with the created **origin-service** Service.

   Configure the ingress as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: origin-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.port: '81'
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.class: performance
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: origin-service
                port:
                  number: 81
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

Step 2 Grayscale release a new version for the Service.

Deploy the workload, Service, and ingress of a new version so that traffic can be routed to the Service of the new version by weight or header.

1. Deploy a workload named **canary-server**.
2. Deploy a Service named **canary-service** and associate it with the created **canary-server** workload. (Create a NodePort Service in a CCE standard cluster or a ClusterIP Service in a CCE Turbo cluster.)
3. Create a grayscale released ingress. For details about parameter settings, see **Parameters**.

   a. **Scenario 1, by HTTP request header**: When the header key-value pair in an HTTP request is matched, the grayscale released Service is accessed.

      Deploy an ingress named **canary-header-ingress** and associate it with the created **canary-service** Service for grayscale release by header.

      Configure the ingress as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: canary-header-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.canary: 'true'                    # Set this parameter to true, indicating that canary annotation is enabled.
    kubernetes.io/elb.canary-by-header: 'location'      # Set the header key to location.
    kubernetes.io/elb.canary-by-header-value: '{"values":["hz","sz","sh"]}'  # Set the header values to hz, sz, and sh.
    kubernetes.io/elb.class: performance        # Only dedicated load balancers support grayscale release.
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.port: '81'
spec:
  ingressClassName: cce
  rules:
```



```
  - host: example.com
    http:
      paths:
        - path: /
          pathType: ImplementationSpecific
          backend:
            service:
              name: canary-service
              port:
                number: 80
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

b. **Scenario 2: by proportion**: Configure the proportion of requests for accessing the grayscale released Service.

Deploy an ingress named **canary-weight-ingress** and associate it with the created **canary-service** Service for grayscale release by weight.

Configure the ingress as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: canary-weight-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.canary: 'true'             # Set this parameter to true, indicating that canary annotation is enabled.
    kubernetes.io/elb.canary-weight: '40'        # Configure a weight, indicating that 40% of the request traffic will be routed to the Service of the new version.
    kubernetes.io/elb.class: performance         # Only dedicated load balancers support grayscale release.
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.port: '81'
spec:
  ingressClassName: cce
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: canary-service
                port:
                  number: 81
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

c. **Scenario 3, by cookie**: When the cookie key-value pair in a request is matched, the grayscale released Service is accessed.

Deploy an ingress named **canary-cookie-ingress** and associate it with the created **canary-service** Service for grayscale release by cookie.

Configure the ingress as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: canary-cookie-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.canary: 'true'                 # Set this parameter to true, indicating that canary annotation is enabled.
    kubernetes.io/elb.canary-by-cookie: 'location'     # Set the header key to location.
    kubernetes.io/elb.canary-by-cookie-value: '{"values":["hz","sz","sh"]}'  // Set the header values to hz, sz, and sh.
    kubernetes.io/elb.class: performance               # Only dedicated load balancers support grayscale release.
```



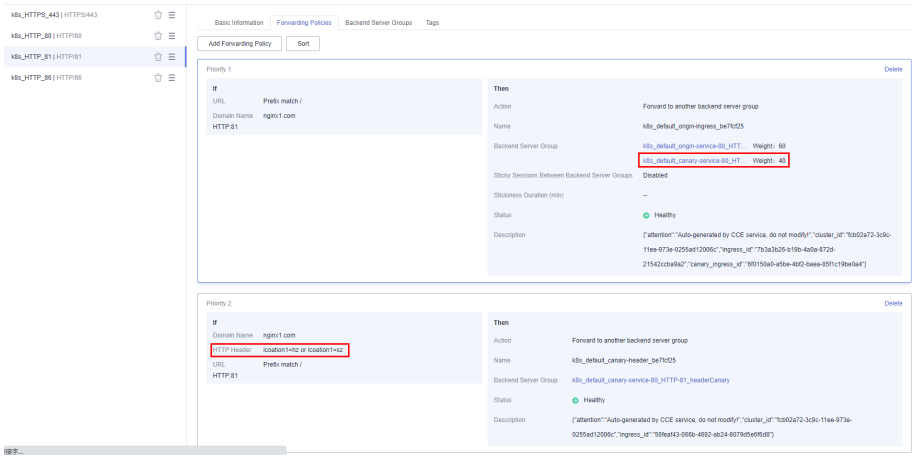
```
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.port: '81'
spec:
  ingressClassName: cce
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: canary-service
                port:
                  number: 81
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

4. Check whether the new version is grayscale released.

   a. Check the creation of the ingress.

   ```
   $ kubectl get ingress
   NAME                     CLASS   HOSTS         ADDRESS         PORTS   AGE
   canary-cookie-ingress    cce     example.com   88.88.88.165    80      16s
   canary-header-ingress    cce     example.com   88.88.88.165    80      46s
   canary-weight-ingress    cce     example.com   88.88.88.165    80      117s
   origin-ingress           cce     example.com   88.88.88.165    80      2m25s
   ```

   b. Check ELB rules.

   c. Check the result of a service request.

      i. Use the **location: hz** request header to access the Service.

      ```
      $ curl -H "Host: example.com" -H "location: hz" http://88.88.88.165:81/
      ```

      Expected result:

      ```
      $ new
      ```

      Attempt the access for multiple times. All the returned results are **new**.

      ii. Access the Service without using a request header.

      ```
      $ curl -H "Host: example.com" http://88.88.88.165:81/
      ```

      Expected result: 40% for new, and 60% for old

Step 3 Terminate the old-version Service.

After the Service of the new version runs stably, terminate the Service of the old version. During the Service termination, change the original Service to the new-version Service so that all traffic will be routed to the new-version Service. Then, delete the grayscale released ingress.

1. Change the original Service to the new-version Service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: origin-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.port: '81'
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.class: performance
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: canary-service
                port:
                  number: 81
              property:
                ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

2. Check whether the old-version Service has been terminated.
   a. Initiate a request with a header and a request without a header to access the Service.

```
$ curl -H "Host: example.com" -H "location: hz" http://88.88.88.165:81/
$ curl -H "Host: example.com" http://88.88.88.165:81/
```

   b. Expected result: All results are returned from the new-version Service.
3. Delete the grayscale released ingress.

```
$ kubectl delete ingress canary-weight-ingress canary-header-ingress
```

----End

Parameters

Table 7-115 Parameters for grayscale release

Parameter	Mandatory	Type	Description
kubernetes.io/elb.canary	No	String	Grayscale release status for an ingress. If this parameter is set to **true**, the implementation of grayscale release varies depending on annotation configurations. Option: **true** • Grayscale release applies only to dedicated load balancers. • If this parameter is set to **true**, the configuration cannot be deleted or modified.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.canary-weight	No	String	Weight of a grayscale release. After this parameter is configured, the ingress will be grayscale released based on the weight. ● The value is a positive integer ranging from 0 to 100. It is a percentage of traffic for routing. ● This parameter is mandatory when an ingress is grayscale released by weight. ● Do not configure this parameter with other grayscale release functions.
kubernetes.io/elb.session-affinity-mode	No	String	After weight-based grayscale release is enabled, configure sticky session. This parameter can only be set to **HTTP_COOKIE** for grayscale release.
kubernetes.io/elb.session-affinity-option	No	String	Sticky session timeout after sticky session is enabled for weight-based grayscale release. The parameter value is a JSON string in the following format: `{"persistence_timeout": "1440"}` Parameters: ● Timeout range: 1 to 1440 ● Default value: **1440**
kubernetes.io/elb.canary-by-header	No	String	Key of an ingress header used for grayscale release, indicating the name of a request header. This parameter must be used with **kubernetes.io/elb.canary-by-header-value**. Parameters: Enter 1 to 40 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.canary-by-header-value	No	String	Value of an ingress header used for grayscale release. This parameter must be used with **kubernetes.io/elb.canary-by-header**. The parameter value is an array in JSON format, for example: `'{"values":["a","b"]}'` Parameters: ● Array length: At least one value must be configured. – If domain names and paths are configured for an ingress forwarding policy, a maximum of eight values can be configured. – If only paths are configured for an ingress forwarding policy, a maximum of nine values can be configured. ● Enter 1 to 128 characters. Asterisks (*) and question marks (?) are allowed, but spaces and double quotation marks are not allowed. An asterisk can match zero or more characters, and a question mark can match one character.
kubernetes.io/elb.canary-by-cookie	No	String	Key of cookie-based grayscale release, indicating the name of a request cookie. This parameter must be used with **kubernetes.io/elb.canary-by-cookie-value**. Parameters: Enter 1 to 100 characters, including letters, digits, and other characters (!%'"()*+,./:=?@^\\-_`~).

Parameter	Mandatory	Type	Description
kubernetes.io/elb.canary-by-cookie-value	No	String	Value of cookie-based grayscale release. This parameter must be used with **kubernetes.io/elb.canary-by-cookie**. The parameter value is an array in JSON format, for example: `'{"values":["a","b"]}'` Parameters: ● Array length: At least one value must be configured. – If domain names and paths are configured for an ingress forwarding policy, a maximum of eight values can be configured. – If only paths are configured for an ingress forwarding policy, a maximum of nine values can be configured. ● Enter 1 to 100 characters, including letters, digits, and other characters (! %'"()*+,./:=?@^\\-_`~). Spaces are not allowed.
kubernetes.io/elb.canary-related-ingress-uid	No	String	UID of the original ingress associated with the grayscale release ingress, which is used to display the association between the original ingress and the grayscale release ingress. ● Format: character string ● Value: **metadata.uid** of the original ingress

7.4.3.4.11 Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Ingress

You can add IP addresses to a trustlist or blocklist to control access to a listener of a LoadBalancer ingress.

- Trustlist: Only the IP addresses in the list can access the listener.
- Blocklist: The IP addresses in the list are not allowed to access the listener.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.12-r0 or later
  - v1.25: v1.25.7-r0 or later
  - v1.27: v1.27.4-r0 or later

  - v1.28: v1.28.2-r0 or later
  - Other clusters of later versions
- An IP address group has been created on the ELB console. For details, see **IP Address Group**.

Notes and Constraints

- After a blocklist/trustlist access policy is configured, if you delete the blocklist/trustlist access policy configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.
- If multiple ingresses share the same external port on a load balancer, you are advised to use the same blocklist/trustlist configuration for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring Multiple Ingresses to Use the Same External ELB Port**.

Configuring a Blocklist/Trustlist Access Policy

You can configure a blocklist/trustlist access policy for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

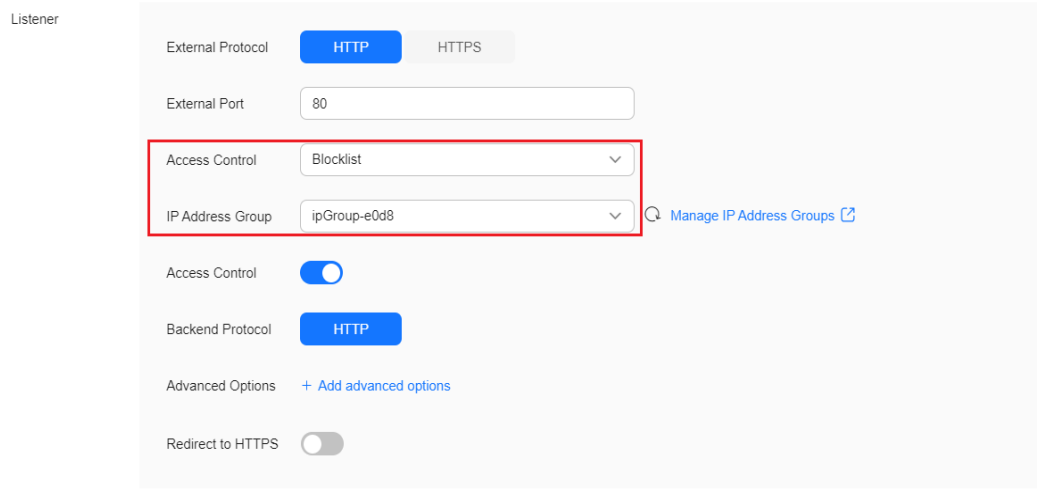
This example explains only key parameters for configuring blocklist/trustlist access policies. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-116 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. A load balancer can be dedicated or shared.	Dedicated

Parameter	Description	Example
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener. • **Access Control** – **Inherit ELB Configurations**: CCE does not modify the existing access control configurations on the ELB console. – **Allow all IP addresses**: No access control is configured. – **Trustlist**: Only the selected IP address group can access the load balancer. – **Blocklist**: The selected IP address group cannot access the load balancer. **NOTE** For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can select a maximum of five IP address groups for access control at a time.	• External Protocol: HTTP • External Port: 80 • Advanced Options Access Control: Blocklist
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-72 Configuring a blocklist/trustlist access policy



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>               # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance              # Load balancer type
    kubernetes.io/elb.port: '80'                      # External port of the load balancer listener
    kubernetes.io/elb.acl-id: <your_acl_id>           # ID of an IP address group for accessing a load
balancer
    kubernetes.io/elb.acl-status: 'on'                # Enable access control.
    kubernetes.io/elb.acl-type: 'white'               # Trustlist for access control
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80               # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```


Table 7-117 Annotations for ELB access control

Parameter	Type	Description
kubernetes.io/elb.acl-id	String	• If this parameter is not specified, CCE does not modify access control on the ELB. • If this parameter is left empty, all IP addresses are allowed to access the load balancer. • If this parameter is set to the IP address group ID of the load balancer, access control is enabled and you need to configure an IP address blocklist or trustlist for the load balancer. Additionally, you need to configure both **kubernetes.io/elb.acl-status** and **kubernetes.io/elb.acl-type**. **NOTE** For clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, or later, you can enter up to five IP address groups and separate them with commas (,). **How to obtain**: Log in to the console. In the **Service List**, choose **Networking** > **Elastic Load Balance**. On the Network Console, choose **Elastic Load Balance** > **IP Address Groups** and copy the **ID** of the target IP address group. For details, see **Creating an IP Address Group**.
kubernetes.io/elb.acl-status	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **on**: Access control is enabled. • **off**: Access control is disabled.
kubernetes.io/elb.acl-type	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. Options: • **black**: indicates a blocklist. The selected IP address group cannot access the load balancer. • **white**: indicates a trustlist. Only the selected IP address group can access the load balancer.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```


If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce     *        121.**.**.**    80      10s
```

----End

7.4.3.4.12 Configuring a Range of Listening Ports for a LoadBalancer Ingress

Ingress allows you to customize listening ports. You can configure both HTTP and HTTPS listeners for a Service. For example, a Service can make available both HTTP port 80 and HTTPS port 443 for external access.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

The following shows an example configuration using an existing load balancer:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.id: 2c623150-17bf-45f1-ae6f-384b036f547e     # ID of an existing load balancer
    kubernetes.io/elb.class: performance     # Load balancer type
    kubernetes.io/elb.listen-ports: '[{"HTTP": 80},{"HTTPS": 443}]'    # Multi-listener configuration
    kubernetes.io/elb.tls-certificate-ids:
6cfb43c9de1a41a18478b868e34b0a82,6cfb43c9de1a41a18478b868e34b0a82   # HTTPS certificate
configuration
  name: ingress-test
  namespace: default
```



```
spec:
  ingressClassName: cce
  rules:
  - host: example.com
    http:
      paths:
      - backend:
          service:
            name: test
            port:
              number: 8888
        path: /
        pathType: ImplementationSpecific
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-118 Annotations for custom listening ports

Parameter	Type	Description
kubernetes.io/elb.listen-ports	String	Configure multiple listening ports for an ingress. The port number ranges from 1 to 65535. The following is an example for JSON characters: `kubernetes.io/elb.listen-ports: '[{"HTTP":80},{"HTTPS":443}]'` • Only the listening ports that comply with both HTTP and HTTPS are allowed. • This function is available only for newly created ingresses in clusters of a version earlier than v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, or v1.30.4-r0. Additionally, after you configure multiple listening ports, the annotations cannot be modified or deleted. In clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later, the annotations can be modified or deleted. • If both **kubernetes.io/elb.listen-ports** and **kubernetes.io/elb.port** are configured, **kubernetes.io/elb.listen-ports** takes a higher priority. • Ingress configuration items such as the blocklist, trustlist, and timeout concurrently take effect on multiple listening ports. When HTTP/2 is enabled for an ingress, HTTP/2 takes effect only on the HTTPS port.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:


```
NAME          CLASS    HOSTS         ADDRESS         PORTS   AGE
ingress-test  cce      example.com   121.**.**.**    80,443  10s
```

----End

7.4.3.4.13 Configuring an HTTP/HTTPS Header for a LoadBalancer Ingress

HTTP headers are a list of strings sent and received by both the client and server on every HTTP request and response. This section describes HTTP headers supported by HTTP and HTTP listeners.

NOTE

- HTTP/HTTPS headers rely on ELB. Before using HTTP/HTTPS headers in a Service, check whether HTTP/HTTPS headers are supported in the current region. For details, see **HTTP/HTTPS Headers**.
- After HTTP or HTTPS is configured, if you delete the HTTP or HTTPS configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Table 7-119 Headers

Header	Feature	Description
X-Forwarded-Port	Transfer Listener Port Number	If this option is enabled, the port number used by the listener will be transmitted to backend servers through the **X-Forwarded-Port** header.
X-Forwarded-For-Port	Transfer Port Number in the Request	If this option is enabled, the port number used by the client will be transmitted to backend servers through the **X-Forwarded-For-Port** header.
X-Forwarded-Host	Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.13-r0 or later
  - v1.25: v1.25.8-r0 or later
  - v1.27: v1.27.5-r0 or later
  - v1.28: v1.28.3-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.

- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- Ingresses support HTTP to HTTPS headers only when dedicated load balancers are used.
- If multiple ingresses share the same external port on a load balancer, you are advised to use the same HTTP/HTTPS header for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring Multiple Ingresses to Use the Same External ELB Port**.

Configuring an HTTP/HTTPS Header

You can configure an HTTP/HTTPS header for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

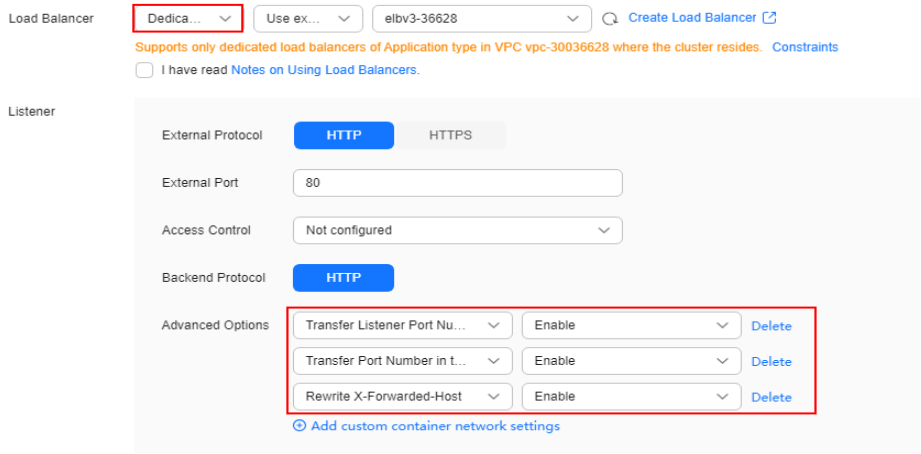
This example explains only key parameters for configuring HTTP/HTTPS headers. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-120 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated

Parameter	Description	Example
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener. • **Advanced Options** – **Transfer Listener Port Number**: If this function is enabled, the listening port on the load balancer can be transferred to backend servers through the HTTP header of the packet. – **Transfer Port Number in the Request**: If this function is enabled, the source port on the client can be transferred to backend servers through the HTTP header of the packet. – **Rewrite X-Forwarded-Host**: If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request header and transferred to backend servers.	• External Protocol: HTTP • External Port: 80 • **Advanced Options** – Transfer Listener Port Number: Enable – Transfer Port Number in the Request: Enable – Rewrite X-Forwarded-Host: Enable
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-73 Configuring HTTP/HTTPS headers



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>              # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance              # Load balancer type
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.x-forwarded-port: 'true'       # Obtain the listener port number.
    kubernetes.io/elb.x-forwarded-for-port: 'true'   # Obtain the client port number for requests.
    kubernetes.io/elb.x-forwarded-host: 'true'       # Rewrite X-Forwarded-Host.
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 80               # Replace 80 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```


Table 7-121 Key parameters

Parameter	Type	Description
kubernetes.io/elb.x-forwarded-port	String	A load balancer can obtain the port number of a listener using **X-Forwarded-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a listener port number. • **false**: Disable the function of obtaining a listener port number.
kubernetes.io/elb.x-forwarded-for-port	String	A load balancer can obtain a client port number for requests using **X-Forwarded-For-Port** and transmit the port number to the packets of the backend server. • **true**: Enable the function of obtaining a client port number for requests. • **false**: Disable the function of obtaining a client port number for requests.
kubernetes.io/elb.x-forwarded-host	String	• **true**: Enable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header will be rewritten using the **Host** header of the client request and transmitted to backend servers. • **false**: Disable the function of rewriting **X-Forwarded-Host**. Then, the **X-Forwarded-Host** header of the client will be transmitted to backend servers.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce      *         121.**.**.**    80      10s
```

----End

7.4.3.4.14 Configuring GZIP Data Compression for a LoadBalancer Ingress

LoadBalancer ingresses support data compression, which reduces the size of files to be transferred, improves file transfer efficiency, and reduces the bandwidth needed for the transmission.

NOTE

- This function relies on ELB capabilities. Before using this function, check whether it is supported in the current region. For details about the regions where this function is supported, see **Elastic Load Balance Function Overview**.
- After data compression is configured, if you delete the data compression configuration on the CCE console or delete the target annotation from the YAML file, the configuration on the ELB will be retained.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- Ingresses support GZIP data compression only when dedicated load balancers are used.
- If multiple ingresses share the same external port on a load balancer, you are advised to use the same GZIP compression configuration for these ingresses. Otherwise, the configuration of the first created ingress will take precedence. For details, see **Configuring Multiple Ingresses to Use the Same External ELB Port**.

Configuring GZIP Data Compression

You can configure GZIP data compression for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

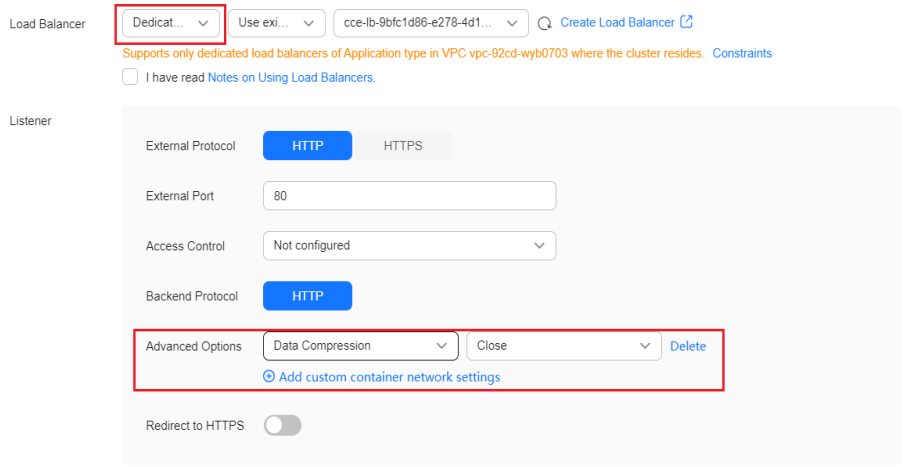
NOTE

This example explains only key parameters for configuring GZIP data compression. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-122 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener. • **Advanced Options** **Data Compression**: If this function is enabled, specific files will be compressed. If you do not enable this function, files will not be compressed. **NOTE** – Brotli can compress all file formats. – GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json	• External Protocol: HTTP • External Port: 80 • Advanced Options Data Compression: Enable
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-74 Configuring GZIP data compression



Step 4 Click **OK**.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>              # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.class: performance              # Load balancer type
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.gzip-enabled: 'true'            # Enable data compression.
spec:
  rules:
  - host: ''
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service.
            port:
              number: 8080             # Replace 8080 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```


Table 7-123 Key parameters

Parameter	Type	Description
kubernetes.io/elb.gzip-enabled	String	• **true**: Data compression is enabled, and specific file types will be compressed. • **false**: Data compression is disabled, and no files will be compressed. By default, data compression is disabled. The files in the following format can be compressed: • Brotli can compress all file formats. • GZIP can compress the files in the following format: text, xml text, plain text, css application, javascript application, x-javascript application, rss+xml application, atom+xml application, xml application, or json This function is available only for HTTP/HTTPS listeners of dedicated load balancers.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce      *         121.**.**.**    80      10s
```

----End

7.4.3.4.15 Configuring URL Redirection for a LoadBalancer Ingress

Ingress can redirect specific access requests to a specified path. The following is an example of YAML file for configuring an ingress redirection rule. In this example, the request for accessing **example.com** is redirected to **example.com/testa** and status code 301 is returned.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later

  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Configuring a Rule for Redirecting an Ingress to a URL

You can configure a rule for redirecting an ingress to a URL using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

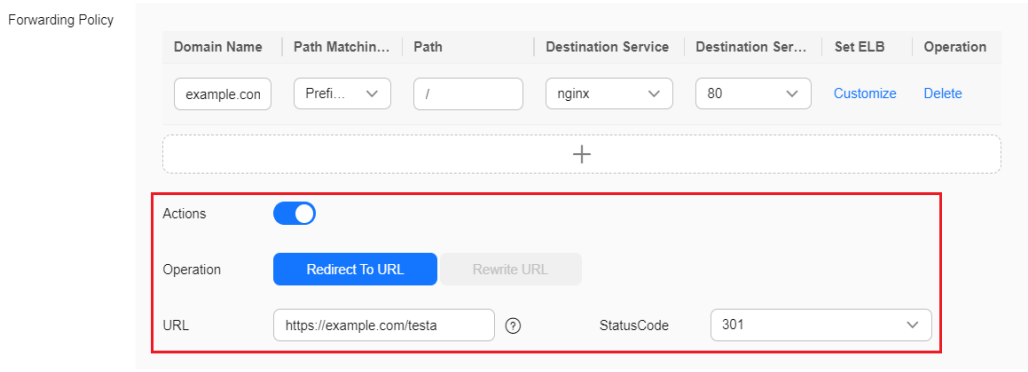
This example explains only key parameters for configuring URL redirection. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-124 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener.	• External Protocol: HTTP • External Port: 80

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service. • **Actions** – **Operation**: Select **Redirect to URL**. When an access request meets the forwarding policy, the request will be redirected to a specified URL, and a specific status code will be returned. – **URL**: A URL starting with **http://** or **https://** is valid. **NOTE** HTTP routes can be redirected to HTTP or HTTPS routes. HTTPS routes can be redirected only to HTTPS routes. – **StatusCode**: The return code can be 301, 302, 303, 307, or 308.	• Domain Name: example.com • Path Matching Rule: Prefix match • Path: / • Destination Service: nginx • Destination Service Port: 80 • Actions: – Operation: Redirect to URL – URL: https://example.com/testa – StatusCode: 301

Figure 7-75 Configuring a rule for redirecting an ingress to a URL



Step 4 Click **OK**.

----End

Using kubectl

An ingress can be redirected to a URL using annotations. Example:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-redirect-url
  namespace: default
  annotations:
    kubernetes.io/elb.id: df76342f-e898-402a-bac8-bde5bf974da8
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.redirect-url: https://example.com/testa       # Information about the redirection to a URL
    kubernetes.io/elb.redirect-url-code: '301'    # Code returned after the ingress is redirected to a URL
spec:
  rules:
    - host: "example.com"
      http:
        paths:
          - path: /
            backend:
              service:
                name: test-service
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```


Table 7-125 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.redirect-url	Yes	String	URL for redirection Format: A valid URL must start with **http://** or **https://**, for example, **https://example.com/**. Parameter: This configuration takes effect on all forwarding rules of a single ingress. After the configuration is deleted, the target URL redirection rule will be automatically cleared. Either this annotation or the annotation of a grayscale release can be configured.
kubernetes.io/elb.redirect-url-code	No	String	Code returned after an ingress is redirected to a URL. Format: The return code can be 301, 302, 303, 307, or 308. Parameter: The default value is **301**.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/test-redirect-url created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME              CLASS    HOSTS         ADDRESS         PORTS   AGE
test-redirect-url  cce     example.com   121.**.**.**    80      10s
```

Step 5 Use curl to verify the redirection, where *${ELB_IP}* is the IP address accessed by the target ingress.

```
curl -I -H "Host:example.com" ${ELB_IP}
```

The access path will be redirected to **example.com/testa**.

```
HTTP/1.1 301 Moved Permanently
Date: Thu, 07 Mar 2024 11:04:31 GMT
Content-Type: text/html
Content-Length: 134
Connection: keep-alive
Location: https://example.com/testa
Server: elb
```

----End

7.4.3.4.16 Configuring URL Rewriting for a LoadBalancer Ingress

Dedicated LoadBalancer ingresses allow you to rewrite the URLs that match a regular expression. To rewrite the URL for such an ingress, do as follows:

- Configure a path that matches a regular expression for the ingress, for example, **/first/(.*)/(.*)/end**.
- Configure a rewrite annotation to match the regular expression in the path.

For example:

- Set path to **/first/(.*)/(.*)/end** and annotation to **/$1/$2**. When the request sent by the user is **/first/aaa/bbb/end**, the path matches **/first/(.*)/(.*)/end**. The rewriting rule replaces **$1** with **aaa** and **$2** with **bbb**, the request path received by the backend server is **/aaa/bbb**.
- Set path to **/first/(.*)/end** and annotation to **/newpath/$1**. When the request sent by the user is **/first/aaa/end**, the path matches **/first/(.*)/end**. The rewriting rule replaces **$1** with **aaa**, the request path received by the backend server is **/newpath/aaa**.

NOTE

This function relies on the ELB capability and is being introduced to all regions. Before using this function, verify if it is available by checking the console options in the current region.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Configuring URL Rewriting for a LoadBalancer Ingress

You can configure a URL rewriting rule for an ingress using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

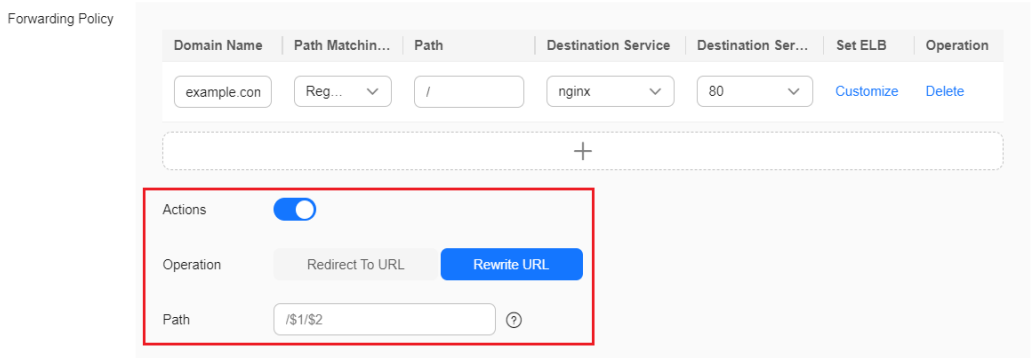
This example explains only key parameters for configuring rewrite. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-126 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener.	• External Protocol: HTTP • External Port: 80

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **RegEx match** if URL rewriting is used. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service. • **Actions** – **Operation**: Select **Rewrite URL**. When an access request meets the forwarding policy, the URL will be rewritten based on the matching rule. You are allowed to configure a regular expression for the patch matching rule, and the result obtained using the regular expression can be used for rewriting the URL. – **Path**: A proper rule matching a regular expression must start with a slash (/).	• Domain Name: example.com • Path Matching Rule: RegEx match • Path: /first/(.*)/(.*)/end • Destination Service: nginx • Destination Service Port: 80 • Actions: – Operation: Rewrite URL – Path: /$1/$2

Figure 7-76 Configuring a URL rewriting rule for an ingress



Step 4 Click **OK**.

----End

Using kubectl

URL rewriting rules of an ingress can be configured using annotations. The following is an example:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-rewrite-url
  namespace: default
  annotations:
    kubernetes.io/elb.id: df76342f-e898-402a-bac8-bde5bf974da8
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.rewrite-target: /$1/$2      # Rewrite path
spec:
  rules:
    - host: "example.com"
      http:
        paths:
          - path: /first/(.*)/(.*)/end      # This path will be rewritten.
            backend:
              service:
                name: test-service
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: REGEX
            pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-127 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.rewrite-target	Yes	String	Information about the rewritten path. Format: A proper rule matching a regular expression must start with a slash (/). Parameter: This configuration takes effect on the URL of a single ingress matching the regular expression. After the configuration is deleted, the target URL rewriting rule will be automatically cleared. Either this annotation or the annotation of a grayscale release can be configured.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/test-rewrite-url created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME               CLASS    HOSTS     ADDRESS         PORTS   AGE
test-rewrite-url   cce      *         121.**.**.**    80      10s
```

Step 5 Use curl to verify the rewriting, where *${ELB_IP}* is the IP address accessed by the target ingress.

```
# curl -H "Host:example.com" ${ELB_IP}/first/aaa/bbb/end
```

The access path will be rewritten to **/aaa/bbb**.

----End

7.4.3.4.17 Redirecting HTTP to HTTPS for a LoadBalancer Ingress

Ingresses can forward HTTP access requests to HTTPS listeners. The following is an example for redirecting the requests for accessing **example.com/test** of an ingress to HTTPS port 443.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.14-r0 or later
  - v1.25: v1.25.9-r0 or later
  - v1.27: v1.27.6-r0 or later
  - v1.28: v1.28.4-r0 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

Ingresses support HTTP to HTTPS redirection only when dedicated load balancers are used.

Redirecting HTTP to HTTPS

You can configure HTTP to HTTPS redirection using either the CCE console or kubectl.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

NOTE

This example explains only key parameters for redirecting HTTP to HTTPS. You can configure other parameters as required. For details, see **Creating a LoadBalancer Ingress on the Console**.

Table 7-128 Key parameters

Parameter	Description	Example
Name	Enter an ingress name.	ingress-test
Load Balancer	Select a load balancer to be associated with the ingress or automatically create a load balancer. In this example, only dedicated load balancers are supported.	Dedicated
Listener	• **External Protocol**: **HTTP** and **HTTPS** are available. • **External Port**: specifies the port of the load balancer listener. **NOTE** After configuration, this setting cannot be modified in clusters of a version earlier than v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, or v1.30.4-r0. In clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later, this setting can be modified. • **Redirect to HTTPS**: After this function is enabled, you can configure the HTTPS port. – **External Port**: Enter an HTTPS port. – **Certificate Source**: TLS secret and ELB server certificates are supported. – **Server Certificate**: Use a certificate created on ELB. If no certificate is available, go to the ELB console and create one. For details, see **Adding a Certificate**.	• External Protocol: HTTP • External Port: 80 • Redirect to HTTPS: enabled – External Port: 443 – Certificate Source: ELB server certificate – Server Certificate: cert-test

Parameter	Description	Example
Forwarding Policy	• **Domain Name**: Enter an actual domain name to be accessed. If it is left blank, the ingress can be accessed through the IP address. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access. • **Path Matching Rule**: Select **Prefix match**, **Exact match**, or **RegEx match**. • **Path**: Enter the path provided by a backend application for external access. The path added must be valid in the backend application, or the forwarding cannot take effect. • **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically. • **Destination Service Port**: Select the access port of the destination Service.	• Domain Name: You do not need to configure this parameter. • Path Matching Rule: RegEx match • Path: / • Destination Service: nginx • Destination Service Port: 80

Figure 7-77 Redirecting HTTP to HTTPS

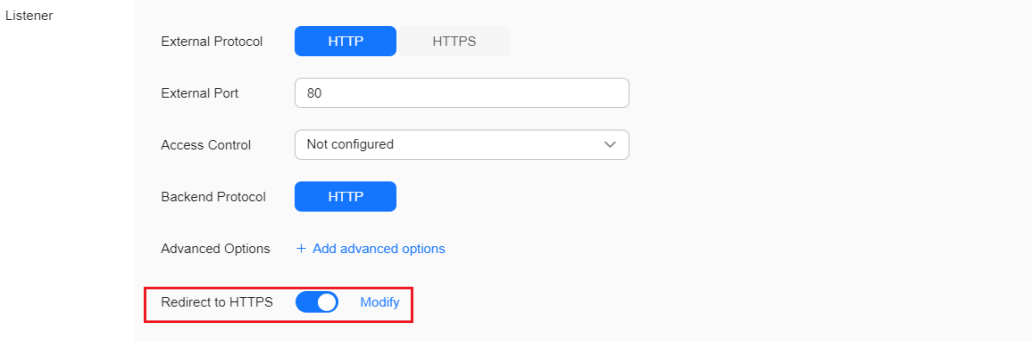
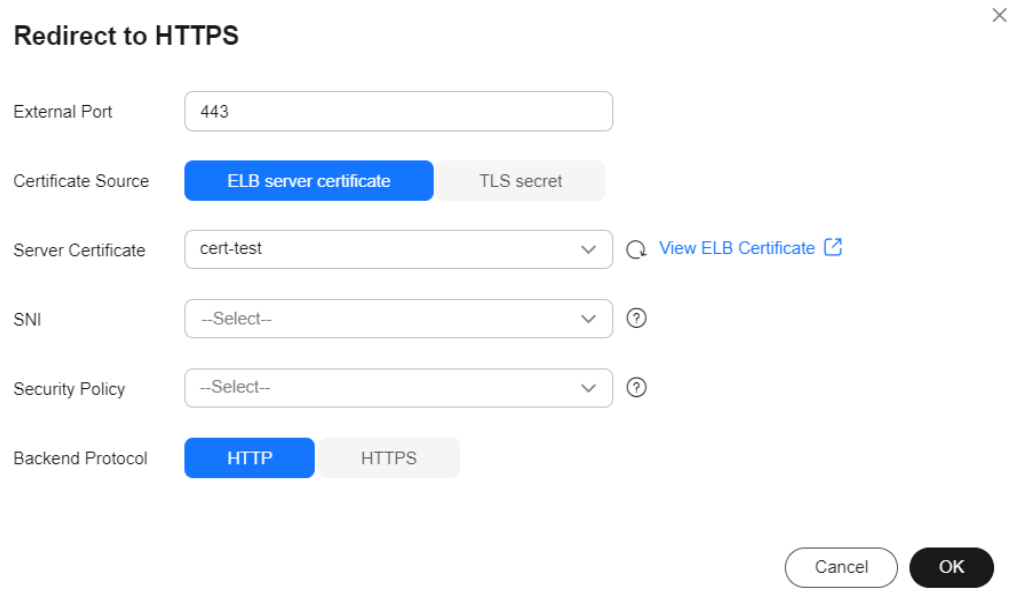


Figure 7-78 Configuring a port for redirecting HTTP to HTTPS



Step 4 Click **OK**.

----End

Using kubectl

You can use annotations to redirect the requests of an ingress to an HTTPS listener. The following is an example:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress associated with an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-redirect-listener
  namespace: default
  annotations:
    kubernetes.io/elb.id: df76342f-e898-402a-bac8-bde5bf974da8
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.listen-ports: '[{"HTTP":80},{"HTTPS":443}]'          # Multi-port configuration
    kubernetes.io/elb.ssl-redirect: 'true'      # Enable redirection from HTTP to HTTPS.
    kubernetes.io/elb.tls-certificate-ids: 6cfb43c9de1a41a18478b868e34b0a82      # HTTPS listener server
certificate
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: test-service
                port:
```



```
            number: 80
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
ingressClassName: cce
```

Table 7-129 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.listen-ports	Yes	String	Multi-port listening configuration. After configuration, this setting cannot be modified in clusters of a version earlier than v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, or v1.30.4-r0. In clusters of v1.23.18-r10, v1.25.16-r0, v1.27.16-r0, v1.28.13-r0, v1.29.8-r0, v1.30.4-r0, or later, this setting can be modified. Format: The value is a JSON string, for example: `[{"HTTP":80},{"HTTPS":443}]` Parameter: The port number ranges from 1 to 65535. **NOTE** You can configure this annotation along with the one for enabling HTTP/2, but HTTP/2 takes effect only on HTTPS ports.
kubernetes.io/elb.ssl-redirect	Yes	String	Whether to enable redirection from HTTP to HTTPS. Format: The value can be **true** or **false**. Parameter: **true** indicates that redirection is enabled. If the value is **false** or the parameter is unavailable, redirection is disabled. **NOTE** Either this annotation or the annotation of a grayscale release can be configured.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/test-redirect-listener created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME                    CLASS   HOSTS   ADDRESS         PORTS   AGE
test-redirect-listener  cce     *       121.**.**.**    80      10s
```

----End

7.4.3.4.18 Configuring the Priorities of Forwarding Rules for LoadBalancer Ingresses

When ingresses use the same load balancer listener, forwarding rules can be prioritized based on the following rules:

- Forwarding rules of different ingresses: The rules are sorted based on the priorities (ranging from 1 to 1000) of the **kubernetes.io/elb.ingress-order** annotation. A smaller value indicates a higher priority.
- Forwarding rules of an ingress: If the **kubernetes.io/elb.rule-priority-enabled** annotation is set to **true**, the forwarding rules are sorted based on the sequence in which they are added during ingress creation. A forwarding rule added earlier indicates a higher priority. If the **kubernetes.io/elb.rule-priority-enabled** annotation is not configured, the default sorting of the forwarding rules on the load balancer will be used.

If the preceding annotations are not configured, the default sorting of the forwarding rules on the load balancer will be used, regardless of whether the forwarding rules are of the same ingress or different ingresses under the same load balancer listener.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.15-r0 or later
  - v1.25: v1.25.10-r0 or later
  - v1.27: v1.27.7-r0 or later
  - v1.28: v1.28.5-r0 or later
  - v1.29: v1.29.1-r10 or later
  - Other clusters of later versions
- An available workload has been deployed in the cluster for external access. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A Service for external access has been configured for the workload. **Services Supported by LoadBalancer Ingresses** lists the Service types supported by LoadBalancer ingresses.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

The following shows an example configuration using an existing load balancer:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '88'
    kubernetes.io/elb.id: 2c623150-17bf-45f1-ae6f-384b036f547e    # ID of an existing load balancer
    kubernetes.io/elb.class: performance    # Load balancer type
    kubernetes.io/elb.ingress-order: '1'    # Priority of a forwarding rule of different ingresses
    kubernetes.io/elb.rule-priority-enabled: 'true'    # The forwarding rules of an ingress are sorted based on the forwarding rule sequence in paths.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /test1
            backend:
              service:
                name: test1
                port:
                  number: 8081
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          - path: /test2
            backend:
              service:
                name: test2
                port:
                  number: 8081
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-130 Annotations for configuring the priorities of forwarding rules

Parameter	Type	Description
kubernetes.io/elb.ingress-order	String	Specifies the sequence of forwarding rules of different ingresses. The value ranges from 1 to 1000. A smaller value indicates a higher priority. The priority of a forwarding rule must be unique under the same load balancer listener. This parameter is available only for dedicated load balancers. **NOTE** When this annotation is configured, the **kubernetes.io/elb.rule-priority-enabled** annotation is enabled by default. The forwarding rules of each ingress will be sorted.

Parameter	Type	Description
kubernetes.io/elb.rule-priority-enabled	String	This parameter can only be set to **true**, indicating to sort the forwarding rules of an ingress. The priorities of the forwarding rules are determined based on the sequence in which they are added during ingress creation. A forwarding rule added earlier indicates a higher priority. If this parameter is not configured, the default sorting of the forwarding rules on the load balancer will be used. After this parameter is enabled, it cannot be disabled. This parameter is available only for dedicated load balancers.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce     *         121.**.**.**     88      10s
```

----End

7.4.3.4.19 Configuring a Custom Header Forwarding Policy for a LoadBalancer Ingress

Dedicated load balancer ingresses support custom header forwarding policies. You can configure different header key-value pairs to determine the backend Service to which data is forwarded.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.16-r0 or later
  - v1.25: v1.25.11-r0 or later
  - v1.27: v1.27.8-r0 or later
  - v1.28: v1.28.6-r0 or later
  - v1.29: v1.29.2-r0 or later
  - Other clusters of later versions
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: 08eab934-1636-4d90-a4cd-cb3fa4330411
    kubernetes.io/elb.class: performance  # A dedicated load balancer is required.
    # Path /a can be accessed only through Header key1=value1 or key1=value2. The accessed Service is svc-a:80.
    kubernetes.io/elb.headers.svc-a: |
      {
          "key": "key1",
          "values": [
              "value1",
              "value2"
          ]
      }
    # Path /b can be accessed only through Header key2=value1 or key2=value2. The accessed Service is svc-b:81.
    kubernetes.io/elb.headers.svc-b: |
      {
          "key": "key2",
          "values": [
              "value1",
              "value2"
          ]
      }
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /a
            backend:
              service:
                name: svc-a
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
          - path: /b
            backend:
              service:
```



```
            name: svc-b
            port:
              number: 81
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-131 Annotations for configuring a custom header forwarding policy

Parameter	Type	Description
kubernetes.io/elb.headers.*${svc_name}*	String	Custom header of the Service associated with an ingress. *${svc_name}* is the Service name. Format: a JSON string, for example, {"key": "test", "values": ["value1", "value2"]} ● **key/value** indicates the key-value pair of the custom header. A maximum of eight values can be configured. Enter 1 to 40 characters for a key. Only letters, digits, hyphens (-), and underscores (_) are allowed. Enter 1 to 128 characters for a value. Asterisks (*) and question marks (?) are allowed, but spaces and double quotation marks are not allowed. An asterisk can match zero or more characters, and a question mark can match one character. ● After a custom header forwarding policy is configured for an ingress, a grayscale release policy cannot be created for the ingress. ● Enter 1 to 51 characters for *${svc_name}*.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce      *         121.**.**.**    80      10s
```

----End

7.4.3.4.20 Configuring a Custom EIP for a LoadBalancer Ingress

You can customize the EIP bound to a load balancer that is automatically created by CCE by adding the **kubernetes.io/elb.custom-eip-id** annotation to an ingress.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.18-r0 or later
  - v1.25: v1.25.13-r0 or later
  - v1.27: v1.27.10-r0 or later
  - v1.28: v1.28.8-r0 or later
  - v1.29: v1.29.4-r0 or later
  - v1.30: v1.30.1-r0 or later
  - Other clusters of later versions
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- A custom EIP for an ingress can be configured only when an ingress is being updated, and the ingress' annotation contains **kubernetes.io/elb.eip-id**.
- A custom EIP must be unbound to any resources.
- After you configure a custom EIP for a load balancer, if the existing EIP bound to the load balancer was automatically created by CCE during load balancer creation and is not being used by any other resources, the existing EIP will be deleted automatically when the associated ingress is deleted. However, if the existing EIP was manually created, it will be unbound from the load balancer when you delete the ingress, and you will need to manually delete the EIP.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Automatically create a load balancer with an EIP bound when creating an ingress. For details, see **Automatically Creating a Load Balancer While Creating an Ingress**.

An example YAML file of an ingress created using a shared load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"test-eip
","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_g-
vm","name":"test-eip"}'
    kubernetes.io/elb.class: union
    kubernetes.io/elb.eip-id: 10183660-0bb7-47d4-a899-18891b1ab2f7     # ID of the EIP automatically
assigned during load balancer creation
    kubernetes.io/elb.id: aed5d5c9-65eb-42ab-9f80-57825cbae309
    kubernetes.io/elb.ip: 1.1.1.1
```



```
    kubernetes.io/elb.port: "80"
  name: test-eip
  namespace: default
spec:
  ingressClassName: cce
  rules:
  - http:
      paths:
      - backend:
          service:
            name: test-eip
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
status:
  loadBalancer:
    ingress:
    - ip: 192.168.1.138
```

Step 3 Modify the ingress configurations and add the **kubernetes.io/elb.custom-eip-id** annotation.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"test-eip
","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_g-
vm","name":"test-eip"}'
    kubernetes.io/elb.class: union
    kubernetes.io/elb.eip-id: 10183660-0bb7-47d4-a899-18891b1ab2f7     # ID of the EIP automatically
assigned during load balancer creation
    kubernetes.io/elb.custom-eip-id: 57bf8bb2-8c7d-4d07-8799-aae16a421802  # ID of the custom EIP
    kubernetes.io/elb.id: aed5d5c9-65eb-42ab-9f80-57825cbae309
    kubernetes.io/elb.ip: 1.1.1.1
    kubernetes.io/elb.port: "80"
  name: test-eip
  namespace: default
spec:
  ingressClassName: cce
  rules:
  - http:
      paths:
      - backend:
          service:
            name: test-eip
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
status:
  loadBalancer:
    ingress:
    - ip: 192.168.1.138
```

Table 7-132 Key parameters

Parameter	Type	Description
kubernetes.io/elb.custom-eip-id	String	ID of the custom EIP, which can be seen on the EIP console The EIP must be bindable.

Step 4 After the ingress is updated, check the ingress again.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"test-eip
","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_g-
vm","name":"test-eip"}'
    kubernetes.io/elb.class: union
    kubernetes.io/elb.eip-id: 10183660-0bb7-47d4-a899-18891b1ab2f7     # ID of the EIP automatically
assigned during load balancer creation
    kubernetes.io/elb.custom-eip-id: 57bf8bb2-8c7d-4d07-8799-aae16a421802  # ID of the custom EIP
    kubernetes.io/elb.custom-eip-status: '{"id":"57bf8bb2-8c7d-4d07-8799-
aae16a421802","public_ip_address":"3.3.3.3"}' # After the custom EIP is configured, record the EIP's ID
and IP address.
    kubernetes.io/elb.id: aed5d5c9-65eb-42ab-9f80-57825cbae309
    kubernetes.io/elb.ip: 1.1.1.1
    kubernetes.io/elb.port: "80"
  name: test-eip
  namespace: default
spec:
  ingressClassName: cce
  rules:
  - http:
      paths:
      - backend:
          service:
            name: test-eip
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
status:
  loadBalancer:
    ingress:
    - ip: 192.168.1.138
```

----End

7.4.3.4.21 Configuring Cross-Origin Access for LoadBalancer Ingresses

The same-origin policy of browsers prevents a web page loaded by one origin from directly requesting resources from another origin in web development. Cross-origin resource sharing (CORS) is a secure solution that allows cross-origin requests.

CORS can be used in the following scenarios:

- Separated frontend and backend: In web development, frontend applications are often deployed under a domain name (for example, app.example.com), while the backend API service uses a different domain name (for example, api.example.com). This can cause cross-origin resource sharing to be blocked, but CORS can help resolve this issue by allowing the frontend to obtain data from the API service.
- Third-party service integration: If your website needs to call third-party APIs (for example, the APIs of a map service or social media login platform), configure CORS to enable cross-origin resource sharing.

- CDN: If your website is using a CDN to provide static resources from a domain name that differs from the primary domain, you can use CORS to load these resources.

NOTE

- ELB is necessary for accessing ingresses across different origins. To use cross-origin access, submit a service ticket to request the enabling of ELB cross-origin access.
- If you enable cross-origin access for a LoadBalancer ingress, you will not be able to enable **URL redirection**, **URL rewriting**, or **HTTP to HTTPS redirection**.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.18-r10 or later
  - v1.25: v1.25.16-r0 or later
  - v1.27: v1.27.16-r0 or later
  - v1.28: v1.28.13-r0 or later
  - v1.29: v1.29.8-r0 or later
  - v1.30: v1.30.4-r0 or later
  - Other clusters of later versions
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/elb.class: performance # Only dedicated load balancers are supported.
    kubernetes.io/elb.cors-allow-origin: 'http://example.com' # The origin that can be accessed
    kubernetes.io/elb.cors-allow-headers: 'fake-header-1' # The allowed request header
    kubernetes.io/elb.cors-expose-headers: 'fake-header-2' #Response header to be exposed
    kubernetes.io/elb.cors-allow-methods: 'GET,POST' # Allowed HTTP request methods
```



```
    kubernetes.io/elb.cors-allow-credentials: 'true' # Credentials can be sent.
    kubernetes.io/elb.cors-max-age: '3600' # Cache duration of a pre-check request
    kubernetes.io/elb.port: '80'
    kubernetes.io/elb.id: 3f5906fb-2b07-4e33-b3fe-b095f03d86a6
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: nginx-03657
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: cce
```

Table 7-133 Annotations for cross-origin access

Parameter	Type	Description	Example Value
kubernetes.io/elb.cors-allow-origin	Array[string]	Specify **Access-Control-Allow-Origin** for the origin that can be accessed. Options: • Wildcard (*): indicates that all domain names can be accessed. • Domain names: start with **http://** or **https://**. Level-1 wildcard domain names are supported. The format is **http(s)://example.com** or **http(s)://example.com:port**, where the port number ranges from 1 to 65535. You can enter multiple values by separating them with commas (,).	kubernetes.io/elb.cors-allow-origin: 'http://example.com'
kubernetes.io/elb.cors-allow-headers	Array[string]	Specify **Access-Control-Allow-Headers** for allowed request headers. You can enter multiple values by separating them with commas (,).	kubernetes.io/elb.cors-allow-headers: 'fake-header-1'

Parameter	Type	Description	Example Value
kubernetes.io/elb.cors-expose-headers	Array[string]	Specify **Access-Control-Expose-Headers** for custom response headers that can be accessed by a cross-origin request. This allows you to retrieve non-standard response headers using client-side JavaScript code. You can enter multiple values by separating them with commas (,).	kubernetes.io/elb.cors-expose-headers: 'fake-header-2'
kubernetes.io/elb.cors-allow-methods	Array[string]	Specify **Access-Control-Allow-Methods** for allowed HTTP request methods. You can enter multiple values by separating them with commas (,).	kubernetes.io/elb.cors-allow-methods: 'GET,POST'
kubernetes.io/elb.cors-allow-credentials	String	Specify **Access-Control-Allow-Credentials** to control the sending of credentials (such as cookies). Options: • **true**: Credentials can be sent. • **false**: Credentials cannot be sent. Once configured, the settings cannot be deleted. To remove the configuration, use **kubernetes.io/elb.cors-disabled** to delete all cross-origin settings.	kubernetes.io/elb.cors-allow-credentials: 'true'
kubernetes.io/elb.cors-max-age	String	Specify **Access-Control-Max-Age** for the cache duration of a CORS pre-check request. Unit: second. Value range: -1 to 172800 Properly configure this parameter based on service requirements. If the value is too low, pre-check requests may happen frequently. If the value is too high, the CORS policy may not take effect immediately.	kubernetes.io/elb.cors-max-age: '3600'

Parameter	Type	Description	Example Value
kubernetes.io/elb.cors-disabled	String	Disable all cross-origin settings. Options: • **true**: disables all cross-origin settings. The parameter values in the YAML file will not be deleted, but all cross-origin settings will not take effect. • **false**: default value, indicating that the cross-origin settings take effect based on your settings.	kubernetes.io/elb.cors-disabled: 'true'

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

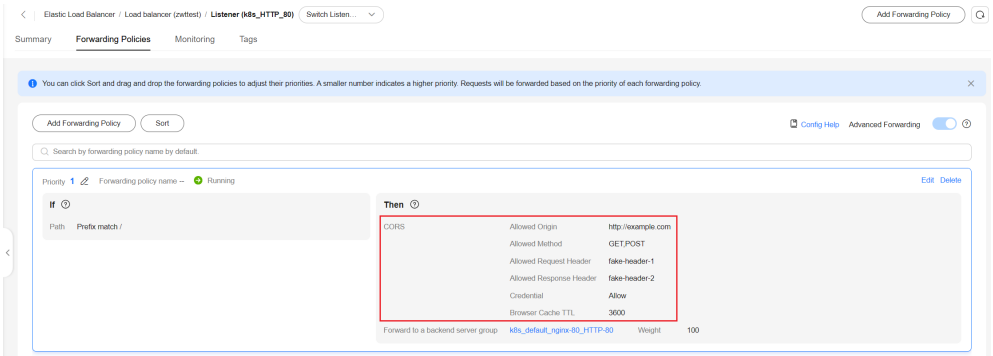
If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce      *         121.**.**.**    80      10s
```

----End

Verifying Cross-Origin Access

1. Log in to the ELB console, locate the load balancer used by the ingress, and check whether the forwarding policy of the listener contains cross-origin configurations.



2. Use curl to send a cross-origin request and check the response header.

```
curl -X OPTIONS -H 'Origin: {Origin}' {Accessed ingress URL}
```

For example, if the origin is **example.com** and the accessed ingress URL is **121.**.**.**:80**, run the following command:

```
curl -X OPTIONS -H 'Origin: example.com' 121.**.**.**:80
```

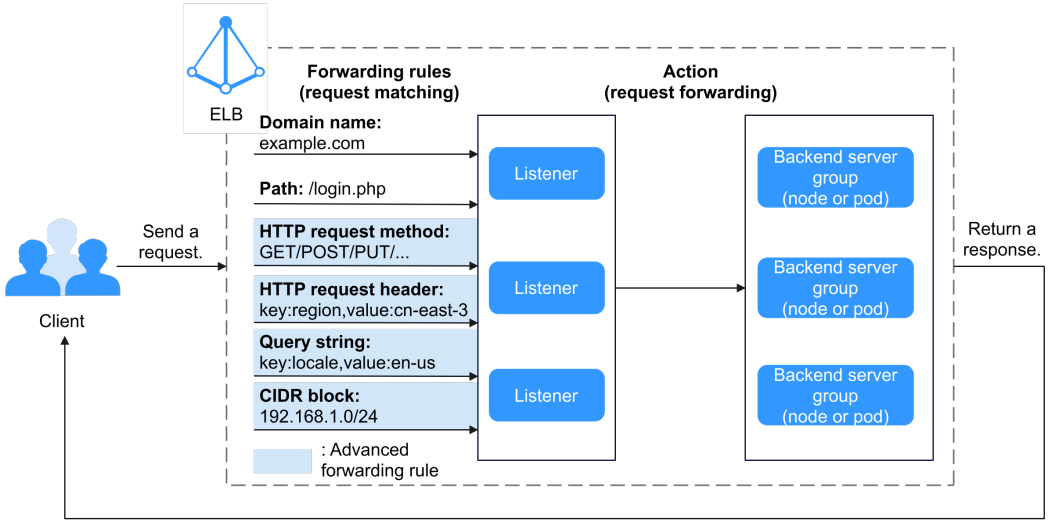
The **Access-Control-**** request header will be added to the original backend response.


```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: fake-header-1
Access-Control-Expose-Headers: fake-header-2
Access-Control-Allow-Methods: GET, POST
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 3600
```

7.4.3.4.22 Configuring Advanced Forwarding Rules for a LoadBalancer Ingress

Ingresses offer diverse forwarding rules that can match listeners based on different request parameters like HTTP request methods, headers, query strings, CIDR blocks, and cookies. Each listener is associated with an ELB access port. This facilitates flexible service distribution and resource allocation.

Figure 7-79 How an advanced forwarding rule operates



Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the following requirements:
  - v1.23: v1.23.18-r10 or later
  - v1.25: v1.25.16-r0 or later
  - v1.27: v1.27.16-r0 or later
  - v1.28: v1.28.13-r0 or later
  - v1.29: v1.29.8-r0 or later
  - v1.30: v1.30.4-r0 or later
  - Other clusters of later versions
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the**

forwarding policy priority as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: ab53c3b2-xxxx-xxxx-xxxx-5ac3eb2887be
    kubernetes.io/elb.port: '80'
    # Access the svc-hello1 service. Ensure that this service is available.
    kubernetes.io/elb.conditions.svc-hello1: |
      [
        {
          "type": "Method",
          "methodConfig": {
            "values": [
              "GET",
              "POST"
            ]
          }
        },
        {
          "type": "Header",
          "headerConfig": {
            "key": "gray-hello",
            "values": [
              "value1",
              "value2"
            ]
          }
        },
        {
          "type": "Cookie",
          "cookieConfig": {
            "values": [
              {
                "key": "querystringkey1",
                "value": "querystringvalue2"
              },
              {
                "key": "querystringkey3",
                "value": "querystringvalue4"
              }
            ]
          }
        },
        {
          "type": "QueryString",
          "queryStringConfig": {
            "key": "testKey",
            "values": [
              "testValue"
            ]
          }
        },
        {
```



```
          "type": "SourceIp",
          "sourceIpConfig": {
            "values": [
              "192.168.0.0/16",
              "172.16.0.0/16"
            ]
          }
        }
      ]
  name: ingress-test
spec:
  ingressClassName: cce
  rules:
   - http:
      paths:
      - path: /hello1
        pathType: ImplementationSpecific
        backend:
          service:
            name: svc-hello1
            port:
              number: 80
```

Table 7-134 Annotations for advanced forwarding rules

Parameter	Type	Description
kubernetes.io/elb.conditions.*${svc_name}*	String	Configure an advanced forwarding rule. *${svc_name}* indicates the Service name, which can contain a maximum of 48 characters. If the annotation value is set to *[]*, the advanced forwarding rule is deleted. An annotation value is in the form of a JSON array. For details, see **Table 7-135**. **NOTICE** The rules in a condition array are connected by an AND relationship, while the values in the same rule block are connected by an OR relationship. For example, if both Method and QueryString are configured, the target traffic can be distributed only when both rules are met. However, if the Method value is GET or POST, the target traffic can be distributed only when both rules are met and the Method value must be GET or POST.

Table 7-135 Array structure

Parameter	How to Use	Example
type	Matching type. Options: • **Method**: HTTP requests are forwarded based on the matched method. This parameter must be used with **methodConfig**. This parameter can be configured only once. • **Header**: HTTP requests are forwarded based on the matched header. This parameter must be used with **headerConfig**. • **Cookie**: HTTP requests are forwarded based on the matched cookie. This parameter must be used with **cookieConfig**. • **QueryString**: HTTP requests are forwarded based on the matched character string. This parameter must be used with **queryStringConfig**. • **SourceIp**: HTTP requests are forwarded based on the matched CIDR block. This parameter must be used with **sourceIpConfig**. This parameter can be configured only once.	None

methodConfig	An HTTP request method that is used to forward requests. This parameter is used only when **type** is set to **Method**.
	Multiple request methods can be concurrently configured, including **GET**, **POST**, **PUT**, **DELETE**, **PATCH**, **HEAD**, and **OPTIONS**.
		You only need to configure values (an array) for **methodConfig**.
		{
		"type": "Method",
		"methodConfig": {
		"values": [
		"GET",
		"POST"
		]
		}
		}

Parameter	How to Use	Example

headerConfig	An HTTP request header that is used to forward requests. This parameter is used only when **type** is set to **Header**.
	• A key can only contain letters, digits, underscores (_), and hyphens (-).
	The first letter of the **User-agent** and **Connection** HTTP request headers must be capitalized.
	• Multiple values can be configured for a key. A value can only contain letters, digits, and special characters (!#$ %&'()*+,.\/:;<=>?@[]^-_'{|}~). Asterisks (*) and question marks (?) can be used as wildcard characters.
		You can only configure one key for each **headerConfig** rule. If you need multiple keys, configure multiple **headerConfig** rules.
		{
		"type": "Header",
		"headerConfig": {
		"key": "gray-hello",
		"values": [
		"value1",
		"value2"
		]
		}
		}

cookieConfig	A cookie that is used to forward requests. This parameter is used only when **type** is set to **Cookie**. A cookie consists of a key and a value, which must be configured separately.
	• A key can contain 1 to 100 characters and cannot start or end with a space.
	• A value can contain 1 to 100 characters. Configure a value for each key.
	You can enter multiple key-value pairs that can contain letters, digits, and special characters (! %'"()*+,./:=?@^-_`~).
		When configuring **cookieConfig**, you can configure multiple key-value pairs in **values**, and these pairs will have an OR relationship with each other.
		{
		"type": "Cookie",
		"cookieConfig": {
		"values": [
		{
		"key": "querystringkey1",
		"value": "querystringvalue2"
		},
		{
		"key": "querystringkey3",
		"value": "querystringvalue4"
		}
		]
		}
		}

Parameter	How to Use	Example

queryStringConfig	A character string that is used to forward requests. If the character string in a request matches the one in the configured forwarding rule, the request will be forwarded. This parameter is used only when **type** is set to **QueryString**.
	A query string consists of a key and one or more values. Configure the key and values separately.
	● A key can only contain letters, digits, and special characters (!$'()*+,./:;=?@^-_').
	● Multiple values can be configured for a key. A value can only contain letters, digits, and special characters (!$'()*+,./:;=?@^-_'). Asterisks (*) and question marks (?) can be used as wildcard characters.
		You can only configure one key for each **queryStringConfig** rule. If you need multiple keys, configure multiple **QueryStringConfig** rules.
		{
		"type": "QueryString",
		"queryStringConfig": {
		"key": "testKey",
		"values": [
		"testValue"
		]
		}
		}

sourceIpConfig	A CIDR block that is used to forward requests. This parameter is used only when **type** is set to **SourceIp**.
	Example CIDR block: 192.168.1.0/24 or 2020:50::44/127
		You only need to configure values (an array) for **sourceIpConfig**.
		{
		"type": "SourceIp",
		"sourceIpConfig": {
		"values": [
		"192.168.0.0/16",
		"172.16.0.0/16"
		]
		}
		}

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

----End

7.4.3.4.23 Configuring Advanced Forwarding Actions for a LoadBalancer Ingress

Dedicated load balancers offer different forwarding actions to effectively distribute traffic. ELB directs client requests to backend servers based on the specified forwarding rules.

Figure 7-80 How an advanced forwarding action operates

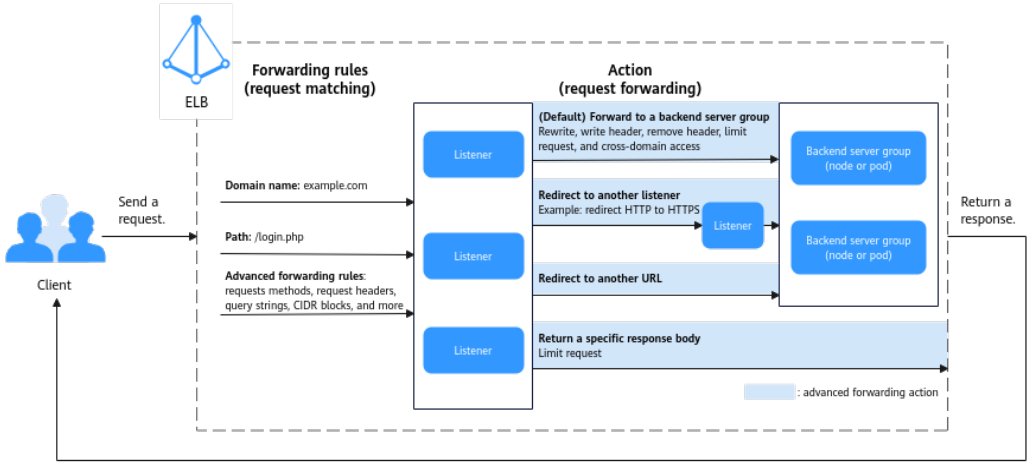


Table 7-136 Advanced forwarding actions

Action	Additional Action	Description	Reference	Cluster Version
Forward to a backend server group	None	Default forwarding action of a LoadBalancer ingress. Requests are directly forwarded to backend servers (backend nodes or pods in a cluster) without being processed.	Creating a LoadBalancer Ingress on the Console	No requirements on cluster versions
	Rewrite	Rewrite the request paths.	Configuring URL Rewriting for a LoadBalancer Ingress	• v1.23: v1.23.14-r0 or later • v1.25: v1.25.9-r0 or later • v1.27: v1.27.6-r0 or later • v1.28: v1.28.4-r0 or later • Other clusters of later versions

Action	Additional Action	Description	Reference	Cluster Version
	Cross-origin access	Support cross-origin access of resources.	**Configuring Cross-Origin Access for LoadBalancer Ingresses**	• v1.23: v1.23.18-r10 or later • v1.25: v1.25.16-r0 or later • v1.27: v1.27.16-r0 or later • v1.28: v1.28.13-r0 or later • v1.29: v1.29.8-r0 or later • v1.30: v1.30.4-r0 or later • Other clusters of later versions
	Write/ Remove header	Write or delete the configured header in a request before accessing the backend server.	**Configuring Header Writing or Deletion for a LoadBalancer Ingress**	
	Limit traffic on requests	Support traffic limit on requests.	**Configuring Request Limiting for a LoadBalancer Ingress**	• v1.27: v1.27.16-r10 or later • v1.28: v1.28.15-r0 or later • v1.29: v1.29.10-r0 or later • v1.30: v1.30.6-r0 or later • Other clusters of later versions
Redirect to another listener (redirect HTTP to HTTPS)	None	Forward HTTP access requests to HTTPS listeners.	**Redirecting HTTP to HTTPS for a LoadBalancer Ingress**	• v1.23: v1.23.14-r0 or later • v1.25: v1.25.9-r0 or later • v1.27: v1.27.6-r0 or later • v1.28: v1.28.4-r0 or later • Other clusters of later versions

Action	Additional Action	Description	Reference	Cluster Version
Redirect to another URL	None	Redirect a specific access request to a specified URL and return a 3xx return code.	Configuring URL Redirection for a LoadBalancer Ingress	
Return a specific response body	None	Directly return a fixed response and do not continue to forward the response to the backend server.	Returning a Specific Response Body for a LoadBalancer Ingress	• v1.25: v1.25.16-r10 or later • v1.27: v1.27.16-r10 or later • v1.28: v1.28.15-r0 or later • v1.29: v1.29.10-r0 or later • v1.30: v1.30.6-r0 or later • Other clusters of later versions
	Limit traffic on requests	Support traffic limit on requests.	Configuring Request Limiting for a LoadBalancer Ingress	• v1.27: v1.27.16-r10 or later • v1.28: v1.28.15-r0 or later • v1.29: v1.29.10-r0 or later • v1.30: v1.30.6-r0 or later • Other clusters of later versions

NOTE

CCE allows you to configure rewrite, header writing/deletion, and traffic limit on requests based on ELB's **advanced forwarding** for LoadBalancer ingresses. These functions are coming soon. To use some advanced forwarding actions that are not available on the console yet, submit a **service ticket** to enable required functions.

Prerequisites

- A CCE standard or Turbo cluster is available, and the cluster version meets the requirements.
- The cluster can be accessed using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- This feature is only available when dedicated load balancers are used.
- This feature relies on the ELB advanced forwarding policies. Once the ELB advanced forwarding policies are enabled, the forwarding policy priority is not determined by the domain name or path matching. You can **customize the forwarding policy priority** as needed. For details about the forwarding policy priority, see **Forwarding Policy Priorities of LoadBalancer Ingresses**.

Configuring Header Writing or Deletion for a LoadBalancer Ingress

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 034baaf0-40e8-4e39-b0d9-bf6e5b883cf9
    kubernetes.io/elb.port: "80"
    # Configure header writing or deletion for the Service named test-service.
    kubernetes.io/elb.actions.test-service: |
      [{
        "type": "InsertHeader",
        "InsertHeaderConfig": {
          "key": "aa",
          "value_type": "USER_DEFINED",
          "value": "aa"
        }
      },
      {
        "type": "InsertHeader",
        "InsertHeaderConfig": {
          "key": "bb",
          "value_type": "SYSTEM_DEFINED",
          "value": "ELB-ID"
        }
      },
      {
        "type": "InsertHeader",
        "InsertHeaderConfig": {
          "key": "cc",
          "value_type": "REFERENCE_HEADER",
          "value": "cc"
        }
      },
      {
        "type": "RemoveHeader",
        "RemoveHeaderConfig": {
          "key": "dd"
        }
      },
      {
        "type": "RemoveHeader",
        "RemoveHeaderConfig": {
          "key": "ee"
        }
      }]
  name: test
```



```
  namespace: default
spec:
  ingressClassName: cce
  rules:
    - http:
        paths:
          - backend:
              service:
                name: test-service
                port:
                  number: 8888
            path: /
            pathType: ImplementationSpecific
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-137 Annotations for advanced forwarding actions

Parameter	Type	Description
kubernetes.io/elb.actions.*${svc_name}*	String	Configure an advanced forwarding action for an ingress. *${svc_name}* indicates the Service name, which can contain a maximum of 51 characters. If the annotation value is set to *[]*, all advanced forwarding actions will be deleted. The value of the annotation for configuring header writing or deletion is a JSON string array. For details, see **Table 7-138**. For example: `[{"type":"InsertHeader","InsertHeaderConfig":{"key":"aa","value_type":"USER_DEFINED","value":"aa"}}]`

Table 7-138 Parameters for rewriting a header

Parameter	Description	Example
type	Type of header rewriting. Options: • **InsertHeader**: indicates that the header will be written. This field must be used with **InsertHeaderConfig**. • **RemoveHeader**: indicates that the header will be deleted. This field must be used with **RemoveHeaderConfig**. **NOTE** For advanced forwarding actions, you can add a maximum of five header writing or deletion configurations to an annotation.	None

Parameter	Description	Example

InsertHeaderConfig	Indicates that the header will be written. This parameter is used only when **type** is set to **InsertHeader**.
	• **key**: key of the rewritten header. A key consists of 1 to 40 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed. A key cannot contain any of the following characters (case-insensitive): connection, upgrade, content-length, transfer-encoding, keep-alive, te, host, cookie, remoteip, authority, x-forwarded-host, x-forwarded-for, x-forwarded-for-port, x-forwarded-tls-certificate-id, x-forwarded-tls-protocol, x-forwarded-tls-cipher, x-forwarded-elb-ip, x-forwarded-port, x-forwarded-elb-id, x-forwarded-elb-vip, x-real-ip, x-forwarded-proto, x-nuwa-trace-ne-in, or x-nuwa-trace-ne-out
	• **value_type**: type of the header to be written. If the header is deleted, this parameter becomes invalid. Options:
	– **USER_DEFINED**: custom header
	– **REFERENCE_HEADER**: header referenced by a user
	– **SYSTEM_DEFINED**: header defined by the system
	• **value**: value of the header to be written. If the header is deleted, this parameter becomes invalid. If **value_type** is set to **SYSTEM_DEFINED**, the **value** can only be **CLIENT-PORT**, **CLIENT-IP**, **ELB-PROTOCOL**, **ELB-ID**, **ELB-PORT**, **ELB-EIP**, or **ELB-VIP**.
		{
		"type": "InsertHeader",
		"InsertHeaderConfig": {
		"key": "aa",
		"value_type": "USER_DEFINED",
		"value": "aa"
		}
		}

Parameter	Description	Example

RemoveHeaderConfig	Indicates that the header will be deleted. This parameter is used only when **type** is set to **RemoveHeader**.
	• **key**: key of the deleted header. A key consists of 1 to 40 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed. A key cannot contain any of the following characters (case-insensitive): connection, upgrade, content-length, transfer-encoding, keep-alive, te, host, cookie, remoteip, authority, x-forwarded-host, x-forwarded-for, x-forwarded-for-port, x-forwarded-tls-certificate-id, x-forwarded-tls-protocol, x-forwarded-tls-cipher, x-forwarded-elb-ip, x-forwarded-port, x-forwarded-elb-id, x-forwarded-elb-vip, x-real-ip, x-forwarded-proto, x-nuwa-trace-ne-in, or x-nuwa-trace-ne-out
		{
		"type": "RemoveHeader",
		"RemoveHeaderConfig": {
		"key": "ee"
		}
		}

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME         CLASS    HOSTS     ADDRESS         PORTS   AGE
ingress-test  cce     *         121.**.**.**     80      10s
```

----End

Returning a Specific Response Body for a LoadBalancer Ingress

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```



```
metadata:
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 034baaf0-40e8-4e39-b0d9-bf6e5b883cf9
    kubernetes.io/elb.port: "80"
    # Configure the capability of returning a fixed response body for the Service named test-service.
    kubernetes.io/elb.actions.test-service: |
    [
     {
       "type": "FixedResponse",
       "fixedResponseConfig": {
          "contentType": "text/plain",
          "statusCode": "503",
          "messageBody": "503 error text"
       }
     }
    ]
  name: test
  namespace: default
spec:
  ingressClassName: cce
  rules:
    - http:
        paths:
          - backend:
              service:
                name: test-service
                port:
                  number: 8888
            path: /
            pathType: ImplementationSpecific
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-139 Annotations for advanced forwarding actions

Parameter	Type	Description
kubernetes.io/elb.actions.*${svc_name}*	String	Configure an advanced forwarding action for an ingress. *${svc_name}* indicates the Service name, which can contain a maximum of 51 characters. If the annotation value is set to *[]*, all advanced forwarding actions will be deleted. The annotation value of a fixed response is a JSON string array. For details, see **Table 7-140**.

Table 7-140 Parameters for a fixed response

Parameter	Description	Example
type	The value is fixed at **FixedResponse**, indicating that a fixed response will be returned. **NOTE** For advanced forwarding actions, you can add a maximum of one fixed response configuration to an annotation.	None

fixedResponseConfig	• **contentType**: format of the returned content. The options are **text/plain**, **text/css**, **text/html**, **application/javascript**, and **application/json**.
	• **statusCode**: By default, 2xx, 4xx, and 5xx status codes are supported.
	• (Mandatory) **messageBody**: Enter 0 to 1024 characters.
		{
		"type": "FixedResponse",
		"fixedResponseConfig": {
		"contentType": "text/plain",
		"statusCode": "503",
		"messageBody": "503 error text"
		}
		}

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

----End

Configuring Request Limiting for a LoadBalancer Ingress

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

An example YAML file of an ingress created using an existing load balancer is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: 034baaf0-40e8-4e39-b0d9-bf6e5b883cf9
    kubernetes.io/elb.port: "80"
    # Configure ELB request limiting for the Service named test-service.
```



```
  kubernetes.io/elb.actions.test-service: |
  [
   {
     "type": "TrafficLimit",
     "trafficLimitConfig": {
        "QPS": 4,
        "perSourceIpQps": 2,
        "burst": 2
     }
   }
  ]
 name: test
 namespace: default
spec:
 ingressClassName: cce
 rules:
  - http:
      paths:
        - backend:
            service:
              name: test-service
              port:
                number: 8888
          path: /
          pathType: ImplementationSpecific
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Table 7-141 Annotations for advanced forwarding actions

Parameter	Type	Description
kubernetes.io/elb.actions.*${svc_name}*	String	Configure an advanced forwarding action for an ingress. *${svc_name}* indicates the Service name, which can contain a maximum of 51 characters. If the annotation value is set to *[]*, all advanced forwarding actions will be deleted. The value of an annotation for configuring traffic limit is a JSON array. For details, see **Table 7-142**.

Table 7-142 Parameters for configuring traffic limit

Parameter	Description	Example
type	The value is fixed at **TrafficLimit**, indicating that traffic limit is enabled for a load balancer. **NOTE** For advanced forwarding actions, you can add a maximum of one traffic limit configuration to an annotation.	None

Parameter	Description	Example

trafficLimitConfig	• **QPS**: the total rate limiting, which specifies the maximum number of QPS. The value ranges from 1 to 100000. If the number of requests reaches the specified value, new requests will be discarded and 503 Service Unavailable will be returned to the client.
	• (Optional) **perSourceIpQps**: request limiting based on the source IP address of a client. The value ranges from 1 to 100000.
	If both **QPS** and **perSourceIpQps** are configured, the latter value must be smaller than the former. If the number of requests reaches the specified value, new requests will be discarded and 503 Service Unavailable will be returned to the client.
	• (Optional) **burst**: the size of a burst buffer. The value ranges from 0 to 100000. The burst rate allows for exceeding the average rate temporarily to handle sudden bursts of requests. For instance, if the limit is set to 5 but the burst rate is 10, five more requests can be processed within 1 second. However, if the number of requests exceeds 10, only five requests are allowed within 1 second.
		{
		"type": "TrafficLimit",
		"trafficLimitConfig": {
		"QPS": "4",
		"perSourceIpQps": "2",
		"burst": "2"
		}
		}

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:


```
NAME          CLASS    HOSTS    ADDRESS         PORTS   AGE
ingress-test  cce      *        121.**.**.**    80      10s
```

----End

7.4.3.5 Forwarding Policy Priorities of LoadBalancer Ingresses

CCE sets up forwarding policies on the ELB console based on the rules specified in the ingress configurations when creating LoadBalancer ingresses.

To handle more complex traffic routing needs, CCE has integrated the **advanced forwarding policies** of ELB, which means it supports advanced features like URL redirection and rewriting. However, note that the sorting logic for the advanced forwarding policies is different from the logic used for common forwarding policies.

Specifically:

- If the ELB advanced forwarding policies are not enabled, the forwarding policies are sorted by domain name or path. For details, see **Default Sorting**.
- After the ELB advanced forwarding policies are enabled, requests are matched based on the priorities and then forwarded to clients. For details, see **Priority**.

Default Sorting

If the ELB advanced forwarding policies are not enabled, the default sorting rule of forwarding policies is as follows:

- Forwarding rule priorities are independent of each other regardless of domain names. When a request matches both a domain name-based rule and a URL-based rule, the domain name-based rule is matched first.
- URL-based forwarding rules are applied in the following order of priority: an exact match rule, a prefix match rule, and a regular expression match rule. For multiple matches of the same type, only the longest URL-based forwarding rule will be applied.

Table 7-143 Example of a forwarding policy with the default sorting

Forwarding Policy	Specified Value	Order
URL (exact match)	/test1/test2/test3	1
URL (prefix match)	/test1/test2	2
URL (prefix match)	/test1	3

Once a forwarding policy is created, the system automatically sorts it based on the default sorting rule. Examples are as follows:

- The access request to **www.example.com/test1/test2** matches both forwarding policies 2 and 3. If the match types are the same, the priority is determined by the length of the URL, with longer URLs having higher priority. In this case, the access request will be forwarded based on the forwarding policy 2.

- The access request to **www.example.com/test1/test2/test3** matches forwarding policies 1, 2, and 3. An exact match is preferred, so the access request will be forwarded based on the forwarding policy 1.

Priority

With ELB advanced forwarding policies enabled, each request is matched based on the forwarding policy priority (a smaller value indicates a higher priority). Once a forwarding policy is matched, the request is forwarded based on this forwarding policy.

The rules for configuring the priority of a forwarding policy are as follows:

- The existing forwarding policies will maintain their original priority sequence before an advanced forwarding policy is configured.
- Once an advanced forwarding policy is set up, any new forwarding policy added will have the lowest priority. The default forwarding policy always has the lowest priority and is not included in the sorting process. Additionally, you have the option to manually specify the priority of a new forwarding policy. For details, see **Configuring the Priorities of Forwarding Rules for LoadBalancer Ingresses**.

Table 7-144 Example of a forwarding policy configured with priorities

Forwarding Policy	Specified Value	Priority
URL (prefix match)	/test1	1
URL (exact match)	/test1	2

In this example, the access request to **www.example.com/test1** satisfies both forwarding policies 1 and 2, so the request will be forwarded based on the **forwarding policy 1**.

7.4.3.6 Configuring Multiple Ingresses to Use the Same External ELB Port

In a cluster, multiple ingresses can share a listener, allowing them to use the same port on a single load balancer. If two ingresses have different listener configurations, the listener configuration of the earlier ingress (known as the first route) will be used. For details about how to check the first route, see **How Can I Determine Which Ingress the Listener Settings Have Been Applied To?**

The following table lists listener parameters.

Listener	Annotation	Documentation
Configuring ELB Certificates	kubernetes.io/elb.tls-certificate-ids kubernetes.io/elb.tls-ciphers-policy	**Configuring an HTTPS Certificate for a LoadBalancer Ingress**

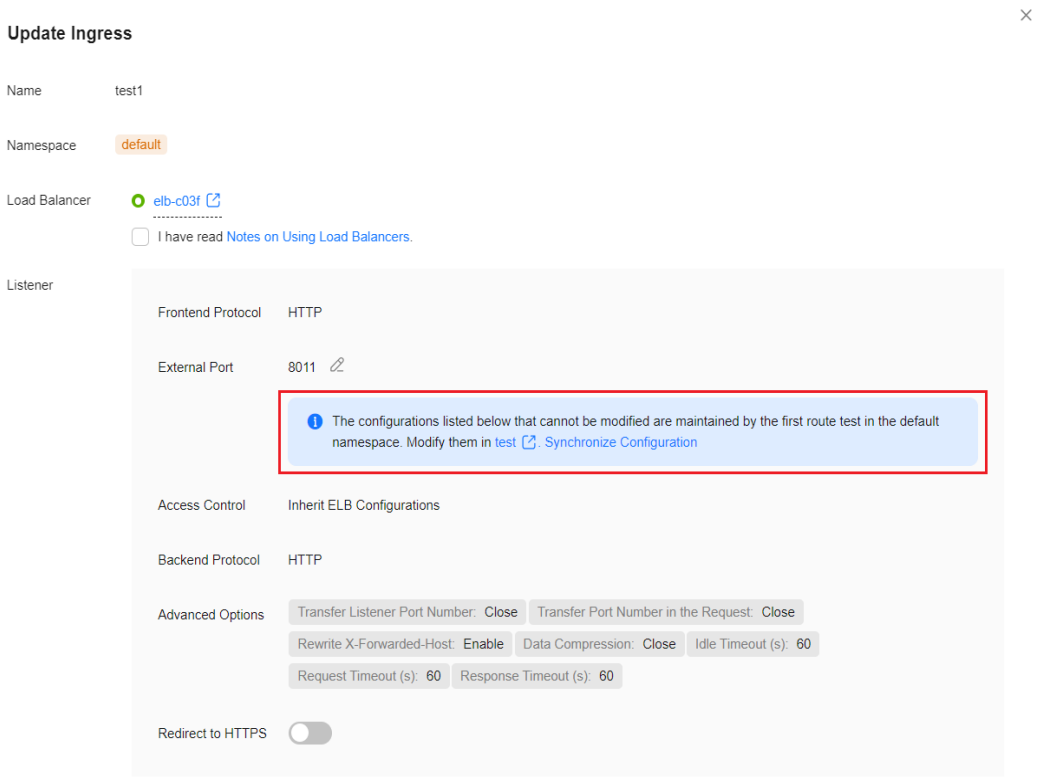
Listener	Annotation	Documentation
Using HTTP/2	kubernetes.io/elb.http2-enable	**Configuring HTTP/2 for a LoadBalancer Ingress**
Configuring Timeout for an Ingress	kubernetes.io/elb.keepalive_timeout kubernetes.io/elb.client_timeout kubernetes.io/elb.member_timeout	**Configuring Timeout for a LoadBalancer Ingress**
Configuring a Blocklist/Trustlist	kubernetes.io/elb.acl-id kubernetes.io/elb.acl-status kubernetes.io/elb.acl-type	**Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Ingress**
Configuring an HTTP/HTTPS Header	kubernetes.io/elb.x-forwarded-port kubernetes.io/elb.x-forwarded-for-port kubernetes.io/elb.x-forwarded-host	**Configuring an HTTP/HTTPS Header for a LoadBalancer Ingress**
Enabling GZIP	kubernetes.io/elb.gzip-enabled	**Configuring GZIP Data Compression for a LoadBalancer Ingress**

Ensure that the configurations of different listeners for various ingresses are synchronized. To do so, perform the following operations:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. Then, click the **Ingresses** tab, and choose **More** > **Update** in the **Operation** column.

Step 3 **Synchronize Configuration** is available if the listener configuration of the ingress differs from that of the ELB. Click **Synchronize Configuration**. Then, the listener configuration will be automatically synchronized.



----End

7.4.4 Nginx Ingresses

7.4.4.1 Creating an Nginx Ingress on the Console

In Kubernetes, an ingress is a resource object that controls how Services within a cluster can be accessed from outside the cluster. You can use ingresses to configure different forwarding rules to access pods in a cluster. The following uses an **Nginx workload** as an example to describe how to create an Nginx ingress on the console.

Prerequisites

- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see **ClusterIP** or **NodePort**.
- To add Nginx Ingress, ensure that the NGINX Ingress Controller add-on has been installed in the cluster. For details, see **Installing the Add-on**.

Notes and Constraints

- **It is not recommended modifying any configuration of a load balancer on the ELB console. Otherwise, the Service will be abnormal.** If you have modified the configuration, uninstall the nginx-ingress add-on and reinstall it.

- The URL registered in an ingress forwarding policy must be the same as the URL used to access the backend Service. Otherwise, a 404 error will be returned.
- The selected or created load balancer must be in the same VPC as the current cluster, and it must match the load balancer type (private or public network).
- The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.

Creating an Nginx Ingress

This section uses an Nginx workload as an example to describe how to create an Nginx ingress.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Services & Ingresses** in the navigation pane, click the **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

- **Name**: Customize the name of an ingress, for example, **nginx-ingress-demo**.
- **Namespace**: Select the namespace to which the ingress is to be added.
- **nginx-ingress**: This option is displayed only after the **NGINX Ingress Controller** add-on is installed in the cluster.
  - **External Protocol**: The options are **HTTP** and **HTTPS**. The default number of the listening port reserved when NGINX Ingress Controller is installed is 80 for HTTP and 443 for HTTPS. To use HTTPS, configure a certificate.
  - **Certificate Source**: source of a certificate for encrypting and authenticating HTTPS data transmission.
    - If you select a TLS key, you must create a key certificate of the IngressTLS or kubernetes.io/tls type beforehand. For details, see **Creating a Secret**.
    - If you select the default certificate, NGINX Ingress Controller will use its default certificate for encryption and authentication. You can configure the default certificate during **NGINX Ingress Controller** installation. If the default certificate is not configured, the certificate provided by NGINX Ingress Controller will be used.
  - **SNI**: stands for Server Name Indication (SNI), which is an extended protocol of TLS. SNI allows multiple TLS-compliant domain names for external access using the same IP address and port number, and different domain names can use different security certificates. After SNI is enabled, the client is allowed to submit the requested domain name when initiating a TLS handshake request. After receiving the TLS request, the load balancer searches for the certificate based on the domain name in the request. If the certificate corresponding to the domain name is found, the load balancer returns the certificate for authorization. Otherwise, the default certificate (server certificate) is returned for authorization.
- **Forwarding Policy**: When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL),

the request is forwarded to the corresponding target Service for processing. Click **Add Forwarding Policies** to add multiple forwarding policies.

- **Domain Name**: actual domain name. Ensure that the entered domain name has been registered and archived. After the ingress is created, bind the domain name to the IP address of the automatically created load balancer (IP address of the ingress access address). If a domain name rule is configured, the domain name must always be used for access.
- **Path Matching Rule**:
  - **Default**: Prefix match is used by default.
  - **Prefix match**: If the URL is set to **/healthz**, the URL that meets the prefix can be accessed, for example, **/healthz/v1** and **/healthz/v2**.
  - **Exact match**: The URL can be accessed only when it is fully matched. For example, if the URL is set to **/healthz**, only **/healthz** can be accessed.
- **Path**: access path, for example, **/healthz**

  NOTE

  - The access path matching rule of Nginx Ingress is based on the path prefix separated by the slash (/) and is case-sensitive. If the subpath separated by a slash (/) matches the prefix, the access is normal. However, if the prefix is only a part of the character string in the subpath, the access is not matched. For example, if the URL is set to /healthz, /healthz/v1 is matched, but /healthzv1 is not matched.
  - The access path added here must exist in the backend application. Otherwise, the forwarding fails.

    For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.
- **Destination Service**: Select an existing Service or create a Service. Any Services that do not match the search criteria will be filtered out automatically.
- **Destination Service Port**: Select the access port of the destination Service.
- **Operation**: Click **Delete** to delete the configuration.

- **Annotation**: The value is in the format of key-value pairs. You can use **annotations** to obtain the configurations supported by Nginx ingresses.

Step 4 Click **OK**.

After the ingress is created, it is displayed in the ingress list.

----End

7.4.4.2 Creating an Nginx Ingress Using kubectl

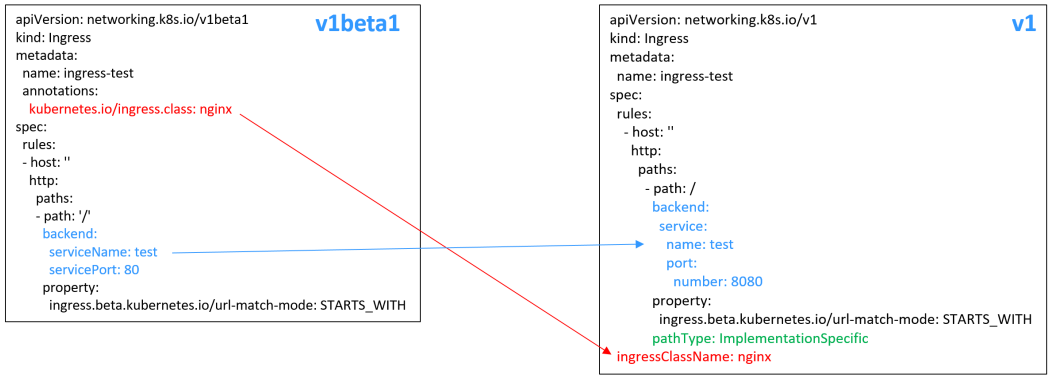
This section uses an **Nginx workload** as an example to describe how to create an Nginx ingress using kubectl.

Ingress API Version Upgrade in CCE Clusters v1.23

In CCE clusters of v1.23 or later, the ingress version is switched to **networking.k8s.io/v1**.

Compared with v1beta1, v1 has the following differences in parameters:

- The ingress type is specified by **spec.ingressClassName** instead of **kubernetes.io/ingress.class** in **annotations**.
- The format of **backend** has changed.
- The **pathType** parameter must be specified for each path. The options are as follows:
  - **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE, which is the same as v1beta1.
  - **Exact**: exact matching of the URL, which is case-sensitive.
  - **Prefix**: matching based on the URL prefix separated by a slash (/). The match is case-sensitive, and elements in the path are matched one by one. A path element refers to a list of labels in the path separated by a slash (/).



Prerequisites

- The NGINX Ingress Controller add-on has been installed in a cluster. For details, see **Installing the Add-on**.
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a workload by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**.
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see **ClusterIP** or **NodePort**.

Creating an Nginx Ingress

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```


> **NOTE**
>
> Starting from cluster v1.23, the ingress version is switched from **networking.k8s.io/v1beta1** to **networking.k8s.io/v1**. For details about the differences between v1 and v1beta1, see **Ingress API Version Upgrade in CCE Clusters v1.23**.

The following uses HTTP as an example to describe how to configure the YAML file:

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx  # Nginx Ingress is used. If multiple Nginx Ingress controllers are installed in the cluster, replace nginx with the custom name of the controller associated with the ingress.
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx  # Nginx Ingress is used.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

Table 7-145 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/ingress.class	Yes (only for clusters of v1.21 or earlier)	String	**nginx**: indicates that Nginx Ingress is used. This option is available only after the NGINX Ingress Controller add-on is installed. This parameter is mandatory when an ingress is created by calling the API.

Parameter	Mandatory	Type	Description
ingressClassName	Yes (only for clusters of v1.23 or later)	String	**nginx**: indicates that Nginx Ingress is used. This option is available only after the NGINX Ingress Controller add-on is installed. If multiple Nginx Ingress controllers are installed in the cluster, replace *nginx* with the custom **name of the controller** associated with the ingress. Multiple NGINX Ingress Controller add-ons can be installed in one cluster if the add-on version is 2.5.4 or later. In this case, the value of this parameter must be the **controller name** customized during controller installation, which indicates that the ingress is managed by the controller. This parameter is mandatory when an ingress is created by calling the API.
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.

Parameter	Mandatory	Type	Description
path	Yes	String	User-defined route path. All external access requests must match **host** and **path**. **NOTE** • The access path matching rule of Nginx Ingress is based on the path prefix separated by the slash (/) and is case-sensitive. If the subpath separated by a slash (/) matches the prefix, the access is normal. However, if the prefix is only a part of the character string in the subpath, the access is not matched. For example, if the URL is set to /healthz, /healthz/v1 is matched, but /healthzv1 is not matched. • The access path added here must exist in the backend application. Otherwise, the forwarding fails. For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.
ingress.beta.kubernetes.io/url-match-mode	No	String	Route matching policy. Default: **STARTS_WITH** (prefix match) Options: • **EQUAL_TO**: exact match • **STARTS_WITH**: prefix match

Parameter	Mandatory	Type	Description
pathType	Yes	String	Path type. This field is supported only by clusters of v1.23 or later. • **ImplementationSpecific**: The matching method depends on Ingress Controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE. • **Exact**: exact matching of the URL, which is case-sensitive. • **Prefix**: prefix matching, which is case-sensitive. With this method, the URL path is separated into multiple elements by slashes (/) and the elements are matched one by one. If each element in the URL matches the path, the subpaths of the URL can be routed normally. **NOTE** – During prefix matching, each element must be exactly matched. If the last element of the URL is the substring of the last element in the request path, no matching is performed. For example, **/foo/bar** matches **/foo/bar/baz** but does not match **/foo/barbaz**. – When elements are separated by slashes (/), if the URL or request path ends with a slash (/), the slash (/) at the end is ignored. For example, **/foo/bar** matches **/foo/bar/**. See **examples** of ingress path matching.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```


Step 5 Enter **http://*121.**.**.***:80** in the address box of the browser to access the workload (for example, **Nginx workload**).

121.**.**.** indicates the IP address of the unified load balancer.

----End

7.4.4.3 Annotations for Configuring Nginx Ingresses

The nginx-ingress add-on in CCE uses the community chart and image. If the default add-on parameters cannot meet your demands, you can add annotations to define what you need, such as the default backend, timeout, and size of a request body.

This section describes common annotations used for creating an ingress of the Nginx type.

NOTE

- The key value of an annotation can only be a string. Other types (such as Boolean values or numeric values) must be enclosed in quotation marks (""), for example, "true", "false", and "100".
- Nginx Ingress supports native annotations of the community. For details, see **Annotations**.

- **Ingress Type**
- **Configuring a Redirection Rule**
- **Configuring a URL Rewriting Rule**
- **Interconnecting with HTTPS Backend Services**
- **Creating a Consistent Hashing Rule for Load Balancing**
- **Customized Timeout Interval**
- **Customizing a Body Size**
- **Two-Way HTTPS Authentication**
- **Domain Name Regularization**
- **Grayscale Release**
- **Documentation**

Ingress Type

Table 7-146 Ingress type annotations

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/ingress.class	String	• **nginx**: Nginx Ingress is used. • **cce**: A proprietary LoadBalancer ingress is used. This parameter is mandatory when an ingress is created by calling the API. For clusters of v1.23 or later, use the parameter **ingressClassName**. For details, see **Creating an Nginx Ingress Using kubectl**.	Only clusters of v1.21 or earlier

For details about how to use the preceding annotations, see **Creating an Nginx Ingress Using kubectl**.

Configuring a Redirection Rule

Table 7-147 Redirection rule annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/permanent-redirect	String	Permanently redirects an access request to a target website (status code 301).
nginx.ingress.kubernetes.io/permanent-redirect-code	String	Changes the returned status code of a permanent redirection rule to a specified value.
nginx.ingress.kubernetes.io/temporal-redirect	String	Temporarily redirects an access request to a target website (status code 302).
nginx.ingress.kubernetes.io/ssl-redirect	String	Specifies whether an HTTP request can be redirected to HTTPS only through SSL. The default value is **true** when the ingress contains an SSL certificate.
nginx.ingress.kubernetes.io/force-ssl-redirect	String	Indicates whether to forcibly redirect a request to HTTPS even if TLS is not enabled for the ingress. When HTTP is used for access, the request is forcibly redirected (status code 308) to HTTPS.

For details, see **Configuring Redirection Rules for an Nginx Ingress**.

Configuring a URL Rewriting Rule

Table 7-148 URL rewriting rule annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/rewrite-target	String	Target URI where the traffic must be redirected.

For details, see **Configuring URL Rewriting Rules for an Nginx Ingress**.

Interconnecting with HTTPS Backend Services

Table 7-149 Annotations for interconnecting with HTTPS backend services

Parameter	Type	Description
nginx.ingress.kubernetes.io/backend-protocol	String	If this parameter is set to **HTTPS**, HTTPS is used to forward requests to the backend service container.

For details, see **Configuring HTTPS Backend Services for an Nginx Ingress**.

Creating a Consistent Hashing Rule for Load Balancing

Table 7-150 Annotation of consistent hashing for load balancing

Parameter	Type	Description
nginx.ingress.kubernetes.io/upstream-hash-by	String	Enable consistent hashing for load balancing for backend servers. The parameter value can be an Nginx parameter, a text value, or any combination. For example: • **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"** indicates that requests are hashed based on the request URI. • **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri$host"** indicates that requests are hashed based on the request URI and domain name. • **nginx.ingress.kubernetes.io/upstream-hash-by: "${request_uri}-text-value"** indicates that requests are hashed based on the request URI and text value.

For details, see **Configuring Consistent Hashing for Load Balancing of an Nginx Ingress**.

Customized Timeout Interval

Table 7-151 Customized timeout interval annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/proxy-connect-timeout	String	Customized connection timeout interval. You do not need to set the unit when setting the timeout interval. The default unit is second. Example: `nginx.ingress.kubernetes.io/proxy-connect-timeout: '120'`

Customizing a Body Size

Table 7-152 Annotations of customizing a body size

Parameter	Type	Description
nginx.ingress.kubernetes.io/proxy-body-size	String	When the body size in a request exceeds the upper limit, error 413 will be returned to the client. You can use this parameter to adjust the upper limit of the body size. The basic unit of the parameter value is byte. You can use units such as KB, MB, and GB. The unit conversion is as follows: 1 KB = 1024 bytes, 1 MB = 1024 KB, 1 GB =1024 MB Example: `nginx.ingress.kubernetes.io/proxy-body-size: 8m`

Two-Way HTTPS Authentication

Nginx Ingress supports two-way HTTPS authentication between the server and client to ensure secure connections.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following command to create a self-signed CA certificate:

```
openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Ingress Cert Authority'
```

Expected output:

```
Generating a RSA private key
.............++++
.....................................................++++
```



```
writing new private key to 'ca.key'
-----
```

Step 3 Create a server certificate.

1. Run the following command to create a request file for generating a server certificate:

   ```
   openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN=foo.bar.com'
   ```

   Expected output:

   ```
   Generating a RSA private key
   ..............................................................++++
   ..........++++
   writing new private key to 'server.key'
   -----
   ```

2. Run the following command to issue the server request file using the root certificate to generate the server certificate:

   ```
   openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt
   ```

   Expected output:

   ```
   Signature ok
   subject=CN = foo.bar.com
   Getting CA Private Key
   ```

Step 4 Create a client certificate.

1. Run the following command to create a request file for generating a client certificate:

   ```
   openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Ingress'
   ```

   Expected output:

   ```
   Generating a RSA private key
   ..................................++++
   ..............................................++++
   writing new private key to 'client.key'
   -----
   ```

2. Run the following command to issue the client request file using the root certificate to generate the client certificate:

   ```
   openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt
   ```

   Expected output:

   ```
   Signature ok
   subject=CN = Ingress
   Getting CA Private Key
   ```

Step 5 Run the **ls** command to check the created certificates.

Expected output:

```
ca.crt  ca.key  client.crt  client.csr  client.key  server.crt  server.csr  server.key
```

Step 6 Run the following command to create a secret of the CA certificate:

```
kubectl create secret generic ca-secret --from-file=ca.crt=ca.crt
```

Expected output:

```
secret/ca-secret created
```

Step 7 Run the following command to create a secret of the server certificate:

```
kubectl create secret generic tls-secret --from-file=tls.crt=server.crt --from-file=tls.key=server.key
```

Expected output:

```
secret/tls-secret created
```


Step 8 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

- **For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/auth-tls-secret: "default/ca-secret"   # Replace it with your CA certificate secret.
    nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
    nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "true"
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          service:
            name: nginx-test  # Replace it with the name of your target Service.
            port:
              number: 80  # Replace it with the port of your target Service.
        path: /
        pathType: ImplementationSpecific
  tls:
  - hosts:
    - foo.bar.com
    secretName: tls-secret   # Replace it with your TLS certificate secret.
  ingressClassName: nginx
```

- **For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/auth-tls-secret: "default/ca-secret"   # Replace it with your CA certificate secret.
    nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
    nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "true"
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: nginx-test  # Replace it with the name of your target Service.
          servicePort: 80  # Replace it with the port of your target Service.
  tls:
  - hosts:
    - foo.bar.com
    secretName: tls-secret   # Replace it with your TLS certificate secret.
```

Step 9 Run the following command to create an ingress:

```
kubectl create -f ingress-test.yaml
```

Expected output:

```
ingress.networking.k8s.io/ingress-test created
```

Step 10 Run the following command to obtain the IP address of the ingress:

```
kubectl get ingress
```


Expected output:

```
NAME        CLASS   HOSTS         ADDRESS      PORTS     AGE
nginx-test  nginx   foo.bar.com   10.3.xx.xx   80, 443   27m
```

Step 11 Run the following command to update the IP address of the ingress into the **hosts** file and replace the following IP address with the actual IP address of the ingress:

```
echo "10.3.xx.xx  foo.bar.com" | sudo tee -a /etc/hosts
```

Expected output:

```
10.3.xx.xx  foo.bar.com
```

Step 12 Verify the configuration.

- The client does not send the certificate for access.

  ```
  curl --cacert ./ca.crt  https://foo.bar.com
  ```

  Expected output:

  ```
  <html>
  <head><title>400 No required SSL certificate was sent</title></head>
  <body>
  <center><h1>400 Bad Request</h1></center>
  <center>No required SSL certificate was sent</center>
  <hr><center>nginx</center>
  </body>
  </html>
  ```

- The client sends the certificate for access.

  ```
  curl --cacert ./ca.crt --cert ./client.crt --key ./client.key https://foo.bar.com
  ```

  Expected output:

  ```
  <!DOCTYPE html>
  <html>
  <head>
  <title>Welcome to nginx!</title>
  <style>
  body {
      width: 35em;
      margin: 0 auto;
      font-family: Tahoma, Verdana, Arial, sans-serif;
  }
  </style>
  </head>
  <body>
  <h1>Welcome to nginx!</h1>
  <p>If you see this page, the nginx web server is successfully installed and
  working. Further configuration is required.</p>

  <p>For online documentation and support please refer to
  <a href="http://nginx.org/">nginx.org</a>.<br/>
  Commercial support is available at
  <a href="http://nginx.com/">nginx.com</a>.</p>

  <p><em>Thank you for using nginx.</em></p>
  </body>
  </html>
  ```

----End

Domain Name Regularization

Nginx Ingress allows you to configure the **nginx.ingress.kubernetes.io/server-alias** annotation to configure regular expressions for domain names.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For example, the regular expression **~^www\.\d+\.example\.com $,abc.example.com** indicates that you can access the ingress using **www.***{One or more digits}*.**example.com** and **abc.example.com**.

- **For clusters of v1.23 or later:**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/server-alias: '~^www\.\d+\.example\.com$,abc.example.com'
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          service:
            name: nginx-93244  # Replace it with the name of your target Service.
            port:
              number: 80  # Replace it with the port of your target Service.
        path: /
        pathType: ImplementationSpecific
  ingressClassName: nginx
```

- **For clusters of v1.21 or earlier:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ server-alias: '~^www\.\d+\.example\.com$,abc.example.com'
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: nginx-test  # Replace it with the name of your target Service.
          servicePort: 80  # Replace it with the port of your target Service.
```

Step 3 Run the following command to create an ingress:

```
kubectl create -f ingress-test.yaml
```

Expected output:

```
ingress.networking.k8s.io/ingress-test created
```

Step 4 Check the NGINX Ingress Controller configuration.

1. Run the following command to check the NGINX Ingress Controller pods:

```
kubectl get pods -n kube-system | grep nginx-ingress-controller
```

Expected output:

```
cceaddon-nginx-ingress-controller-68d7bcc67-dxxxx       1/1     Running   0          18h
cceaddon-nginx-ingress-controller-68d7bcc67-cxxxx       1/1     Running   0          18h
```

2. Run the following command to check the NGINX Ingress Controller configuration:

```
kubectl exec -n kube-system cceaddon-nginx-ingress-controller-68d7bcc67-dxxxx cat /etc/nginx/nginx.conf | grep -C3 "foo.bar.com"
```


Expected output:

```
        ## start server foo.bar.com
        server {
                server_name foo.bar.com abc.example.com ~^www\.\d+\.example\.com$ ;

                listen 80  ;
                listen [::]:80  ;
--
                }

        }
        ## end server foo.bar.com
```

Step 5 Run the following command to obtain the IP address of the ingress:

```
kubectl get ingress
```

Expected output:

```
NAME         CLASS   HOSTS         ADDRESS      PORTS   AGE
nginx-test   nginx   foo.bar.com   10.3.xx.xx   80      14m
```

Step 6 Use different rules to test service access.

- Run the following command to access the service through **Host: foo.bar.com**:
  ```
  curl -H "Host: foo.bar.com" 10.3.xx.xx/
  ```
  It is expected that the web page can be accessed properly.
- Run the following command to access the service through **Host: www.123.example.com**:
  ```
  curl -H "Host: www.123.example.com" 10.3.xx.xx/
  ```
  It is expected that the web page can be accessed properly.
- Run the following command to access the service through **Host: www.321.example.com**:
  ```
  curl -H "Host: www.321.example.com" 10.3.xx.xx/
  ```
  It is expected that the web page can be accessed properly.

----End

Grayscale Release

Grayscale release can be enabled using annotation **nginx.ingress.kubernetes.io/canary: "true"** and implemented using different annotation settings.

Table 7-153 Grayscale release annotations

Parameter	Type	Description
nginx.ingress.kubernetes.io/canary-weight	String	Percentage of the requests sent to a specified service. The value is an integer ranging from 0 to 100.

Parameter	Type	Description
nginx.ingress.kubernetes.io/canary-by-header	String	Traffic is split based on request headers. If a request header contains a specified header name and the value is **always**, the request will be forwarded to the backend service specified by the canary ingress. If the value is **never**, the request will not be forwarded and can be used for a rollback. If other values are used, the annotation will be ignored and the request traffic will be allocated according to other rules based on the priority.
nginx.ingress.kubernetes.io/canary-by-header-value	String	This annotation must be used with **canary-by-header**. You can customize the value of a request header, including but not limited to **always** or **never**. If the value of a request header matches the specified custom value, the request will be forwarded to the backend service defined by the canary ingress. If the values do not match, the annotation will be ignored and the request traffic will be allocated according to other rules based on the priority.
nginx.ingress.kubernetes.io/canary-by-header-pattern	String	This annotation is similar to **canary-by-header-value**. The only difference is that this annotation uses a regular expression, not a fixed value, to match the value of a request header. If both this annotation and **canary-by-header-value** are configured, this one will be ignored.
nginx.ingress.kubernetes.io/canary-by-cookie	String	Traffic is split based on cookies. If a cookie value is **always**, the request traffic will be allocated for grayscale release. If a cookie value is **never**, the request traffic will not be allocated for grayscale release.

NOTE

- The preceding annotation rules are evaluated based on the following priority: **canary-by-header** -> **canary-by-cookie** -> **canary-weight**.
- When an ingress is marked as a canary ingress, all non-canary annotations except **nginx.ingress.kubernetes.io/load-balance** and **nginx.ingress.kubernetes.io/upstream-hash-by** will be ignored.

For more information, see **Using Nginx Ingress to Implement Grayscale Release and Blue-Green Deployment**.

The configuration examples of some annotations are as follows:

- Weight-based grayscale release: Set the weight of the grayscale release service to **20%**.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
```



```
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "20"
...
```

- Header-based grayscale release: If a request header value is **cce:always**, the request will be routed to the grayscale service. If a request header value is **cce:never**, the request will not be routed to the grayscale service. For other request header values, the request will be allocated to the grayscale service based on the grayscale weight.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "50"
    nginx.ingress.kubernetes.io/canary-by-header: "cce"
...
```

- Header-based grayscale release (custom header value): If a request header value is **cce:test**, the request will be routed to the grayscale service. For other request header values, the request will be allocated to the grayscale service based on the grayscale weight.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "20"
    nginx.ingress.kubernetes.io/canary-by-header: "cce"
    nginx.ingress.kubernetes.io/canary-by-header-value: "test"
...
```

- Cookie-based grayscale release: When a header is not matched and the requested cookie is **region=always**, the request will be routed to the grayscale service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "20"
    nginx.ingress.kubernetes.io/canary-by-header: "cce"
    nginx.ingress.kubernetes.io/canary-by-header-value: "test"
    nginx.ingress.kubernetes.io/canary-by-cookie: "region"
...
```

Documentation

For details about annotation parameters supported by Nginx Ingress, see **Annotations**.

7.4.4.4 Advanced Setting Examples of Nginx Ingresses

7.4.4.4.1 Configuring an HTTPS Certificate for an Nginx Ingress

HTTPS certificates can be configured for ingresses to provide security services.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Ingress supports two TLS secret types: kubernetes.io/tls and IngressTLS. IngressTLS is used as an example. For details, see **Creating a Secret**. For details about examples of the kubernetes.io/tls secret and its description, see **TLS secrets**.

Create a YAML file named **ingress-test-secret.yaml**. The file name can be customized.

```
vi ingress-test-secret.yaml
```

The YAML file is configured as follows:

```
apiVersion: v1
data:
  tls.crt: LS0******tLS0tCg==
  tls.key: LS0tL******0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
  name: ingress-test-secret
  namespace: default
type: IngressTLS
```

NOTE

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files. The values of **tls.crt** and **tls.key** are Base64-encoded.

Step 3 Create a secret.

```
kubectl create -f ingress-test-secret.yaml
```

If information similar to the following is displayed, the secret has been created:

```
secret/ingress-test-secret created
```

Check the created secret.

```
kubectl get secrets
```

If information similar to the following is displayed, the secret has been created:

```
NAME                  TYPE                DATA   AGE
ingress-test-secret   IngressTLS          2      13s
```

Step 4 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: ingress-test-secret  # Replace it with your TLS key certificate.
  rules:
    - host: foo.bar.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
```



```
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: ingress-test-secret   # Replace it with your TLS key certificate.
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: '/'
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: <your_service_port>  # Replace it with the port number of your target Service.
  ingressClassName: nginx
```

Step 5 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 6 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

Step 7 Enter **https://*121.**.**.***:443** in the address box of the browser to access the workload (for example, **Nginx workload**).

121.**.**.** indicates the IP address of the unified load balancer.

----End

7.4.4.4.2 Configuring Redirection Rules for an Nginx Ingress

Configuring a Permanent Redirection Rule

To permanently redirect an access request to a target website (status code 301), use the **nginx.ingress.kubernetes.io/permanent-redirect** annotation. For example, to permanently redirect all access requests to **www.example.com**, run the following command:

```
nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
```

The configuration in an Nginx Ingress is as follows:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

Step 5 Access the ingress. *${ELB_IP}* specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I ${ELB_IP}
```

The access path will be redirected to **www.example.com**.


```
HTTP/1.1 301 Moved Permanently
Date: Sat, 20 Jul 2024 07:55:49 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://www.example.com
```

----End

Configuring the Returned Status Code for the Permanent Redirection Rule

When configuring a permanent redirection rule, you can use the **nginx.ingress.kubernetes.io/permanent-redirect-code** annotation to modify its returned status code. For example, to set the status code for the permanent redirection to 308, run the following command:

```
nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
```

The configuration in an Nginx Ingress is as follows:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
    nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.example.com
    nginx.ingress.kubernetes.io/permanent-redirect-code: '308'
spec:
  rules:
    - host: ''
      http:
        paths:
```



```
      - path: /
        backend:
          serviceName: <your_service_name>  # Replace it with the name of your target Service.
          servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

Step 5 Access the ingress. *${ELB_IP}* specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I ${ELB_IP}
```

The access path will be redirected to **www.example.com**, and the status code is 308.

```
HTTP/1.1 308 Moved Permanently
Date: Sat, 20 Jul 2024 07:55:49 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://www.example.com
```

----End

Configuring a Temporary Redirection Rule

To temporarily redirect an access request to a target website (status code 302), use the **nginx.ingress.kubernetes.io/temporal-redirect** annotation. For example, to temporarily redirect all access requests to **www.example.com**, run the following command:

```
nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
```

The configuration in an Nginx Ingress is as follows:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
spec:
  rules:
    - host: ''
```



```
    http:
      paths:
        - path: /
          backend:
            service:
              name: <your_service_name>  # Replace it with the name of your target Service.
              port:
                number: <your_service_port>  # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/temporal-redirect: https://www.example.com
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

Step 5 Access the ingress. *${ELB_IP}* specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I ${ELB_IP}
```

The access path will be redirected to **www.example.com**, and the status code is 302.

```
HTTP/1.1 302 Moved Temporarily
Date: Sat, 20 Jul 2024 08:01:02 GMT
Content-Type: text/html
Content-Length: 138
Connection: keep-alive
Location: https://www.example.com
```

----End

Redirecting HTTP to HTTPS

By default, if an ingress uses TLS, requests will be redirected (status code 308) to HTTPS when HTTP is used for access. You can configure the redirection by using the **nginx.ingress.kubernetes.io/ssl-redirect** annotation.

- If the value of this annotation is set to **true**, an HTTP access is redirected to HTTPS (status code 308) when the TLS certificate is used.
- If the value of this annotation is set to **false**, an HTTP access cannot be redirected to HTTPS when the TLS certificate is used.

If you need to forcibly redirect an HTTP access to HTTPS without a TLS, you can configure the redirection by setting the value of **nginx.ingress.kubernetes.io/force-ssl-redirect** to **true**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: /
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```


Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

Step 5 Access the ingress. *${ELB_IP}* specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl -I ${ELB_IP}
```

The access path will be redirected to HTTPS, and the status code is 308.

```
HTTP/1.1 308 Permanent Redirect
Date: Sat, 20 Jul 2024 08:03:36 GMT
Content-Type: text/html
Content-Length: 164
Connection: keep-alive
Location: https://${ELB_IP}
```

----End

7.4.4.4.3 Configuring URL Rewriting Rules for an Nginx Ingress

In some application scenarios, the access URL provided by the backend service is different from the path specified in the ingress rule. The ingress directly forwards the access path to the same backend path. If URL rewriting is not configured, 404 is returned for all access requests. For example, if the access path in the ingress rule is set to **/app/demo** and the access path provided by the backend service is **/demo**, access requests are directly forwarded to the **/app/demo** path of the backend service, which does not match the actual access path (**/demo**) provided by the backend service. As a result, 404 is returned.

In this case, you can use the Rewrite method to implement URL rewriting. That is, you can use the **nginx.ingress.kubernetes.io/rewrite-target** annotation to implement rewriting rules for different paths.

Configuring a Rewriting Rule

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
```



```
rules:
  - host: ''
    http:
      paths:
        - path: '/something(/|$)(.*)'
          backend:
            service:
              name: <your_service_name>  # Replace it with the name of your target Service.
              port:
                number: <your_service_port>  # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/something(/|$)(.*)'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

NOTE

As long as **rewrite-target** is specified for one ingress, all paths under the same host in all ingress definitions are case-sensitive, including the ingresses that do not have **rewrite-target** specified.

In the preceding example, the placeholder **$2** specifies that all characters matched by the second parenthesis (.*) are added to the **nginx.ingress.kubernetes.io/rewrite-target** annotation as the rewritten URL path.

For example, the preceding regular expression matching for ingress causes URL rewriting in multiple cases. These cases include:

- The **/something** path is rewritten to the **/** path.
- The **/something/** path is rewritten to the **/** path.
- The **/something/new** path is rewritten to the **/new** path.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```


Step 5 Access the ingress. *${ELB_IP}* specifies the IP address of the load balancer associated with the Nginx ingress.

```
curl ${ELB_IP}/something
```

The access path will be redirected to **/**.

----End

Checking the Configurations of an Ingress

In the **nginx-ingress-controller** container, you can check all configurations of an ingress in the **nginx.conf** file in the **/etc/nginx** directory.

Step 1 Obtain the name of the pod that runs **nginx-ingress-controller**.

```
kubectl get pod -n kube-system | grep nginx-ingress-controller
```

The command output is as follows:

```
cceaddon-nginx-ingress-controller-66855ccb9b-52qcd       1/1     Running   0          37m
```

Step 2 Log in to the **nginx-ingress-controller** container.

```
kubectl exec -it cceaddon-nginx-ingress-controller-66855ccb9b-52qcd -- /bin/bash
```

Step 3 Check the **nginx.conf** file in **/etc/nginx**.

```
cat /etc/nginx/nginx.conf
```

The rewriting rule in this example generates a Rewrite command and writes it to the **location** field in the **nginx.conf** file as follows:

```
## start server _
        server {
                server_name _ ;
                ...
                location ~* "^/something(/|$)(.*)" {
                        set $namespace      "default";
                        set $ingress_name   "ingress-test";
                        set $service_name   "<your_service_name>";
                        set $service_port   "80";
                        ...
                        rewrite "(?i)/something(/|$)(.*)" /$2 break;
                        ...
                }
        }
        ## end server _
```

The basic syntax of the Rewrite command is as follows:

```
rewrite regex replacement [flag];
```

- **regex**: regular expression for matching URIs. In the preceding example, **(?i)/something(/|$)(.*)** is the regular expression for matching URIs, where **(?i)** indicates case-insensitive.
- **replacement**: content to rewrite. In the preceding example, **/$2** indicates that the path is rewritten to all the characters matched by the second parenthesis **(.*)**.
- **flag**: rewrite format.
  - **last**: continues to match the next rule after the current rule is matched.
  - **break**: stops matching after the current rule is matched.
  - **redirect**: returns a temporary redirect with the 302 code.
  - **permanent**: returns a permanent redirect with the 301 code.

----End

Advanced Rewrite Configuration

Some complex, advanced Rewrite requirements can be implemented by modifying the Nginx configuration file **nginx.conf**. However, the **nginx.ingress.kubernetes.io/rewrite-target** annotation function can be customized to meet more complex Rewrite requirements.

- nginx.ingress.kubernetes.io/server-snippet: Add custom settings to the **server** field in the **nginx.conf** file.
- nginx.ingress.kubernetes.io/configuration-snippet: Add custom settings to the **location** field in the **nginx.conf** file.

NOTE

Snippet is not enabled by default when the NGINX Ingress Controller version is 2.4.6 or later (corresponding to the community version v1.9.3). For details, see **Changelog**. If snippet is needed, enable it using **allow-snippet-annotations**.

You can use the preceding two annotations to insert a Rewrite command into the **server** or **location** field in the **nginx.conf** file to rewrite the URL. The following is an example:

```
annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/configuration-snippet: |
      rewrite ^/stylesheets/(.*)$ /something/stylesheets/$1 redirect;  # Add the /something prefix.
      rewrite ^/images/(.*)$ /something/images/$1 redirect;  # Add the /something prefix.
```

In the preceding rules, the **/something** prefix is added to the access URL. Specifically:

- When a user accesses the **/stylesheets/new.css** path, the path is rewritten as the **/something/stylesheets/new.css** path.
- When a user accesses the **/images/new.jpg** path, the path is rewritten as the **/something/images/new.jpg** path.

7.4.4.4.4 Configuring HTTPS Backend Services for an Nginx Ingress

Ingress can function as a proxy for backend services using different protocols. By default, the backend proxy channel of an ingress is an HTTP channel. To create an HTTPS channel, add the following configuration to the **annotations** field:

```
nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
```

An ingress configuration example is as follows:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
```



```
tls:
  - secretName: ingress-test-secret  # Replace it with your TLS key certificate.
rules:
  - host: ''
    http:
      paths:
        - path: '/'
          backend:
            service:
              name: <your_service_name>  # Replace it with the name of your target Service.
              port:
                number: <your_service_port>  # Replace it with the port number of your target Service.
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
    - secretName: ingress-test-secret  # Replace it with your TLS key certificate.
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              serviceName: <your_service_name>  # Replace it with the name of your target Service.
              servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

----End

7.4.4.4.5 Configuring gRPC Backend Services for an Nginx Ingress

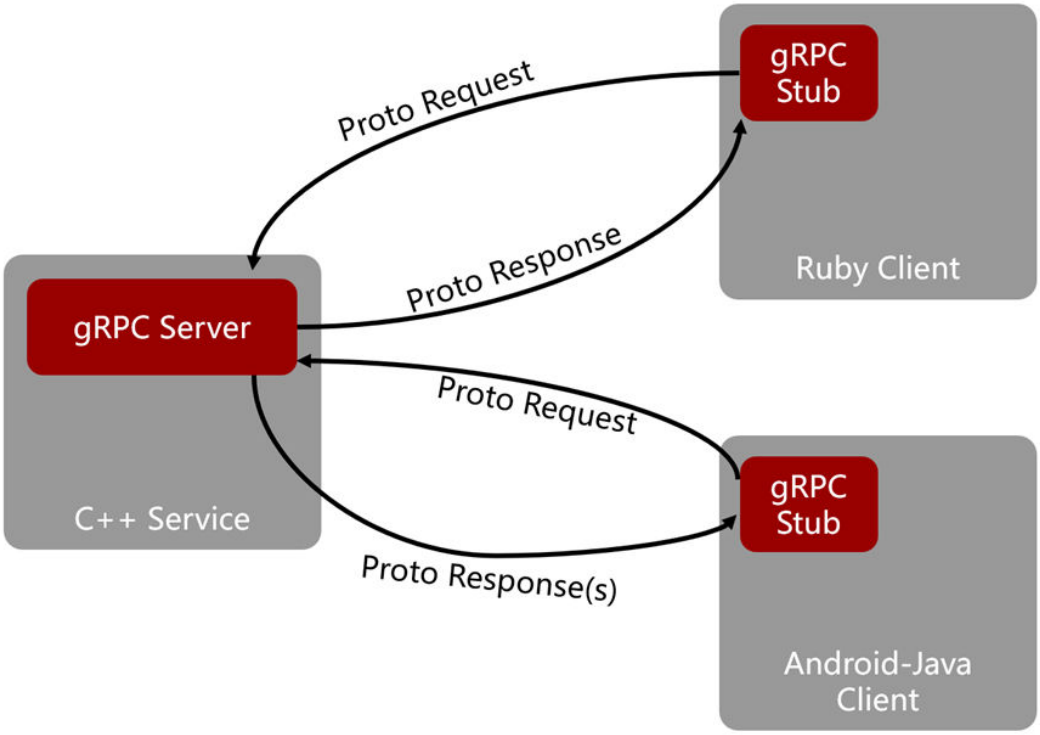
This section describes how to route traffic to gRPC backend services through an Nginx ingress.

Introduction to gRPC

gRPC is a high-performance, open-source universal RPC framework. It uses Protocol Buffers (Protobuf) as its Interface Definition Language (IDL) and underlying message interchange format. In addition, gRPC leverages HTTP/2 as its

underlying transport protocol, which provides features like multiplexing, header compression, and traffic control. This greatly improves the communication efficiency between the client and server. For more information, see **Introduction to gRPC**.

Figure 7-81 gRPC diagram



With gRPC, you can create distributed applications and services more easily by allowing a client application to call a method on a server application on a different machine as if it were a local object. gRPC, similar to other RPC systems, involves defining a service that specifies the methods that can be remotely called, along with their parameters and return types. The server then implements this interface and operates a gRPC server to handle client requests.

Preparations

- A CCE standard cluster is available. For details, see **Buying a CCE Standard/ Turbo Cluster**.
- The **NGINX Ingress Controller** add-on has been installed in the cluster.
- kubectl has been installed and configured. For details, see **Connecting to a Cluster Using kubectl**.
- gRPCurl has been installed. For details, see **gRPCurl**.

Example of the gRPC Service

Define the gRPC service in a proto file, where the client can call the SayHello interface of the helloworld.Greeter service. For details about the source code of the sample gRPC service, see **gRPC Hello World**.

```
syntax = "proto3";
```



```
option go_package = "google.golang.org/grpc/examples/helloworld/helloworld";
option java_multiple_files = true;
option java_package = "io.grpc.examples.helloworld";
option java_outer_classname = "HelloWorldProto";

package helloworld;

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

When using an Nginx ingress, gRPC runs only on the HTTPS port (443 by default). Therefore, in the production environment, you need a domain name and its corresponding SSL certificate. In this example, grpc.example.com and a self-signed SSL certificate are used.

Step 1: Creating an SSL Certificate

Step 1 Copy the following content and save it to the **openssl.cnf** file:

```
[req]
distinguished_name = req_distinguished_name
attributes = req_attributes

[req_distinguished_name]

[req_attributes]

[test_ca]
basicConstraints        = critical,CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer:always
keyUsage                = critical,keyCertSign

[test_server]
basicConstraints        = critical,CA:FALSE
subjectKeyIdentifier    = hash
keyUsage                = critical,digitalSignature,keyEncipherment,keyAgreement
subjectAltName          = @server_alt_names

[server_alt_names]
DNS.1 = grpc.example.com

[test_client]
basicConstraints        = critical,CA:FALSE
subjectKeyIdentifier    = hash
keyUsage                = critical,nonRepudiation,digitalSignature,keyEncipherment
extendedKeyUsage        = critical,clientAuth
```

Step 2 Copy the following content and save it to the **create.sh** file, which must be located in the same directory as the **openssl.cnf** file:

```
#!/bin/bash

# Create the server CA certs.
openssl req -x509                               \
```



```
  -newkey rsa:4096                                \
  -nodes                                        \
  -days 3650                                     \
  -keyout ca_key.pem                               \
  -out ca_cert.pem                                \
  -subj /C=CN/ST=CA/L=SVL/O=gRPC/CN=test-server_ca/   \
  -config ./openssl.cnf                           \
  -extensions test_ca                             \
  -sha256

# Create the client CA certs.
openssl req -x509                                 \
  -newkey rsa:4096                                \
  -nodes                                        \
  -days 3650                                     \
  -keyout client_ca_key.pem                          \
  -out client_ca_cert.pem                          \
  -subj /C=CN/ST=CA/L=SVL/O=gRPC/CN=test-client_ca/   \
  -config ./openssl.cnf                           \
  -extensions test_ca                             \
  -sha256

# Generate a server cert.
openssl genrsa -out server_key.pem 4096
openssl req -new                                  \
  -key server_key.pem                             \
  -days 3650                                    \
  -out server_csr.pem                              \
  -subj /C=CN/ST=CA/L=SVL/O=gRPC/CN=test-server1/   \
  -config ./openssl.cnf                           \
  -reqexts test_server
openssl x509 -req          \
  -in server_csr.pem       \
  -CAkey ca_key.pem          \
  -CA ca_cert.pem          \
  -days 3650               \
  -set_serial 1000         \
  -out server_cert.pem       \
  -extfile ./openssl.cnf    \
  -extensions test_server   \
  -sha256
openssl verify -verbose -CAfile ca_cert.pem  server_cert.pem

# Generate a client cert.
openssl genrsa -out client_key.pem 4096
openssl req -new                                  \
  -key client_key.pem                             \
  -days 3650                                    \
  -out client_csr.pem                              \
  -subj /C=CN/ST=CA/L=SVL/O=gRPC/CN=test-client1/   \
  -config ./openssl.cnf                           \
  -reqexts test_client
openssl x509 -req          \
  -in client_csr.pem       \
  -CAkey client_ca_key.pem  \
  -CA client_ca_cert.pem    \
  -days 3650               \
  -set_serial 1000         \
  -out client_cert.pem       \
  -extfile ./openssl.cnf    \
  -extensions test_client   \
  -sha256
openssl verify -verbose -CAfile client_ca_cert.pem  client_cert.pem

rm *_csr.pem
```

Step 3 Run the following command to generate a certificate:

```
chmod +x ./create.sh && ./create.sh
```


After the command is executed, you can obtain the **server_key.pem** private key file and **server_cert.pem** certificate file.

Step 4 Run the following command to create a TLS secret named **grpc-secret**:

```
kubectl create secret tls grpc-secret --key server_key.pem --cert server_cert.pem
```

----End

Step 2: Creating a Workload for the gRPC Application

Create a gRPC-compliant workload in the cluster.

Step 1 Copy the following YAML content to create the **grpc.yaml** file: In this section, the Docker image built using **an official sample application** is used as an example.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    description: ''
  labels:
    appgroup: ''
    version: v1
  name: grpc-hello
  namespace: default
spec:
  selector:
    matchLabels:
      app: grpc-hello
      version: v1
  template:
    metadata:
      labels:
        app: grpc-hello
        version: v1
    spec:
      containers:
        - name: container-1
          image: {Image repository address}/{Organization name}/grpc-hello:latest  # The image in this section is for reference only.
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
  replicas: 1
```

Step 2 Run the following command to create the workload:

```
kubectl apply -f grpc.yaml
```

----End

Step 3: Creating a Service and Ingress for the Workload

Step 1 Copy the following YAML content to create the **grpc-svc.yaml** file:

```
apiVersion: v1
kind: Service
metadata:
  name: grpc-hello
  namespace: default
  labels:
    app: grpc-hello
spec:
  ports:
    - name: cce-service-0
```



```
    protocol: TCP
    port: 50051
    targetPort: 50051
  selector:
    app: grpc-hello
  type: NodePort
  sessionAffinity: None
```

Step 2 Run the following command to create the Service:

```
kubectl apply -f grpc-svc.yaml
```

Step 3 Copy the following YAML content to create the **grpc-ingress.yaml** file:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grpc-hello
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: GRPC # Specify the gRPC backend service.
spec:
  ingressClassName: nginx
  tls:
    - secretName: grpc-secret
  rules:
    - host: grpc.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: grpc-hello
                port:
                  number: 50051
```

Step 4 Run the following command to create ingress routing rules:

```
kubectl apply -f grpc-ingress.yaml
```

----End

Verification

After gRPCurl is installed, run the following command to check whether the installation is successful:

```
./grpcurl -insecure -servername "grpc.example.com" <ip_address>:443 list
```

In the preceding command, *<ip_address>* is the IP address of the load balancer, which can be obtained by running **kubectl get ingress**.

Expected output:

```
grpc.examples.echo.Echo
grpc.reflection.v1.ServerReflection
grpc.reflection.v1alpha.ServerReflection
helloworld.Greeter
```

7.4.4.4.6 Configuring Consistent Hashing for Load Balancing of an Nginx Ingress

The native Nginx supports multiple load balancing rules, including weighted round robin and IP hash. Nginx Ingress supports load balancing by using consistent hashing based on the native Nginx capabilities.

By default, the IP hash method supported by Nginx uses the linear hash space. The backend server is selected based on the hash value of the IP address.

However, when this method is used to add or delete a node, all IP addresses need to be hashed again and then routed again. As a result, a large number of sessions are lost or the cache becomes invalid. Therefore, consistent hashing is introduced to Nginx Ingress to solve this problem.

Consistent hashing is a special hash algorithm, which constructs a ring hash space to replace the common linear hash space. When a node is added or deleted, only the target route is migrated clockwise, and other routes do not need to be changed. In this way, rerouting can be reduced as much as possible, resolving the load balancing issue caused by dynamic node addition and deletion.

If a consistent hashing rule is configured, the newly added server will share the load of all other servers. Similarly, when a server is removed, all other servers can share the load of the removed server. This balances the load among nodes in the cluster and prevents the avalanche effect caused by the breakdown of a node.

Configuring a Consistent Hashing Rule

Nginx Ingress can use the **nginx.ingress.kubernetes.io/upstream-hash-by** annotation to configure consistent hashing rules. The following is an example:

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **ingress-test.yaml**. The file name can be customized.

```
vi ingress-test.yaml
```

For clusters of v1.23 or later:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"  # Perform hashing based on the
request URI.
spec:
  rules:
    - host: ''
      http:
        paths:
          - path: '/'
            backend:
              service:
                name: <your_service_name>  # Replace it with the name of your target Service.
                port:
                  number: <your_service_port>  # Replace it with the port number of your target Service.
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

For clusters of v1.21 or earlier:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-test
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"  # Perform hashing based on the
request URI.
spec:
```



```
rules:
  - host: ''
    http:
      paths:
        - path: '/'
          backend:
            serviceName: <your_service_name>  # Replace it with the name of your target Service.
            servicePort: <your_service_port>  # Replace it with the port number of your target Service.
```

The value of **nginx.ingress.kubernetes.io/upstream-hash-by** can be an nginx variable, a text value, or any combination:

- **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri"** indicates that requests are hashed based on the request URI.
- **nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri$host"** indicates that requests are hashed based on the request URI and domain name.
- **nginx.ingress.kubernetes.io/upstream-hash-by: "${request_uri}-text-value"** indicates that requests are hashed based on the request URI and text value.

Step 3 Create an ingress.

```
kubectl create -f ingress-test.yaml
```

If information similar to the following is displayed, the ingress has been created:

```
ingress/ingress-test created
```

Step 4 Check the created ingress.

```
kubectl get ingress
```

If information similar to the following is displayed, the ingress has been created:

```
NAME          CLASS   HOSTS     ADDRESS         PORTS   AGE
ingress-test  nginx   *         121.**.**.**    80      10s
```

----End

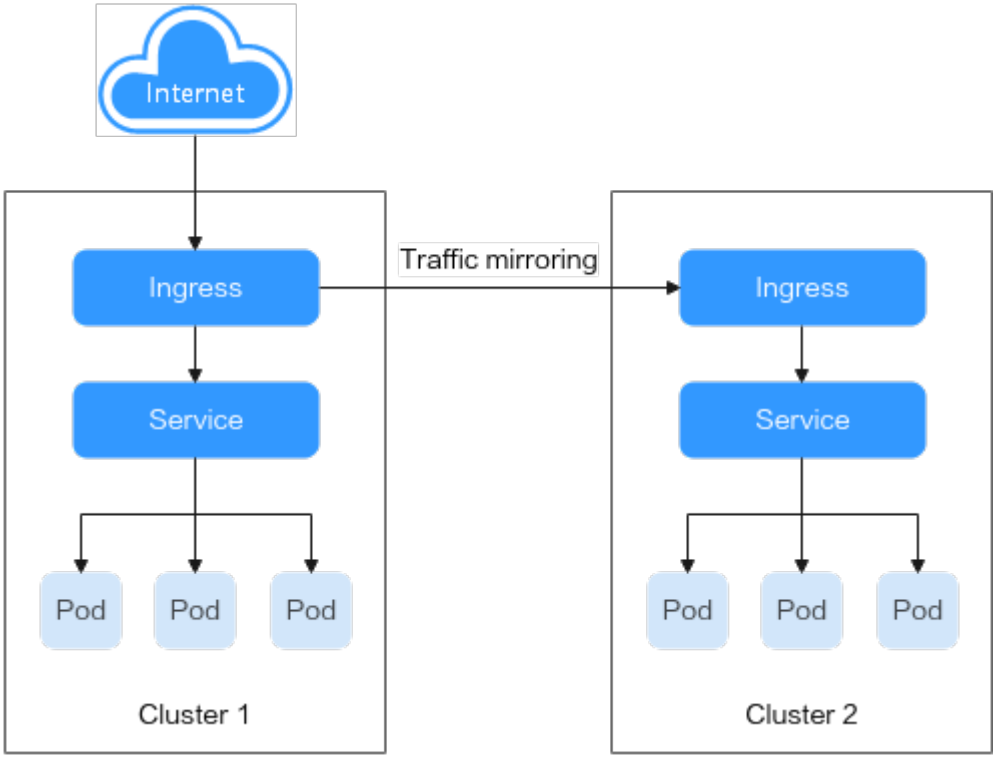
Documentation

Custom NGINX upstream hashing

7.4.4.4.7 Configuring Application Traffic Mirroring for an Nginx Ingress

Nginx ingresses can mirror inter-cluster traffic by duplicating access requests and sending them to the image backend without using the values returned from the backend. This allows for system simulation tests and fault locating without affecting the original requests. This section describes how to use Nginx ingresses to mirror traffic between applications in different clusters.

Figure 7-82 Traffic mirroring



Preparations

- Two CCE standard clusters, cluster 1 and cluster 2, are available. For details, see **Buying a CCE Standard/Turbo Cluster**.
- The **NGINX Ingress Controller** add-on has been installed in both clusters.
- kubectl has been installed and configured to access the two clusters. For details, see **Connecting to Multiple Clusters Using kubectl**.

Step 1: Creating a Workload and Exposing Services Through an Ingress

Step 1 Deploy an application in cluster 1 and test the connection.

Copy the following data and save it into the **nginx-deploy.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    version: v1
  name: nginx
  namespace: default
spec:
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      labels:
        app: nginx
        version: v1
    spec:
      containers:
        - name: container-1
```



```
        image: nginx:latest
        imagePullPolicy: IfNotPresent
    terminationGracePeriodSeconds: 30
    dnsPolicy: ClusterFirst
  replicas: 1
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
  namespace: default
spec:
  selector:
    app: nginx
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      port: 80
      protocol: TCP
  type: NodePort
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  namespace: default
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: nginx
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Step 2 Run the following command to create a workload for cluster 1:

```
kubectl apply -f nginx-deploy.yaml
```

Step 3 Repeat the preceding command to create the same workload in cluster 2.

----End

Step 2: Configuring Traffic Replication

NOTE

- As the destination of traffic, cluster 2 does not require any modifications. You only need to configure cluster 1.
- After the access traffic of the application in cluster 1 is replicated to the application in cluster 2, the client only receives the request response from cluster 1, while discarding the response from cluster 2.

Configure traffic mirroring for cluster 1. For more details, see **Nginx Ingress Mirror**.


```
kubectl patch ingress nginx --type merge --patch '"metadata": {"annotations": {"nginx.ingress.kubernetes.io/
mirror-target": "http://<cluster_2_ingress_ip_address>:80/", "nginx.ingress.kubernetes.io/mirror-host":
"example.com"}}'
```

In the preceding configuration, *<cluster_2_ingress_ip_address>* is the public IP address of the ingress load balancer in cluster 2.

Verification

Run the following command to access the application domain name **example.com** of cluster 1:

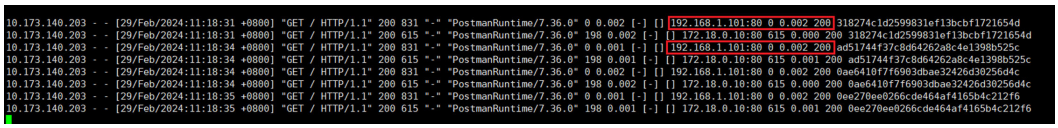
```
curl http://<cluster_1_ingress_ip_address>:80/ -H "Host: example.com"
```

Check nginx-ingress controller logs in cluster 1.

```
kubectl logs -f <cceaddon-nginx-ingress-controller_pod_name> -n kube-system
```

In the preceding command, *<cceaddon-nginx-ingress-controller_pod_name>* is the pod name of nginx-ingress controller.

According to the logs, whenever the domain name for the ingress in cluster 1 is requested, the request is duplicated and sent to the corresponding service in cluster 2.



7.4.4.4.8 Configuring Cross-Origin Access for Nginx Ingresses

The same-origin policy of browsers prevents a web page loaded by one origin from directly requesting resources from another origin in web development. Cross-origin resource sharing (CORS) is a secure solution that allows cross-origin requests.

CORS can be used in the following scenarios:

- Separated frontend and backend: In web development, frontend applications are often deployed under a domain name (for example, frontend.example.com), while the backend API service uses a different domain name (for example, api.example.com). This can cause cross-origin resource sharing to be blocked, but CORS can help resolve this issue by allowing the frontend to obtain data from the API service.
- Third-party service integration: If your website needs to call third-party APIs (for example, the APIs of a map service or social media login platform), configure CORS to enable cross-origin resource sharing.
- CDN: If your website is using a CDN to provide static resources from a domain name that differs from the primary domain, you can use CORS to load these resources.

Configuration Example

This section uses the communication between a frontend application (frontend.example.com) and a backend API service (api.example.com) as an example to describe how to add an annotation to the ingress for cross-origin access. For details about the annotation, see **Enable CORS**.

Example configuration:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-ingress
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/enable-cors: "true"     # Enable CORS.
    nginx.ingress.kubernetes.io/cors-allow-origin: "*" # Configure the origin that can be accessed.
    nginx.ingress.kubernetes.io/cors-allow-methods: "GET, PUT, POST, DELETE, PATCH, OPTIONS" # Allowed HTTP request methods
    nginx.ingress.kubernetes.io/cors-allow-headers: "DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range" # Allowed request headers
    nginx.ingress.kubernetes.io/cors-expose-headers: "Content-Length,Content-Range" # Response headers to be exposed
    nginx.ingress.kubernetes.io/cors-max-age: "3600" # Cache duration of a pre-check request
spec:
  rules:
    - host: api.example.com
      http:
        paths:
          - path: /
            backend:
              service:
                name: example
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
            pathType: ImplementationSpecific
  ingressClassName: nginx
```

Table 7-154 Parameters for configuring cross-origin access

Parameter	Description	Example
nginx.ingress.kubernetes.io/enable-cors	Enable CORS to allow cross-origin access.	nginx.ingress.kubernetes.io/enable-cors: "true"

Parameter	Description	Example
nginx.ingress.kubernetes.io/cors-allow-origin	Specify **Access-Control-Allow-Origin** for the origin that can be accessed. The parameter value is in the format of **http(s)://example.com** or **http(s)://example.com:port**. You can enter multiple values by separating them with commas (,). Wildcards are also supported. For example: ● ***** indicates that access from all domains is allowed. ● Enter a level-1 wildcard domain name. For example, **http(s)://*.example.com** indicates that all subdomain names under the domain name can be accessed. **NOTICE** To enhance security, specify a domain name to prevent unauthorized access to sensitive resources from other domains.	nginx.ingress.kubernetes.io/cors-allow-origin: "https://example.com"
nginx.ingress.kubernetes.io/cors-allow-methods	Specify **Access-Control-Allow-Methods** for allowed HTTP request methods. You can enter multiple values by separating them with commas (,).	nginx.ingress.kubernetes.io/cors-allow-methods: "GET, PUT, POST, DELETE, PATCH, OPTIONS"
nginx.ingress.kubernetes.io/cors-allow-headers	Specify **Access-Control-Allow-Headers** for allowed request headers. You can enter multiple values by separating them with commas (,).	nginx.ingress.kubernetes.io/cors-allow-headers: "DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range"

Parameter	Description	Example
nginx.ingress.kubernetes.io/cors-allow-credentials	Specify **Access-Control-Allow-Credentials** to control the sending of credentials (such as cookies). Options: • **true**: Credentials can be sent. • **false**: Credentials cannot be sent. **NOTICE** For security purposes, if **Access-Control-Allow-Credentials** is set to **true**, the wildcard (*) cannot be used for **Access-Control-Allow-Origin**. This prevents unauthorized domains from accessing sensitive resources, such as cookies or authorization headers containing user credentials.	nginx.ingress.kubernetes.io/cors-allow-credentials: "true"
nginx.ingress.kubernetes.io/cors-expose-headers	Specify **Access-Control-Expose-Headers** for custom response headers that can be accessed by a cross-origin request. This allows you to retrieve non-standard response headers using client-side JavaScript code. For security purposes, browsers can only access standard response headers including **Cache-Control**, **Content-Language**, **Content-Type**, **Expires**, **Last-Modified**, and **Pragma**, when handling cross-origin requests, unless otherwise specified. You can enter multiple values by separating them with commas (,).	nginx.ingress.kubernetes.io/cors-expose-headers: "Content-Length,Content-Range"

Parameter	Description	Example
nginx.ingress.kubernetes.io/cors-max-age	Specify **Access-Control-Max-Age** for the cache duration of a CORS pre-check request. Properly configure this parameter based on service requirements. If the value is too low, pre-check requests may happen frequently. If the value is too high, the CORS policy may not take effect immediately.	nginx.ingress.kubernetes.io/cors-max-age: "3600"

Verifying Cross-Origin Access

Use curl to send a cross-origin request and check the response header.

```
curl -X OPTIONS -H 'Origin: {Origin}' {Accessed ingress URL} -i
```

For example, if the origin is **frontend.example.com** and the accessed ingress URL is **api.example.com**, run the following command:

```
curl -X OPTIONS -H 'Origin: frontend.example.com' api.example.com -i
```

The **Access-Control-**** request header will be added to the original backend response.

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS
Access-Control-Allow-Headers: DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range
Access-Control-Expose-Headers: Content-Length,Content-Range
Access-Control-Allow-Credentials: true
Content-Type: application/json
```

7.4.4.4.9 Nginx Ingress Usage Suggestions

Nginx ingresses use NGINX Ingress Controller (**NGINX Ingress Controller**) in clusters to balance load and control access for traffic. The stability of NGINX Ingress Controller, which is deployed in clusters and uses open-source community charts and images, is closely tied to the configurations used and the current status of the cluster. This section provides practical tips on how to use NGINX Ingress Controller effectively. You can refer to the following configurations for the best results.

Optimized NGINX Ingress Controller Settings

- **Configure a proper number of replicas and resource limits.**

  The default number of pods for NGINX Ingress Controller installed through Settings is 2, but you can modify it according to your service needs.

  When NGINX Ingress Controller is deployed, multiple pods will be allocated to different AZs or different nodes in the same AZ by default.

Additionally, configure proper resource limits for NGINX Ingress Controller to prevent traffic interruption caused by OOM. It is a good practice to set the CPU limit to 1000m or higher and the memory limit to 2 GiB or higher.

- **Deploy NGINX Ingress Controller on a dedicated node for better performance and stability of Nginx Ingress.**

  When creating a node, configure taints on it. During the installation of NGINX Ingress Controller, add the taint of the node to the tolerance policy. For details, see **Deploying the Add-on Pods on Dedicated Nodes**.

- **Optimize the Nginx ingress performance.**

  NGINX Ingress Controller performance optimization involves system parameter optimization and Nginx parameter optimization.

  - System parameter optimization: Huawei Cloud has optimized some common OS parameters by default to improve system performance. However, you should also optimize other parameters like connection queue size and source port range for better results. The optimized system parameters ensure that Nginx can handle a high volume of concurrent requests and access the backend without running out of ports.
  - Nginx parameter optimization

    - **In a high-concurrency environment**, increase the maximum number of keepalive connections between Nginx and the client. This helps to avoid generating a large number of TIME_WAIT connections.
    - Increase the maximum number of connections to a single worker node for NGINX Ingress Controller to handle a high volume of concurrent requests.
    - **In a high-concurrency environment**, you are advised to set the maximum number of idle keepalive connections to 1000.
    - Configure gateway timeout. Make sure that the persistent connection timeout period of backend services is greater than or equal to the connection timeout period of NGINX Ingress Controller.

- **Configure HPA for NGINX Ingress Controller.**

  NGINX Ingress Controller can typically handle bursts of service traffic. However, if you need additional capacity in heavy load scenarios, you can configure HPA to automatically expand the capacity for NGINX Ingress Controller. For details, see **Creating an HPA Policy**.

- **Configure a proper preStop hook for backend services.**

  During the rolling update of a backend service, NGINX Ingress Controller will remove the pods that are being terminated from the backend server, but will retain the connections for requests that are still being processed. If the backend service pods exit immediately after receiving the end signal, the in-progress requests may fail or some traffic may be forwarded to pods that have already exited, leading to traffic loss. To prevent this issue, configure a preStop hook in the backend service pods. This will allow the pods to continue working for a period of time after being removed.

  The following shows a workload configuration example:

```
...
spec:
  template:
```



```
spec:
  containers:
  - name: app
    lifecycle:
      # Configure a preStop hook for pods to exit after 30 seconds.
      # The sleep command must exist in the container.
      preStop:
        exec:
          command:
          - sleep
          - 30
```

Observability of NGINX Ingress Controller

With Huawei Cloud's **Cloud Native Cluster Monitoring**, you can easily monitor NGINX Ingress Controller and gain valuable insights into the status of your ingress traffic. For details, see **Monitoring Metrics of NGINX Ingress Controller**.

Advanced Functions of NGINX Ingress Controller

- **Multiple NGINX Ingress Controllers allowed**

  Deploying multiple NGINX Ingress Controllers in a cluster is a great way to isolate your application's internal network from the external network. For details, see **Installing Multiple NGINX Ingress Controllers**.

- **Blue-green deployment or grayscale release of applications through NGINX Ingress Controller**

  For details, see **Using Nginx Ingress to Implement Grayscale Release and Blue-Green Deployment**.

- **Configuring redirection rules through NGINX Ingress Controller**

  You can use Nginx Ingress to configure redirection rules. For details, see **Configuring Redirection Rules for an Nginx Ingress**.

- **Interconnecting Nginx Ingress with different backend services**

  Nginx Ingress uses HTTP to access backend services by default, but it can also function as a proxy for backend services using other protocols.

  – HTTPS: For details, see **Configuring HTTPS Backend Services for an Nginx Ingress**.

  – gRPC: For details, see **Configuring gRPC Backend Services for an Nginx Ingress**.

7.4.4.4.10 Optimizing NGINX Ingress Controller in High-Traffic Scenarios

Ingress objects provide Layer 7 protocols like HTTP and HTTPS for clusters. Among the available options, Nginx ingresses are widely used. CCE has developed a featured open-source add-on called NGINX Ingress Controller, which is an enhanced version of the community solution. This add-on offers advanced Layer 7 load balancing features. In high-concurrency scenarios, the pre-allocated resources such as CPU and memory and network connections of the add-on may not be robust enough to handle the application needs. This can affect the application performance. This section describes how to optimize NGINX Ingress Controller in high-traffic scenarios.

> **CAUTION**
>
> The optimization involves the rolling upgrade of the NGINX ingress container. You are advised to perform the optimization during off-peak hours.

Prerequisites

The NGINX Ingress Controller add-on has been installed in the target cluster.

Suggestions

The following provides some suggestions on optimizing the NGINX Ingress Controller add-on in high-traffic scenarios.

Using a High-Performance Node

In high-concurrency scenarios, ingresses typically need a large number of CPUs and network connections, so you can select a computing-plus ECS to run the add-on instances.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane. In the right pane, click the **Node Pools** tab.

Step 3 Click **Create Node Pool** in the upper right corner to create a node pool and add two nodes to the node pool. The node pool can be dedicated for the NGINX Ingress Controller add-on. For details, see **Creating a Node Pool**.

For high-concurrency scenarios, you can select a high-performance ECS such as the general computing-plus c7.8xlarge.2 ECS, with 32 cores, 64 GiB of memory, a maximum bandwidth of 30 Gbit/s, and 5.5 million PPS.

Step 4 In the **Advanced Settings** area, configure labels and taints for the node pool.

- Configure a taint and set its key to **nginx-ingress-pod-reserved**, value to **true**, and the effect to **NoExecute**.
- Configure a label and set its key to **nginx-ingress-pod-reserved** and value to **true**.

Step 5 In the navigation pane, choose **Add-ons**, edit the configurations of NGINX Ingress Controller, and modify the scheduling policies.

- **Node Affinity**: Set it to **Custom policies** and configure the add-on to be affinity for the **nginx-ingress-pod-reserved: true** label.
- **Toleration**: Add a toleration and configure the add-on to tolerate the taint whose label is **nginx-ingress-pod-reserved: true** and the taint policy is **NoExecute**.

----End

Modifying the NGINX Ingress Controller Configuration

You can perform the following operations when installing or editing NGINX Ingress Controller:

- Change the resource limit of the **nginx-ingress-controller** container. If an application processes HTTP requests with a PRS of 300,000, you can configure the CPU and memory limits, as well as the requested CPU and memory, to 16 vCPU cores and 20 GiB of memory.
- Set the number of pods to be greater than or equal to 2.
- Disable the metric collection of the add-on.

  If you do not need to obtain metrics, you are advised to disable metric collection to reduce the CPU and memory usage of the Nginx container.
- Optimize the global configuration by doing as follows:
  - **Increase the maximum number of requests that a keepalive connection can handle.**

    Nginx provides the **keepalive_requests** parameter to control how many requests a single keepalive connection between a client and an upstream server can handle. This parameter defaults to 100.

    If the number of requests in a keepalive connection exceeds this limit, the connection will be disconnected and then re-established. For an ingress in a private network, a single client may have a high QPS, such as 10,000 QPS. This can cause Nginx to frequently disconnect keepalive connections from the client, resulting in a large number of connections in the TIME_WAIT state.

    To avoid this, it is recommended that you increase the maximum number of requests for a keepalive connection between Nginx and the client in high-concurrency scenarios. This can be achieved by setting **keep-alive-requests** in Nginx ingresses to **10000**. For details, see **keep-alive-requests**.

    You can also control the number of requests in a keepalive connection between the client and the upstream server by setting **upstream-keepalive-requests**. For details, see **upstream-keepalive-requests**.

    CAUTION

    The **upstream-keepalive-requests** parameter is not needed in a non-high-concurrency scenario. If you set this parameter to a high value, it may cause a load imbalance. This is because when Nginx maintains a keepalive connection with the upstream for too long, the number of times connections scheduled will decrease, resulting in a traffic load imbalance.

  - **Increase the maximum number of idle keepalive connections.**

    You can configure the **keepalive** parameter for the connections between Nginx and an upstream server. This parameter specifies the maximum number of idle connections and defaults to **320**.

    In a high-concurrency scenario, there are many requests and connections. However, in a real production environment, requests are not evenly distributed, and some connections may be temporarily idle. If the number of idle connections increases, they will be terminated. This can result in Nginx frequently disconnecting and reconnecting to the upstream server, leading to a significant increase in the TIME_WAIT connections. To avoid

this, it is recommended that you set **keepalive** to **1000** in high-concurrency scenarios. For details, see **upstream-keepalive-connections**.

- **Increase the maximum number of connections for a single worker.**

  The **max-worker-connections** parameter sets the maximum number of connections that each worker process can handle. For high-concurrency scenarios, it is recommended that you increase this value to, for instance, **65536**, to enable Nginx to manage more connections. For details, see **max-worker-connections**.

- **Reduce the gateway timeout.**

  Nginx connects to an upstream pod via TCP and communicates with it, using three timeout parameter configurations.

  - **proxy-connect-timeout**: specifies the timeout for establishing a connection between Nginx and an upstream pod. The default value for an Nginx ingress is 5 seconds. Nginx and services communicate with each other in the same data center over a private network, so the timeout can be reduced to, for example, 3 seconds. For details, see **proxy-connect-timeout**.
  - **proxy-read-timeout** and **proxy-send-timeout**: control the timeout of read and write operations between Nginx and an upstream pod. The values for an Nginx ingress default to 60 seconds. However, if there are service exceptions that cause a sharp increase in response time, abnormal requests can occupy the ingress gateway for an extended period. To prevent this, it is recommended that, after the latency of all normal service requests reached P99.99, you reduce the read/write timeout between the gateway and upstream pod to a suitable value, such as 30 seconds. This allows Nginx to interrupt abnormal requests promptly and prevent prolonged breakdowns. For details, see **proxy-read-timeout** and **proxy-send-timeout**.

Optimizing Nginx Kernel Parameters

CAUTION

Before modifying a kernel parameter, it is important to have a full understanding of its meaning and function. Exercise caution when making changes, because incorrect settings can lead to unexpected system errors and instability.

You need to make sure that:

1. You have fully understood the functions and impacts of kernel parameters. This helps you set kernel parameters correctly.
2. The parameter values you entered must be valid and meet the expectation, or the modification will not take effect.

You can optimize kernel parameters of Nginx ingresses and configure kernel parameters using **initContainers** by doing as follows:

By default, CCE enables kernel parameter tuning for the NGINX Ingress Controller add-on of 2.2.75, 2.6.26, 3.0.1, and later versions.

- **Increase the size of the connection queue.**

  In a high-concurrency scenario, the connection queue may overflow if it is too small, resulting in the failure to establish some connections. The size of the connection queue for the process listener socket is determined by the **net.core.somaxconn** kernel parameter. By modifying this parameter, you can increase the size of the Nginx ingress connection queue.

  When a process uses the listen system to listen on ports, it passes in the **backlog** parameter, which sets the size of the socket connection queue. The value of **backlog** cannot exceed that of **somaxconn**. The Go program standard library uses the **somaxconn** value as the default queue size when listening. However, Nginx does not read **somaxconn** when listening on the socket. Instead, it reads **nginx.conf**. In the configuration items for listening ports in **nginx.conf**, you can set the **backlog** parameter to specify the connection queue size for Nginx listening port. The following shows an example configuration:

  ```
  server {
    listen  80  backlog=1024;
    ...
  ```

  If the value **backlog** is not specified, the default value **511** is used. By default, the maximum size of the connection queue for the Nginx listening port is 511, even if the value of **somaxconn** is greater than **511**. This can lead to connection queue overflow in high-concurrency scenarios.

  The NGINX Ingress Controller can automatically read and use the value of **somaxconn** as the **backlog** value, which is then written to the generated **nginx.conf** file. This means that the connection queue size for an Nginx ingress is determined solely by **somaxconn**, and the default size in CCE is **4096**. In a high-concurrency scenario, it is recommended that you run the following command to set **somaxconn** to **65535**:

  ```
  sysctl -w net.core.somaxconn=65535
  ```

- **Expand the range of source ports.**

  In a high-concurrency scenario, an Nginx ingress establishes connections with an upstream server using a large number of source ports. The range of source ports is randomly selected from the range defined in the **net.ipv4.ip_local_port_range** kernel parameter. In such scenarios, a small port range can quickly exhaust source ports, leading to abnormal connections. The default source port range for pods created in CCE is 32768 to 60999. To expand the range to 1024 to 65535, run the following command:

  ```
  sysctl -w net.ipv4.ip_local_port_range="1024 65535"
  ```

- **Adjust TIME_WAIT.**

  You can enable TIME_WAIT reuse for Nginx ingresses, allowing TIME_WAIT connections to be reused for new TCP connections. Additionally, reducing the value of **net.ipv4.tcp_fin_timeout** in FIN_WAIT2 state and **net.netfilter.nf_conntrack_tcp_timeout_time_wait** in the TIME_WAIT state can help release resources occupied by them more quickly. To enable TIME_WAIT reuse, run the following commands:

  ```
  sysctl -w net.ipv4.tcp_fin_timeout=15
  sysctl -w net.netfilter.nf_conntrack_tcp_timeout_time_wait=30
  ```

Add **initContainers** to NGINX Ingress Controller pods and configure the preceding kernel parameters. The following shows an example:

```
...
initContainers:
```



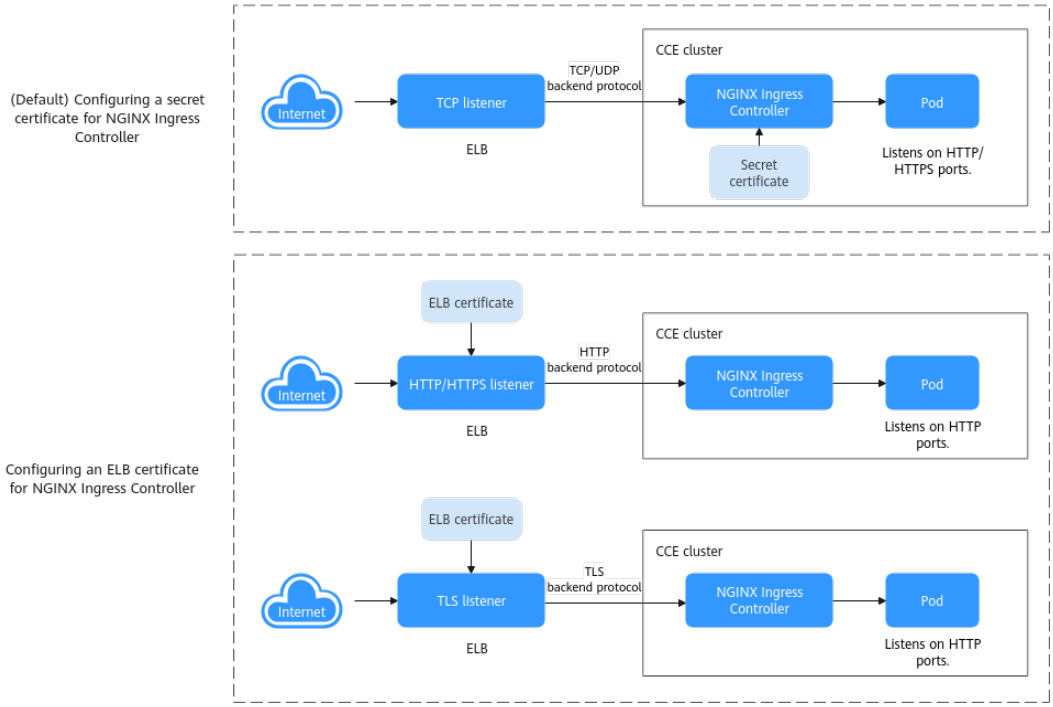
```
- name: setsysctl
  image: ***(By default, CCE uses the nginx-ingress image of the community.)
  securityContext:
    runAsUser: 0
    runAsGroup: 0
    capabilities:
      add:
        - SYS_ADMIN
      drop:
        - ALL
  command:
    - sh
    - -c
    - |
      if [ "$POD_IP" != "$HOST_IP" ]; then
        mount -o remount rw /proc/sys
        if [ $? -eq 0 ]; then
          sysctl -w net.core.somaxconn=65535
          sysctl -w net.ipv4.ip_local_port_range="1024 65535"
          sysctl -w net.ipv4.tcp_fin_timeout=15
          sysctl -w net.netfilter.nf_conntrack_tcp_timeout_time_wait=30
        else
          echo "Failed to remount /proc/sys as read-write. Skipping sysctl commands."
        fi
      fi
  env:
    - name: POD_IP
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: status.podIP
    - name: HOST_IP
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: status.hostIP
```

7.4.4.4.11 Configuring an ELB Certificate for NGINX Ingress Controller

CCE provides the following options for configuring ingress certificates for the Nginx Ingress Controller add-on:

- Secret certificate. Import the required certificate to a CCE secret and specify the default server certificate (**default-ssl-certificate**) for the add-on.
- ELB certificate. Use an ELB certificate created on the ELB console. There is no need to specify a secret certificate for the add-on.

Figure 7-83 Differences between the two methods of configuring certificates for the NGINX Ingress Controller



This section describes how to configure an ELB certificate for the NGINX Ingress Controller add-on and use that ELB certificate to manage the certificates used by requests.

Prerequisites

- The version of the NGINX Ingress Controller add-on installed in the cluster must be 2.2.104, 2.6.53, 3.0.30, or later.
- Before enabling the NGINX Ingress Controller add-on, it is necessary to ensure that the associated dedicated load balancer meets the required protocols. For example, an application load balancer supports only HTTP and HTTPS. If TCP is required, the load balancer must support both application and network load balancing.
- With an ELB certificate configured, the default server certificate (**default-ssl-certificate**) of the NGINX Ingress Controller add-on cannot be used. All external requests must carry the ELB certificate to access services within the cluster.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **NGINX Ingress Controller** on the right, and click **Install**.

If the add-on has been installed, click **Manage**, select the pod for which you want to configure an ELB certificate, and click **Edit**.

Step 2 In the **Extended Parameter Settings** area, modify the parameters settings.

1. Configure the **service.annotations** parameter and check whether the protocol is correct.

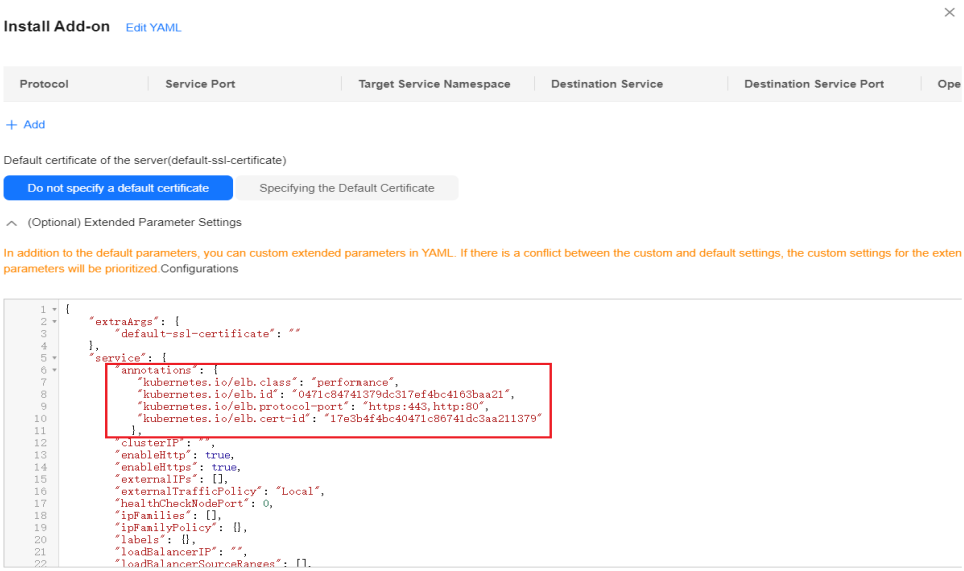
   An example is as follows:

```
...
      "service": {
        "annotations": {
          "kubernetes.io/elb.class": "performance",
          "kubernetes.io/elb.id": "*****",
          "kubernetes.io/elb.protocol-port": "https:443,http:80",
          "kubernetes.io/elb.cert-id": "*****"
      },
...
```

Parameter	Value Type	Description
kubernetes.io/elb.class	String	Load balancer type. – **union**: shared load balancer – **performance**: dedicated load balancer
kubernetes.io/elb.id	String	The load balancer ID. You can view the load balancer ID on the ELB console.
kubernetes.io/elb.protocol-port	String	The listener port of the NGINX Ingress Controller add-on. The listener protocol must match the type supported by the associated load balancer. For example, if the HTTP/HTTPS protocol is used, a dedicated load balancer that supports application load balancing is needed. – HTTP/HTTPS: If ports 443 and 80 are used for HTTPS and HTTP, respectively, the parameter value is *https:443,http:80*. – TLS: If ports 443 and 80 are used for TLS, the parameter value is *tls:443,tls:80*. **NOTE** ■ To configure TLS for ingresses, make sure the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or a later version. ■ TLS relies on ELB. Before enabling TLS on an ingress, check whether TLS is supported in the current region.

Parameter	Value Type	Description
kubernetes.io/elb.cert-id	String	ID of the ELB certificate. To obtain the certificate, log in to the CCE console, choose **Service List** > **Networking** > **Elastic Load Balance**, and click **Certificates** in the navigation pane. In the load balancer list, copy the ID under the target certificate name.

For details about the preceding parameters, see **Configuring HTTP/HTTPS for a LoadBalancer Service**.



2. Configure the **targetPort** parameter to direct the load balancer access traffic to the HTTP port of the container.

   An example is as follows:

```
...
      "targetPorts": {
         "http": "http",
         "https": "http"
      },
...
```

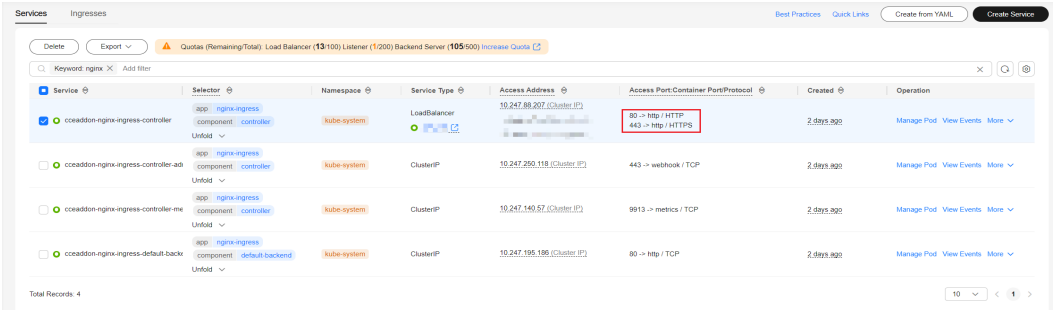
   NOTE

   Once an ELB certificate is configured, the default server certificate (**default-ssl-certificate**) configured for the NGINX Ingress Controller becomes unavailable and invalid. It is essential for external requests to include the configured ELB certificate, because requests without it will not be processed properly.

Step 3 Configure other mandatory parameters as required and click **Install**. For details about the parameters, see **NGINX Ingress Controller**.

Step 4 After completing the configuration, choose **Services & Ingresses** in the navigation pane, switch to the **kube-system** namespace, and check the protocols and

listening ports of the add-on. In this example, the protocol is HTTP and HTTPS instead of TCP.



----End

7.4.5 Migrating Data from a Bring-Your-Own Nginx Ingress to a LoadBalancer Ingress

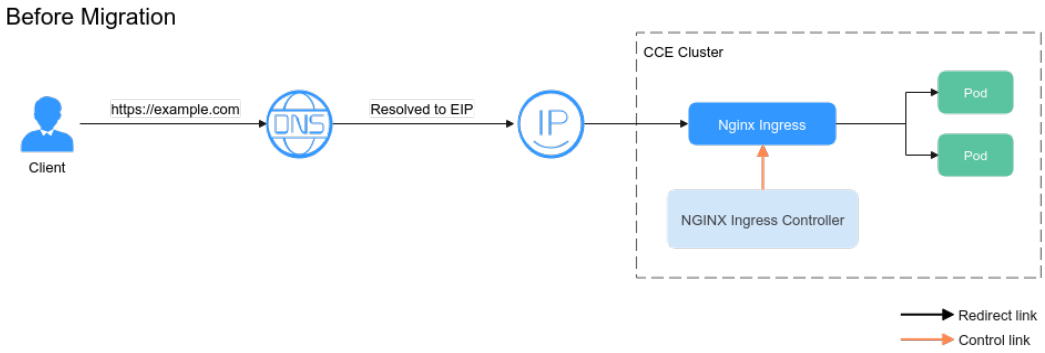
This section describes how to migrate data from a bring-your-own Nginx ingress to a LoadBalancer ingress.

LoadBalancer ingresses are implemented based on Huawei Cloud ELB. LoadBalancer ingresses offer better traffic management compared to bring-your-own Nginx ingresses because of the following advantages:

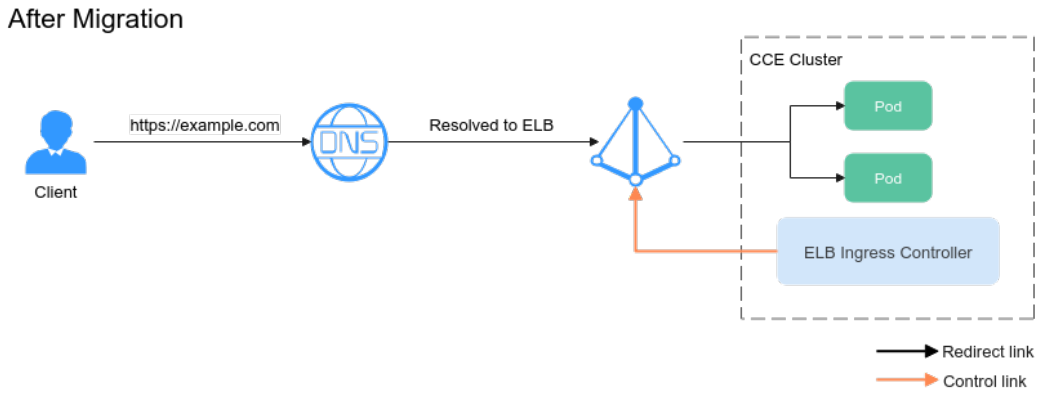
- Fully managed and O&M-free: ELB is a fully managed cloud service that requires no worker node, making it completely O&M-free.
- High availability: ELB enables active-active disaster recovery within a city across AZs. This ensures seamless traffic switchover in the event of a failure. Dedicated load balancers provide a comprehensive health check system to ensure that incoming traffic is only routed to healthy backend servers, improving the availability of your applications.
- Auto scaling: ELB can automatically scale to handle traffic spikes.
- Ultra-high performance: A single load balancer supports up to 1 million queries per second and tens of millions of concurrent connections.
- Integration with cloud services: ELB can run with various cloud services, such as WAF.
- Hot updates of configurations: Configuration changes can be updated in real-time without requiring a process reload. This helps to prevent disruptions to persistent connections.

Example Scenarios

A company is using a CCE cluster on Huawei Cloud to run their services. They have set up ingress forwarding rules using their NGINX Ingress Controller, which includes domain-based and URL-based forwarding rules. Their services are accessible externally through a DNS domain name. When a client requests access to the domain name example.com, Nginx Ingress forwards the requests to the pod associated with the Service based on the forwarding rules created on the ingress.



To develop services, the company must transfer their data from Nginx ingress to a LoadBalancer ingress. To maintain service stability, they want to keep the DNS domain name and IP addresses of their backend servers unchanged. To meet this requirement, you can configure LoadBalancer ingress rules in the CCE cluster to preserve the previous forwarding rules. Then, configure weighted DNS records to transfer data from the Nginx ingress to the LoadBalancer ingress.



NOTICE

- Migrate data during off-peak hours.
- ELB is billed. For details, see **Billing Overview**.

Configuring the Target LoadBalancer Ingress

Step 1 Create an ingress for migration and route it to the original backend service.

In this example, an existing load balancer is used to create an ingress, and the **elbingress.yaml** is for reference only. You can also choose to automatically create a load balancer associated with the ingress. For more information, see **Creating a LoadBalancer Ingress Using kubectl**.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    kubernetes.io/elb.id: <your_elb_id>  # Replace it with the ID of your existing load balancer.
    kubernetes.io/elb.ip: <your_elb_ip>  # Replace it with the IP of your existing load balancer.
    kubernetes.io/elb.class: performance  # A dedicated load balancer is used.
```



```
    kubernetes.io/elb.port: '80'
spec:
  rules:
  - host: 'example.com'          # Replace it with your domain name.
    http:
      paths:
      - path: '/'
        backend:
          service:
            name: <your_service_name>  # Replace it with the name of your target Service, which is the same as the Service associated with the original Nginx ingress.
            port:
              number: 8080          # Replace 8080 with the port number of your target Service.
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
  ingressClassName: cce
```

Step 2 Run the following command to create the ingress:

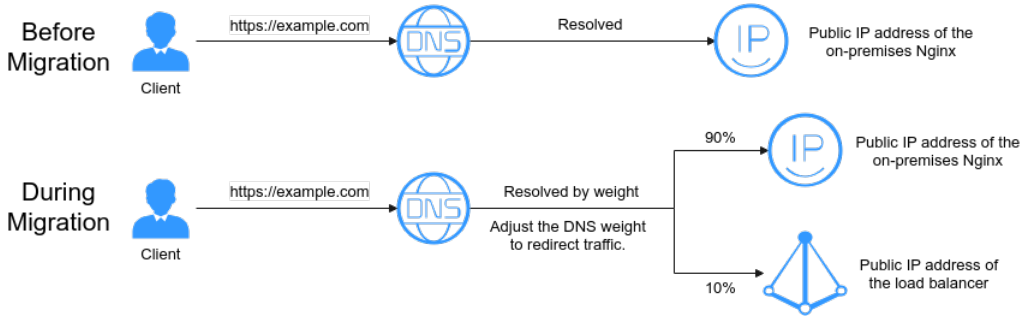
```
kubectl apply -f elbingress.yaml
```

----End

Redirecting Traffic to the LoadBalancer Ingress

Before redirecting traffic, ensure you have created record A to map the service domain name to the public IP address of the company's Nginx ingress. You can configure weighted DNS records to transfer data from the Nginx ingress to the LoadBalancer ingress.

Figure 7-84 Using DNS to redirect traffic



To redirect traffic, do as follows:

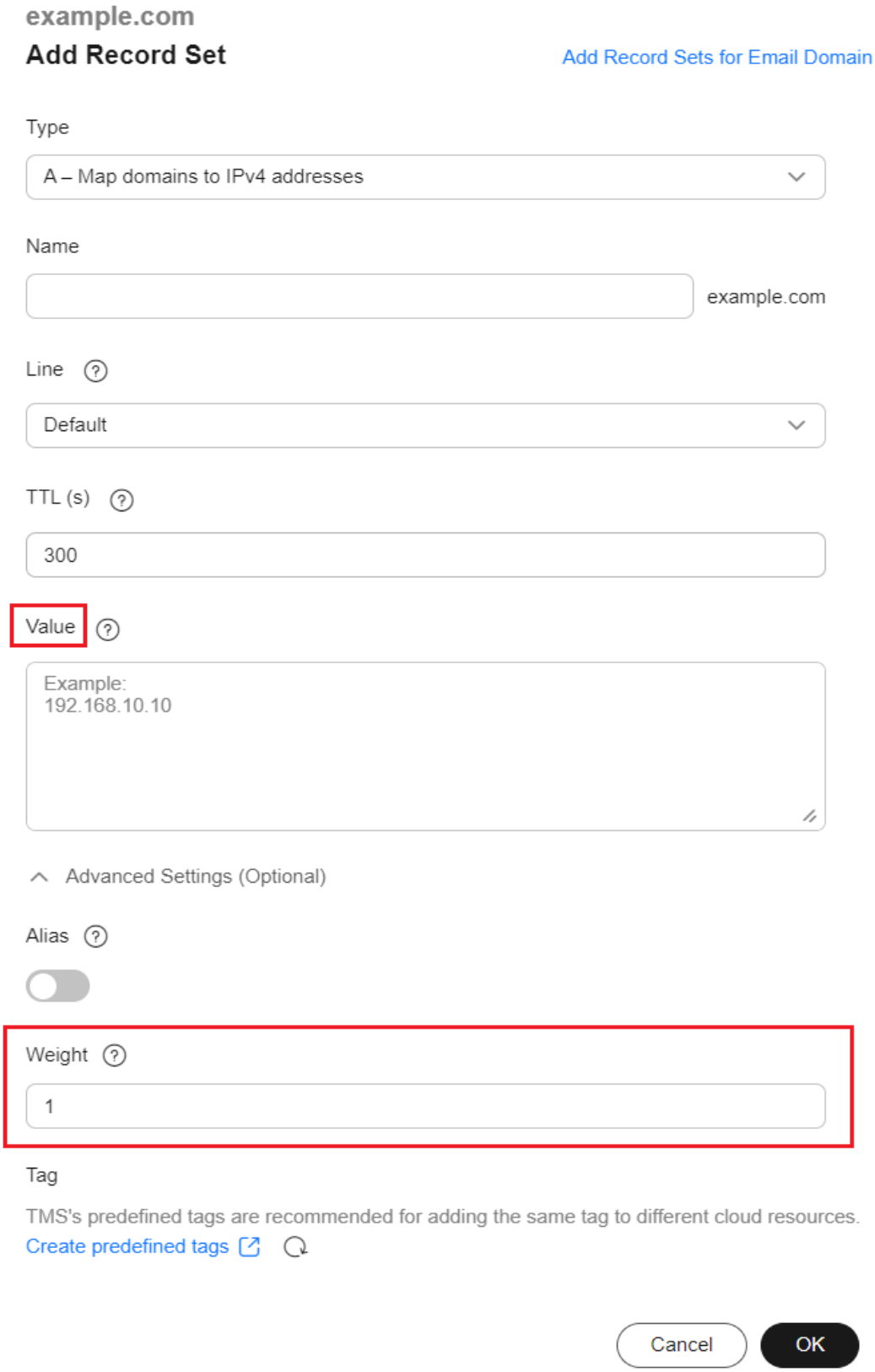
Step 1 Go to **Public Zones**.

Step 2 On the **Public Zones** page, click the domain name (**example.com**) of the public zone.

Step 3 The **Record Sets** page is displayed. You can find an A record, which contains the public IP address of the Nginx ingress. Change the weight of the A record to 9.

Step 4 Add an A record set. To do so, click **Add Record Set**, and set **Value** to the public IP address of the target LoadBalancer ingress and **Weight** to **1**.

Figure 7-85 Adding a record set



Step 5 Gradually increase the weight of the DNS record for the LoadBalancer ingress to 100% without affecting any services.

----End

Deleting Redundant Resources

After all persistent connections to the Nginx ingress are released and there is no more traffic being forwarded to it, you can safely release any unnecessary resources such as the Nginx ingress, NGINX ingress Controller, and webhook after observing for a period of time.

7.5 DNS

7.5.1 Overview

Introduction to CoreDNS

When you create a cluster, the **CoreDNS add-on** is installed to resolve domain names in the cluster.

You can view the pod of the CoreDNS add-on in the kube-system namespace.

```
$ kubectl get po --namespace=kube-system
NAME                       READY   STATUS    RESTARTS   AGE
coredns-7689f8bdf-295rk    1/1     Running   0          9m11s
coredns-7689f8bdf-h7n68    1/1     Running   0          11m
```

After CoreDNS is installed, it becomes a DNS. After the Service is created, CoreDNS records the Service name and IP address. In this way, the pod can obtain the Service IP address by querying the Service name from CoreDNS.

nginx.<namespace>.svc.cluster.local is used to access the Service. **nginx** is the Service name, **<namespace>** is the namespace, and **svc.cluster.local** is the domain name suffix. In actual use, you can omit **<namespace>.svc.cluster.local** in the same namespace and use the ServiceName.

An advantage of using ServiceName is that you can write ServiceName into the program when developing the application. In this way, you do not need to know the IP address of a specific Service.

After CoreDNS is installed, there is also a Service in the kube-system namespace, as shown below.

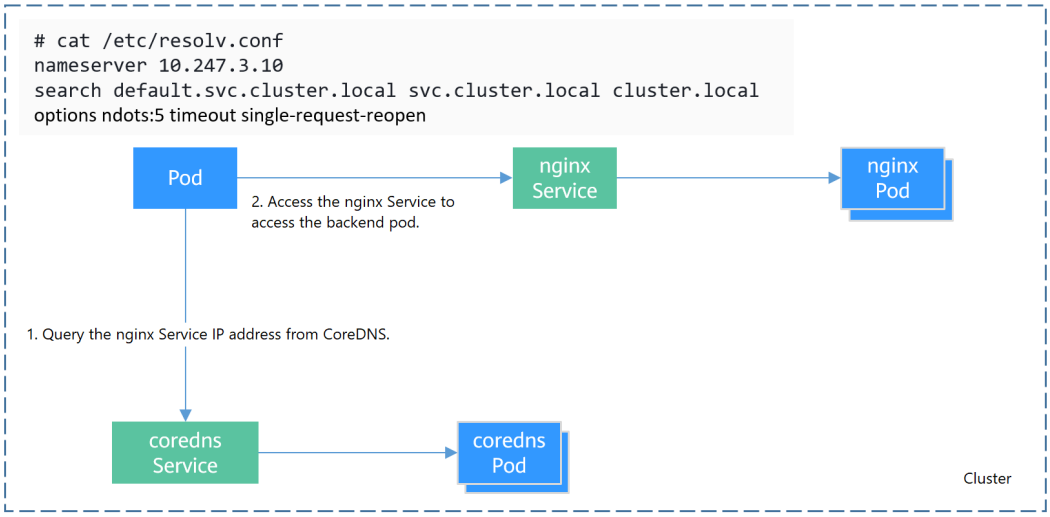
```
$ kubectl get svc -n kube-system
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP                PORT(S)                  AGE
coredns       ClusterIP   10.247.3.10    <none>                     53/UDP,53/TCP,8080/TCP   13d
```

By default, after other pods are created, the address of the CoreDNS Service is written as the address of the domain name resolution server in the **/etc/resolv.conf** file of the pod. Create a pod and view the **/etc/resolv.conf** file as follows:

```
$ kubectl exec test01-6cbbf97b78-krj6h -it -- /bin/sh
/ # cat /etc/resolv.conf
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5 timeout single-request-reopen
```


When a user accesses the *Service name:Port* of the Nginx pod, the IP address of the Nginx Service is resolved from CoreDNS, and then the IP address of the Nginx Service is accessed. In this way, the user can access the backend Nginx pod.

Figure 7-86 Example of domain name resolution in a cluster



How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be configured for each pod. Kubernetes supports DNS policies **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. For details, see **DNS for Services and Pods**. These policies are specified in the **dnsPolicy** field in the pod-specific.

- **Default**: Pods inherit the name resolution configuration from the node that the pods run on. The custom upstream DNS server and the stub domain cannot be used together with this policy.
- **ClusterFirst**: Any DNS query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream name server inherited from the node. Cluster administrators may have extra stub domains and upstream DNS servers configured.
- **ClusterFirstWithHostNet**: For pods running with hostNetwork, set its DNS policy **ClusterFirstWithHostNet**.
- **None**: It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the **dnsPolicy** field in the pod-specific.

NOTE

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

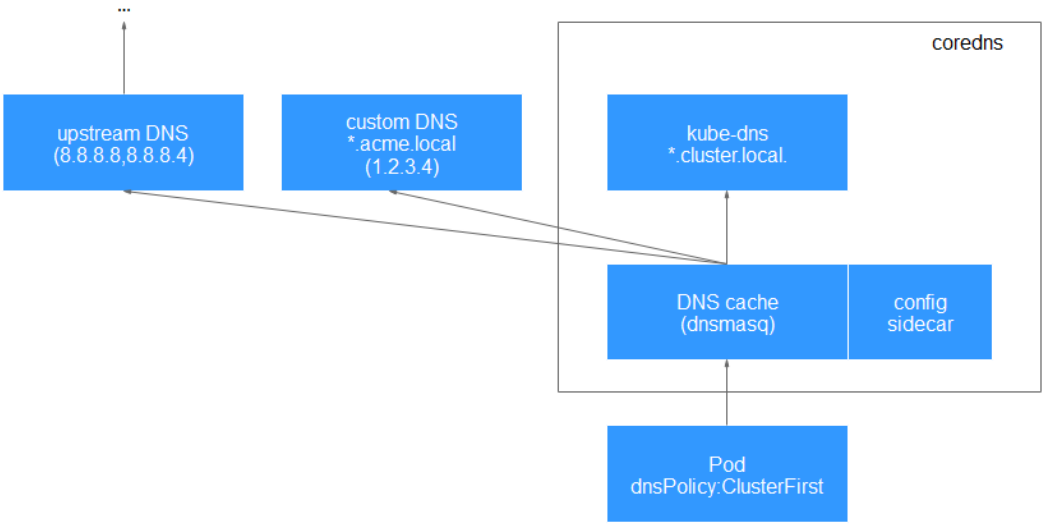
Routing

Without stub domain configurations: Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

With stub domain configurations: If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in CoreDNS.
2. From the caching layer, the suffix of the request is examined and then the request is forwarded to the corresponding DNS:
   - Names with the cluster suffix, for example, **.cluster.local**: The request is sent to CoreDNS.
   - Names with the stub domain suffix, for example, **.acme.local**: The request is sent to the configured custom DNS resolver that listens, for example, on 1.2.3.4.
   - Names that do not match the suffix (for example, **widget.com**): The request is forwarded to the upstream DNS.

Figure 7-87 Routing



Related Operations

You can also configure DNS in a workload. For details, see **DNS Configuration**.

You can also use CoreDNS to implement user-defined domain name resolution. For details, see **Using CoreDNS for Custom Domain Name Resolution**.

You can also use DNSCache to improve the DNS resolution performance. For details, see **Using NodeLocal DNSCache to Improve DNS Performance**.

7.5.2 DNS Configuration

Every Kubernetes cluster has a built-in DNS add-on (Kube-DNS or CoreDNS) to provide domain name resolution for workloads in the cluster. When handling a high concurrency of DNS queries, Kube-DNS/CoreDNS may encounter a performance bottleneck, that is, it may fail occasionally to fulfill DNS queries. Kubernetes workloads occasionally generate unnecessary DNS queries, which can overload the DNS system during periods of high query concurrency. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about DNS, see **CoreDNS**.

DNS Configuration Items

Run the **cat /etc/resolv.conf** command on a Linux node or container to view the DNS resolver configuration file. The following is an example DNS resolver configuration of a container in a Kubernetes cluster:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

Configuration Options

- **nameserver**: an IP address list of a name server that the resolver will query. If this parameter is set to 10.247.x.x, the resolver will query the kube-dns/CoreDNS. If this parameter is set to another IP address, the resolver will query a cloud or on-premises DNS server.
- **search**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. For CCE clusters, the search list is currently limited to three domains per container. When a nonexistent domain name is being resolved, eight DNS queries will be initiated because each domain name (including those in the search list) will be queried twice, one for IPv4 and the other for IPv6.
- **options**: options that allow certain internal resolver variables to be modified. Common options include timeout and ndots.

  The value **ndots:5** means that if a domain name has fewer than 5 dots (.), DNS queries will be attempted by combining the domain name with each domain in the search list in turn. If no match is found after all the domains in the search list are tried, the domain name is then used for DNS query. If the domain name has 5 or more than 5 dots, it will be tried first for DNS query. In case that the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list in turn.

  For example, the domain name **www.***.com** has only two dots (smaller than the value of **ndots**), and therefore the sequence of DNS queries is as follows: **www.***.com.default.svc.cluster.local**, **www.***.com.svc.cluster.local**, **www.***.com.cluster.local**, and **www.***.com**. This means that at least seven DNS queries will be initiated before the domain name is resolved into an IP address. It is clear that when many unnecessary DNS queries will be initiated to access an external domain name. There is room for improvement in workload's DNS configuration.

NOTE

For more information about configuration options in the resolver configuration file used by Linux operating systems, visit **http://man7.org/linux/man-pages/man5/resolv.conf.5.html**.

Configuring DNS for a Workload Using the Console

Kubernetes provides DNS-related configuration options for applications. The use of application's DNS configuration can effectively reduce unnecessary DNS queries in

certain scenarios and improve service concurrency. The following procedure uses an Nginx application as an example to describe how to add DNS configurations for a workload on the console.

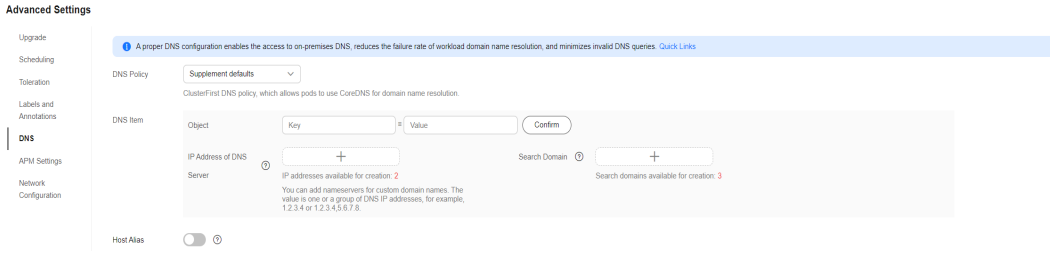
Step 1 Log in to the CCE console, access the cluster console, select **Workloads** in the navigation pane, and click **Create Workload** in the upper right corner.

Step 2 Configure basic information about the workload. For details, see **Creating a Workload**.

Step 3 In the **Advanced Settings** area, click the **DNS** tab and set the following parameters as required:

- **DNS Policy**: The DNS policies provided on the console correspond to the **dnsPolicy** field in the YAML file. For details, see **Table 7-155**.
  - **Supplement defaults**: corresponds to **dnsPolicy=ClusterFirst**. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks.
  - **Replace defaults**: corresponds to **dnsPolicy=None**. You must configure **IP Address** and **Search Domain**. Containers only use the user-defined IP address and search domain configurations for domain name resolution.
  - **Inherit defaults**: corresponds to **dnsPolicy=Default**. Containers use the domain name resolution configuration from the node that pods run on and cannot resolve the cluster-internal domain names.
- **Optional Objects**: The options parameters in the **dnsConfig field**. Each object may have a name property (required) and a value property (optional). After setting the properties, click **confirm to add**.
  - **timeout**: Timeout interval, in seconds.
  - **ndots**: Number of dots (.) that must be present in a domain name. If a domain name has dots fewer than this value, the operating system will look up the name in the search domain. If not, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name.
- **IP Address of DNS Server**: **nameservers** in **dnsConfig**. You can configure a domain name server for a custom domain name. The value is one or a group of DNS IP addresses.
- **Search Domain**: **searches** in the **dnsConfig**. A list of DNS search domains for hostname lookup in the pod. This property is optional. When specified, the provided list will be merged into the search domain names generated from the chosen DNS policy in **dnsPolicy**. Duplicate domain names are removed.
- **Host Alias**: Add the mapping between domain names and IP addresses to the local configuration file **/etc/hosts** of a pod for simplified local domain name resolution. For details, see **Adding entries to Pod /etc/hosts with HostAliases**

Figure 7-88 DNS configuration of a workload



Step 4 Click **Create Workload**.

----End

Configuring DNS Using the Workload YAML

When creating a workload using a YAML file, you can configure the DNS settings in the YAML. The following is an example for an Nginx application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: None
      dnsConfig:
        options:
          - name: ndots
            value: '5'
          - name: timeout
            value: '3'
        nameservers:
          - 10.2.3.4
        searches:
          - my.dns.search.suffix
```

- **dnsPolicy**

  The **dnsPolicy** field is used to configure a DNS policy for an application. The default value is **ClusterFirst**. The following table lists **dnsPolicy** configurations.

Table 7-155 dnsPolicy

Parameter	Description
ClusterFirst (default value)	Custom DNS configuration added to the default DNS configuration. By default, the application connects to CoreDNS (CoreDNS of the CCE cluster connects to the DNS on the cloud by default). The custom dnsConfig will be added to the default DNS parameters. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks. The search list (**search** option) and **ndots: 5** are present in the DNS configuration file. Therefore, when accessing an external domain name and a long cluster-internal domain name (for example, kubernetes.default.svc.cluster.local), the search list will usually be traversed first, resulting in at least six invalid DNS queries. The issue of invalid DNS queries disappears only when a short cluster-internal domain name (for example, kubernetes) is being accessed.
ClusterFirstWithHostNet	By default, the applications configured with the **host network** are interconnected with the DNS configuration of the node where the pod is located. The DNS configuration is specified in the DNS file that the kubelet **--resolv-conf** parameter points to. In this case, the CCE cluster uses the DNS on the cloud. If workloads need to use Kube-DNS/CoreDNS of the cluster, set **dnsPolicy** to **ClusterFirstWithHostNet** and container's DNS configuration file is the same as ClusterFirst, in which invalid DNS queries still exist. ... spec: containers: - image: nginx:latest imagePullPolicy: IfNotPresent name: container-1 restartPolicy: Always **hostNetwork: true** dnsPolicy: **ClusterFirstWithHostNet**
Default	The DNS configuration of the node where the pod is located is inherited, and the custom DNS configuration is added to the inherited configuration. Container's DNS configuration file is the DNS configuration file that the kubelet's **--resolv-conf** flag points to. In this case, a cloud DNS is used for CCE clusters. Both **search** and **options** fields are left unspecified. This configuration can only resolve the external domain names registered with the Internet, and not cluster-internal domain names. This configuration is free from the issue of invalid DNS queries.
None	The default DNS configuration is replaced by the custom DNS configuration, and only the custom DNS configuration is used. If **dnsPolicy** is set to **None**, the **dnsConfig** field must be specified because all DNS settings are supposed to be provided using the **dnsConfig** field.

NOTE

If the **dnsPolicy** field is not specified, the default value is **ClusterFirst** instead of **Default**.

- **dnsConfig**

  The **dnsConfig** field is used to configure DNS parameters for workloads. The configured parameters are merged to the DNS configuration file generated according to **dnsPolicy**. If **dnsPolicy** is set to **None**, the workload's DNS configuration file is specified by the **dnsConfig** field. If **dnsPolicy** is not set to **None**, the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated according to **dnsPolicy**.

  Table 7-156 dnsConfig

Parameter	Description
options	An optional list of objects where each object may have a name property (required) and a value property (optional). The contents in this property will be merged to the options generated from the specified DNS policy in **dnsPolicy**. Duplicate entries are removed.
nameservers	A list of IP addresses that will be used as DNS servers. If workload's **dnsPolicy** is set to **None**, the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in **dnsPolicy** with duplicate addresses removed. **NOTE** A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file. • If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS**, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid. • If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS** and **NodeLocal DNSCache**, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.
searches	A list of DNS search domains for hostname lookup in the pod. This property is optional. When specified, the provided list will be merged into the search domain names generated from the chosen DNS policy in **dnsPolicy**. Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.

Configuration Examples

The following example describes how to configure DNS for workloads.

- **Use Case 1: Using Kube-DNS/CoreDNS Built in Kubernetes Clusters**

  Scenario

  Kubernetes in-cluster Kube-DNS/CoreDNS applies to resolving only cluster-internal domain names or cluster-internal domain names + external domain names. This is the default DNS for workloads.

  Example:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    namespace: default
    name: dns-example
  spec:
    containers:
    - name: test
      image: nginx:alpine
    dnsPolicy: ClusterFirst
    imagePullSecrets:
      - name: default-secret
  ```

  Container's DNS configuration file:

  ```
  nameserver 10.247.3.10
  search default.svc.cluster.local svc.cluster.local cluster.local
  options ndots:5
  ```

- **Use Case 2: Using a Cloud DNS**

  Scenario

  A DNS cannot resolve cluster-internal domain names and therefore applies to the scenario where workloads access only external domain names registered with the Internet.

  Example:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    namespace: default
    name: dns-example
  spec:
    containers:
    - name: test
      image: nginx:alpine
    dnsPolicy: Default  # The DNS configuration file that the kubelet --resolv-conf parameter points to is used. In this case, the CCE cluster uses the DNS on the cloud.
    imagePullSecrets:
      - name: default-secret
  ```

  Container's DNS configuration file:

  ```
  nameserver 100.125.x.x
  ```

- **Use Case 3: Using Kube-DNS/CoreDNS for Workloads Running with hostNetwork**

  Scenario

  By default, a DNS is used for workloads running with hostNetwork. If workloads need to use Kube-DNS/CoreDNS, set **dnsPolicy** to **ClusterFirstWithHostNet**.

  Example:

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    name: nginx
  spec:
    hostNetwork: true
  ```



```
dnsPolicy: ClusterFirstWithHostNet
containers:
- name: nginx
  image: nginx:alpine
  ports:
  - containerPort: 80
imagePullSecrets:
  - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 4: Customizing Application's DNS Configuration**

  Scenario

  You can flexibly customize the DNS configuration file for applications. Using **dnsPolicy** and **dnsConfig** together can address almost all scenarios, including the scenarios in which an on-premises DNS will be used, multiple DNSs will be cascaded, and DNS configuration options will be modified.

  Example 1: Using Your On-Premises DNS

  *Set **dnsPolicy** to **None** so application's DNS configuration file is generated based on **dnsConfig**.*

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4  # IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
  imagePullSecrets:
    - name: default-secret
```

  Container's DNS configuration file:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

  Example 2: Modifying the ndots Option in the DNS Configuration File to Reduce Invalid DNS Queries

  Set **dnsPolicy** to a value other than **None** so the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated based on **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
```



```
  - name: test
    image: nginx:alpine
  dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
    - name: ndots
      value: "2" # The ndots:5 option in the DNS configuration file generated based on the ClusterFirst policy is changed to ndots:2.
  imagePullSecrets:
    - name: default-secret
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

Example 3: Using Multiple DNSs in Serial Sequence

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx:alpine
  dnsPolicy: ClusterFirst  # Added DNS configuration. The cluster connects to CoreDNS by default.
  dnsConfig:
    nameservers:
    - 10.2.3.4 # IP address of your on-premises DNS
  imagePullSecrets:
    - name: default-secret
```

NOTE

A maximum of three DNS addresses can be configured for a nameserver in the container DNS configuration file.

- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS**, you can add two custom DNS addresses in addition to the CoreDNS address. Excess DNS addresses are invalid.
- If **dnsPolicy** is set to **ClusterFirst** and the cluster uses **CoreDNS** and **NodeLocal DNSCache**, you can add one custom DNS address in addition to the CoreDNS and NodeLocal DNSCache addresses. Excess DNS addresses are invalid.

Container's DNS configuration file:

```
nameserver 10.247.3.10 10.2.3.4
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

7.5.3 Using CoreDNS for Custom Domain Name Resolution

Challenges

When using CCE, you may need to resolve custom internal domain names in the following scenarios:

- In the legacy code, a fixed domain name is configured for calling other internal services. If the system decides to use Kubernetes Services, the code refactoring workload could be heavy.
- A service is created outside the cluster. Data in the cluster needs to be sent to the service through a fixed domain name.

Solution

There are several CoreDNS-based solutions for custom domain name resolution:

- **Configuring the Stub Domain for CoreDNS**: You can configure a DNS server for a specific domain name and perform this operation on the console.
- **Using the CoreDNS Hosts plug-in to configure resolution for any domain name**: You can add any record set for a specific domain name, which is similar to adding a record set in the local **/etc/hosts** file.
- **Using the CoreDNS Rewrite plug-in to point a domain name to ClusterIP Services**: Perform CNAME resolution on a domain name in the cluster and redirect it to another domain name in the cluster. This is like giving an alias to the Kubernetes Service, and you do not have to get the IP address of the record set beforehand.
- **Using the CoreDNS Forward plug-in to set the on-premises DNS as the upstream DNS**: Use the on-premises DNS as the external DNS server. You can manage a large number of records on the on-premises DNS, without modifying the CoreDNS configuration when adding or deleting records

Precautions

Improper modification on CoreDNS configuration may cause domain name resolution failures in the cluster. Perform tests before and after the modification.

Configuring the Stub Domain for CoreDNS

Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works.

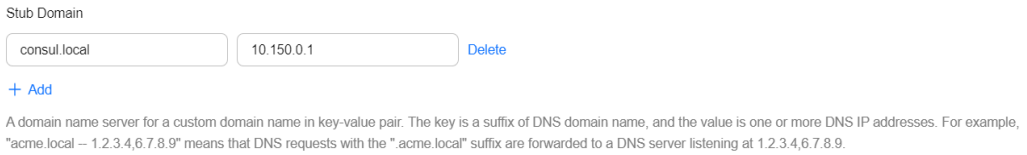
Assume that a cluster administrator has a Consul DNS server located at 10.150.0.1 and all Consul domain names have the suffix **.consul.local**.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.

Step 3 Add a stub domain in the **Parameters** area. The format is a key-value pair. The key is a DNS suffix domain name, and the value is a DNS IP address or a group of DNS IP addresses, for example, **consul.local -- 10.150.0.1**.

Figure 7-89 Adding a stub domain



Corefile:

```
.:5353 {
    bind {$POD_IP}
    cache 30 {
        servfail 5s
    }
    errors
```



```
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
consul.local:5353 {
    bind {$POD_IP}
    errors
    cache 30
    forward . 10.150.0.1
}
```

Step 4 Click **OK**.

Step 5 Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

Modifying the CoreDNS Hosts Configuration File

After modifying the hosts file in CoreDNS, you do not need to configure the hosts file in each pod to add resolution records.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.

Step 3 Edit extended parameters in **Parameters** and add the following content to the **plugins** field:

```
{
  "configBlock": "192.168.1.1 www.example.com\nfallthrough",
  "name": "hosts"
}
```

NOTICE

The **fallthrough** field must be configured. **fallthrough** indicates that when the domain name to be resolved cannot be found in the hosts file, the resolution task is transferred to the next add-on of CoreDNS. If **fallthrough** is not specified, the task ends and the domain name resolution stops. As a result, the domain name resolution in the cluster fails.

For details about how to configure the hosts file, visit **https://coredns.io/plugins/hosts/**.

Figure 7-90 Modifying the CoreDNS hosts configuration

Corefile:

```
.:5353 {
    bind {$POD_IP}
    hosts {
      192.168.1.1 www.example.com
      fallthrough
    }
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

Step 4 Click **OK**.

Step 5 Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

Adding the CoreDNS Rewrite Configuration to Point the Domain Name to ClusterIP Services

Use the Rewrite plug-in of CoreDNS to resolve a specified domain name to the domain name of a Service. For example, the request for accessing the example.com domain name is redirected to the example.default.svc.cluster.local domain name, that is, the example service in the default namespace.

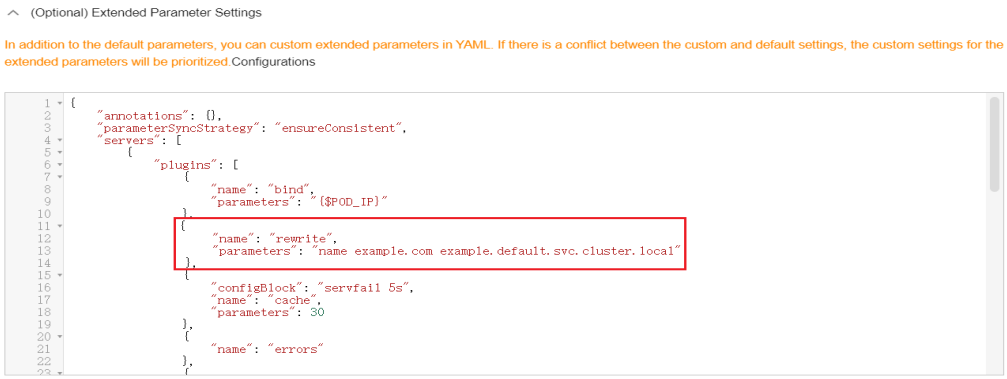
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.

Step 3 Edit extended parameters in **Parameters** and add the following content to the **plugins** field:


```
{
  "name": "rewrite",
  "parameters": "name example.com example.default.svc.cluster.local"
}
```

Figure 7-91 Adding the CoreDNS rewrite configuration to point the domain name to ClusterIP Services



Corefile:

```
.:5353 {
    bind {$POD_IP}
    rewrite name example.com example.default.svc.cluster.local
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

Step 4 Click **OK**.

Step 5 Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

Using CoreDNS to Cascade On-premises DNS

If the domain name is not in the Kubernetes domain, CoreDNS will use the **/etc/resolv.conf** file of the node for resolution by default. You can also change the resolution address to that of the external DNS.

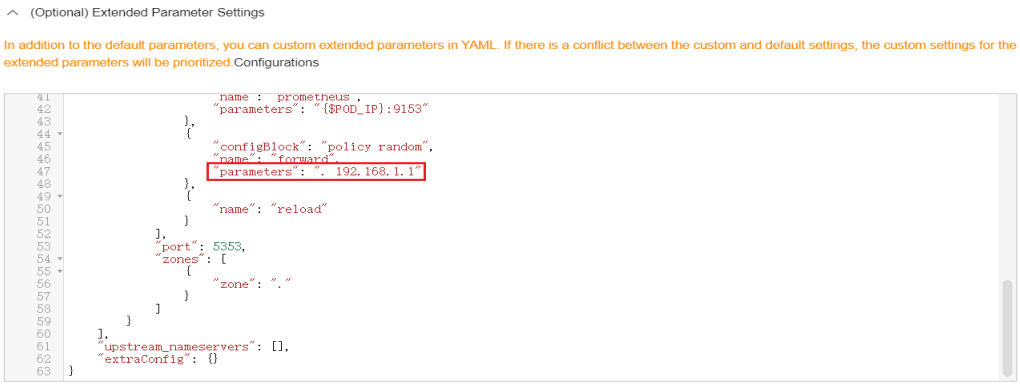
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.

Step 3 Edit extended parameters in **Parameters** and modify the following content in the **plugins** field:


```
{
    "configBlock": "policy random",
    "name": "forward",
    "parameters": ". 192.168.1.1"
}
```

Figure 7-92 Using CoreDNS to cascade on-premises DNS



Corefile:

```
.:5353 {
    bind {$POD_IP}
    cache 30
    errors
    health {$POD_IP}:8080
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    loadbalance round_robin
    prometheus {$POD_IP}:9153
    forward . 192.168.1.1 {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
```

Step 4 Click **OK**.

Step 5 Choose **ConfigMaps and Secrets** in the navigation pane, select the **kube-system** namespace, and view the ConfigMap data of **coredns** to check whether the update is successful.

----End

7.5.4 Using NodeLocal DNSCache to Improve DNS Performance

Challenges

When the number of DNS requests in a cluster increases, the load of CoreDNS increases and the following issues may occur:

- Increased delay: CoreDNS needs to process more requests, which may slow down the DNS query and affect service performance.

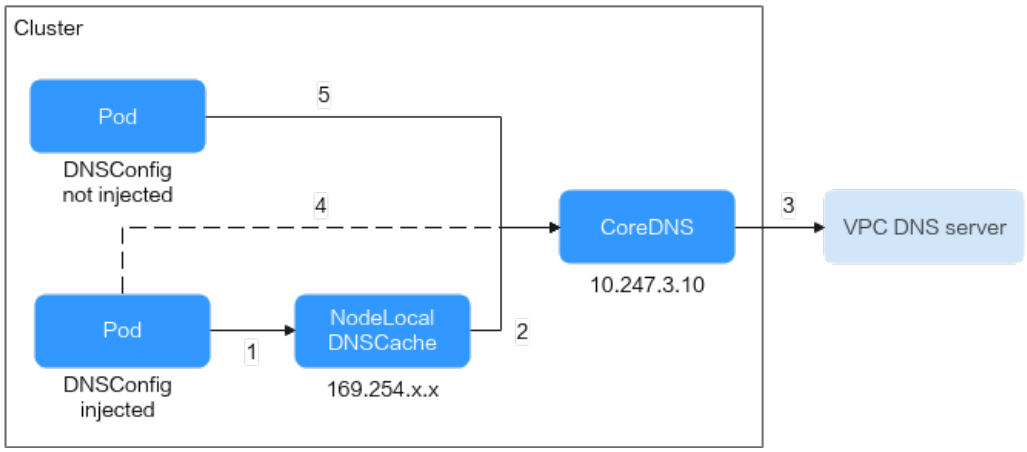
- Increased resource usage: To ensure DNS performance, CoreDNS requires higher specifications.

Solution

To minimize the impact of DNS delay, deploy NodeLocal DNSCache in the cluster to improve the networking stability and performance. NodeLocal DNSCache runs a DNS cache proxy on cluster nodes. All pods with DNS configurations use the DNS cache proxy running on nodes instead of the CoreDNS service for domain name resolution. This reduces CoreDNS's load and improves the cluster DNS performance.

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

Figure 7-93 NodeLocal DNSCache query path



The resolution rules are as follows:

- 1. By default, the pods with DNSConfig injected use NodeLocal DNSCache to resolve requested domain names.
- 2. If NodeLocal DNSCache cannot resolve domain names, it will ask CoreDNS for resolution.
- 3. CoreDNS uses the DNS server in the VPC to resolve the domain names out of the cluster.
- 4. If a pod with DNSConfig injected cannot access NodeLocal DNSCache, CoreDNS will resolve the domain name.
- 5. By default, CoreDNS resolves domain names for the pods without DNSConfig injected.

Notes and Constraints

- Only clusters of version 1.19 or later support the **NodeLocal DNSCache** add-on.
- CCI does not support NodeLocal DNSCache. When the workload is dynamically scaled to CCI, pods on CCI cannot access NodeLocal DNSCache. As a result, domain name resolution times out. Therefore, when creating a workload that is scaled to CCI, add the **node-local-dns-injection: disabled**

label to the pod to prevent automatic injection of DNSConfig. For details, see **Common Issues**.

- The **node-local-dns-injection** label is the system label used by NodeLocal DNSCache. Use this label only to **prevent an automatic DNSConfig injection**.

Installing the Add-on

CCE provides add-on **NodeLocal DNSCache** that simplifies the installation of NodeLocal DNSCache.

NOTE

NodeLocal DNSCache serves as a transparent caching proxy for CoreDNS and does not provide plug-ins such as hosts or rewrite. If you want to enable these plug-ins, modify the CoreDNS configurations.

Step 1 (Optional) Modify the CoreDNS configuration so that the CoreDNS preferentially uses UDP to communicate with the upstream DNS server.

The NodeLocal DNSCache uses TCP to communicate with the CoreDNS. The CoreDNS communicates with the upstream DNS server based on the protocol used by the request source. However, the cloud server does not support TCP. To use NodeLocal DNSCache, modify the CoreDNS configuration so that UDP is preferentially used to communicate with the upstream DNS server, preventing resolution exceptions.

Perform the following operations. In the forward add-on, specify **prefer_udp** as the protocol used by requests. After the modification, CoreDNS preferentially uses UDP to communicate with the upstream system.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Add-ons**. Then, click **Edit** under **CoreDNS**.
3. Edit the advanced configuration under **Parameters** and modify the following content in the **plugins** field:

```
{
    "configBlock": "prefer_udp",
    "name": "forward",
    "parameters": ". /etc/resolv.conf"
}
```

Corefile:

```
Corefile: |-
  .:5353 {
      bind {$POD_IP}
      cache 30 {
          servfail 5s
      }
      errors
      health {$POD_IP}:8080
      kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          fallthrough in-addr.arpa ip6.arpa
      }
      loadbalance round_robin
      prometheus {$POD_IP}:9153
      forward . /etc/resolv.conf {
          prefer_udp
      }
```



```
    reload
    ready {$POD_IP}:8081
}
```

Step 2 Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane, locate **NodeLocal DNSCache** on the right, and click **Install**.

Step 3 On the **Install Add-on** page, select the add-on specifications and set related parameters.

- **DNSConfig**: After this function is enabled, a DNSConfig admission controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-local-dns-injection=enabled** based on admission webhooks and automatically configures DNSConfig for pods. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.

  After automatic injection is enabled, you can customize the following configuration items for DNSConfig (supported when the add-on version is 1.6.7 or later):

  NOTE

  If DNSConfig has been configured in the pod when automatic injection is enabled, DNSConfig in the pod will be used first.

  - (Optional) **IP Address of DNS Server**: IP address list of the DNS server obtained when the container resolves the domain name. NodeLocal DNSCache and CoreDNS IP addresses are added by default. You have the option to add an additional IP address, but duplicates will be removed.
  - (Optional) **Search Domain**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. You can add up to three extra search domains, but any duplicates will be removed.
  - (Optional) **ndots**: specifies that if a domain name has fewer periods (.) than the specified value of **ndots**, it will be combined with the search domain list for DNS query. If the domain name still cannot be resolved, it will be used for DNS query. The system will perform a DNS query on a domain name if the number of periods (.) in it is greater than or equal to the value of **ndots**. If the domain name cannot be resolved correctly, the system will sequentially combine it with the search domain list and then perform a DNS query.

- **Target Namespace**: This parameter is available after **DNSConfig** is enabled. Only NodeLocal DNSCache of v1.3.0 or later supports this function.
  - **Enable All**: CCE adds the **node-local-dns-injection=enabled** label to all created namespaces excluding built-in ones (such as **kube-system**), identifies namespace creation requests, and automatically adds the label to newly created namespaces.
  - **Manual Configure**: You must manually add the **node-local-dns-injection=enabled** label to the namespaces requiring the injection of DNSConfig. For details, see **Managing Namespace Labels**.

Step 4 Click **Install**.

----End

Using NodeLocal DNSCache

By default, application requests are sent through the CoreDNS proxy. To use node-local-dns as the DNS cache proxy, use any of the following methods:

- Auto injection: Automatically configure the **dnsConfig** field of the pod when creating the pod. (This function is not available for pods in system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

Auto Injection

To enable auto injection, perform the following operations:

Step 1 Install the NodeLocal DNSCache add-on and enable auto DNSConfig injection. For details, see **Installing the Add-on**.

Step 2 Use kubectl to access the cluster and add the **node-local-dns-injection=enabled** label to the namespace. For example, run the following command to add the label to the **default** namespace:

```
kubectl label namespace default  node-local-dns-injection=enabled
```

Step 3 Create a workload in the namespace with auto injection enabled. For details, see **Creating a Deployment**.

NOTICE

The pods for which auto injection is enabled must meet the following requirements:

- The new pod does not run in system namespaces such as kube-system and kube-public.
- The **node-local-dns-injection=disabled** label for disabling DNS injection is not added to the new pod.
- The new pod's **DNSPolicy** is **ClusterFirstWithHostNet**. Alternatively, the pod does not use the host network and **DNSPolicy** is **ClusterFirst**.

Step 4 Check the pod configuration.

```
kubectl get pod <pod_name> -oyaml
```

After auto injection is enabled, the following **dnsConfig** settings will be automatically added to the created pod. In addition to the NodeLocal DNSCache IP address 169.254.20.10, the CoreDNS IP address 10.247.3.10 is added to **nameservers**, ensuring high availability of the service DNS server.

```
...
  dnsConfig:
    nameservers:
      - 169.254.20.10
      - 10.247.3.10
    searches:
      - default.svc.cluster.local
```



```
    - svc.cluster.local
    - cluster.local
  options:
    - name: timeout
      value: ''
    - name: ndots
      value: '5'
    - name: single-request-reopen
...
```

----End

Manual Configuration

Manually add the **dnsConfig** settings to pods.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a pod and add the NodeLocal DNSCache IP address 169.254.20.10 to **nameservers** in **dnsConfig**.

NOTE

The NodeLocal DNSCache addresses of different cluster types are as follows:

- CCE standard cluster: 169.254.20.10
- CCE Turbo cluster: 169.254.1.1

Create a **nginx.yaml** file. The following shows an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
  dnsConfig:
    nameservers:
    - 169.254.20.10
    - 10.247.3.10
    searches:
    - default.svc.cluster.local
    - svc.cluster.local
    - cluster.local
    options:
    - name: ndots
      value: '2'
  imagePullSecrets:
  - name: default-secret
```

Step 3 Run the following command to create a pod:

```
kubectl create -f nginx.yaml
```

----End

Common Issues

- How Do I Prevent Auto DNSConfig Injection on a Specific Pod?

  Solution:

  Add **node-local-dns-injection: disabled** to the **labels** field in the pod template. Example:


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
        node-local-dns-injection: disabled   # Prevent auto DNSConfig injection.
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
      imagePullSecrets:
        - name: default-secret
```

7.6 Cluster Network Settings

7.6.1 Adding a Secondary VPC CIDR Block for a Cluster

Scenario

When creating a cluster, deploy it in a VPC. If the planned VPC is too small and IP addresses are insufficient, you can use a secondary VPC CIDR block to support your service scaling. This section describes how to add a secondary VPC CIDR block for your cluster.

Notes and Constraints

Only CCE standard clusters and CCE Turbo clusters of v1.21 and later are supported.

Planning a Secondary CIDR Block

Before adding a secondary CIDR block, plan it properly to prevent CIDR conflicts. Note the following points:

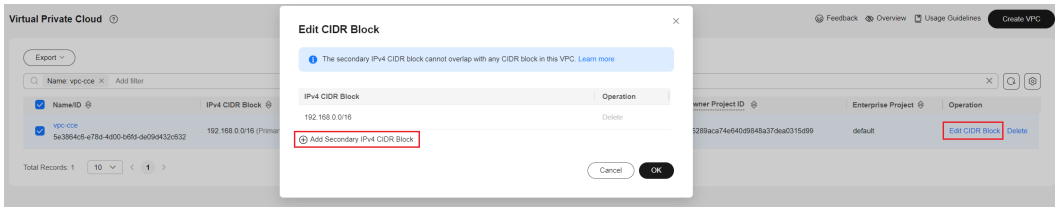
1. All subnets (including extended subnets) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.
2. CIDR blocks 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 may conflict with the IP addresses allocated to the cluster master nodes. Do not select them as secondary CIDR blocks.
3. If an ECS that is not in a cluster in the same VPC needs to access the cluster, Secure Network Address Translation (SNAT) is performed. The pod source address is the node IP address instead of the pod IP address.
4. ECSs in a secondary CIDR block cannot access pods in the cluster unless this CIDR block has been used to add nodes in the cluster.

Procedure

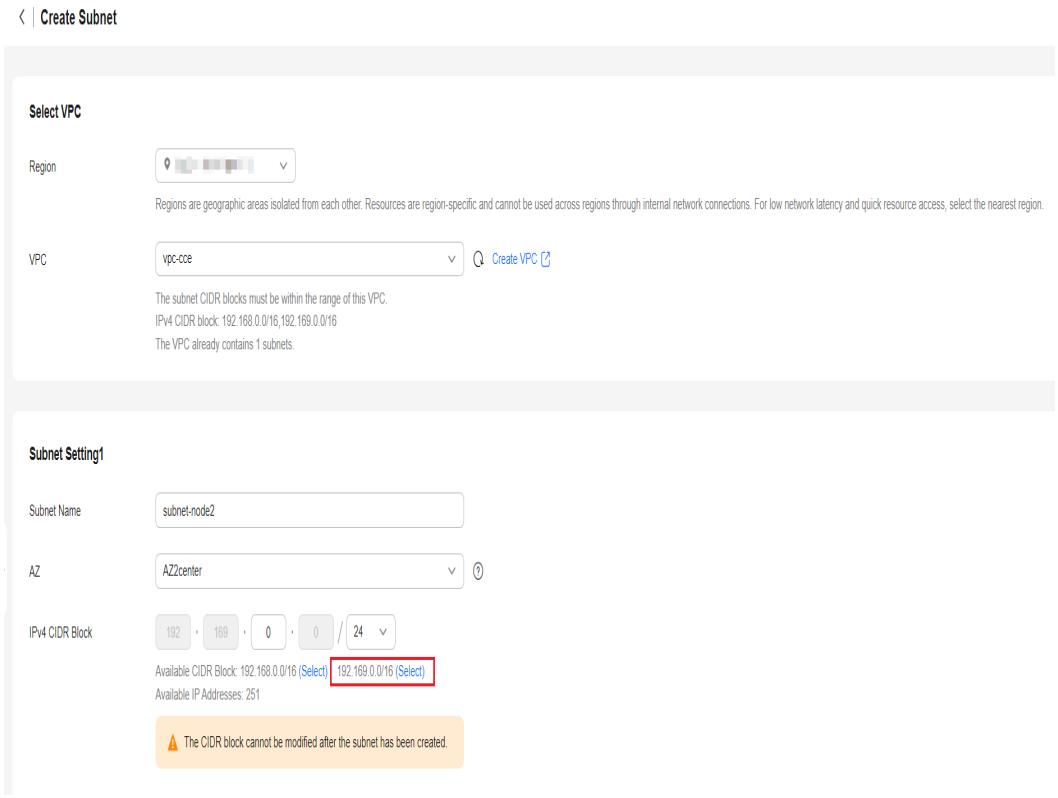
Step 1 Log in to the VPC console. In the navigation pane, choose **Virtual Private Cloud** > **My VPCs**. In the **Operation** column of the VPC to which the cluster belongs, click **Edit CIDR Block** and then **Add Secondary IPv4 CIDR Block**.

Enter the secondary CIDR block to be added, for example, 192.169.0.0/16.

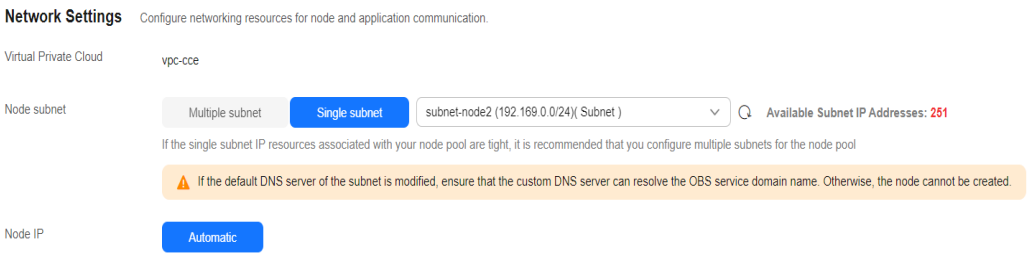
Figure 7-94 Adding a secondary IPv4 CIDR block



Step 2 In the navigation pane, choose **Virtual Private Cloud** > **Subnets**. Click **Create Subnet**. In **IPv4 CIDR Block**, enter the newly added secondary IPv4 CIDR block. Configure other parameters as planned, and click **OK**. Then, you can create subnets in the secondary IPv4 CIDR block for the cluster.



Step 3 After a subnet is created using the secondary IPv4 CIDR block, select the subnet when creating a node or node pool in the **Network Settings** area.



----End

7.7 Configuring Intra-VPC Access

This section describes how to access an intranet from a container (outside the cluster in a VPC), including intra-VPC access and cross-VPC access.

Intra-VPC Access

The performance of accessing an intranet from a container varies depending on the container network models of a cluster.

- **Container tunnel network**

  The container tunnel network encapsulates network data packets through tunnels based on the node network. A container can access other resources in the same VPC as long as the node can access the resources. If the access fails, check whether the security group of peer resources allows the access from the node where the container is located.

- **Cloud Native Network 2.0**

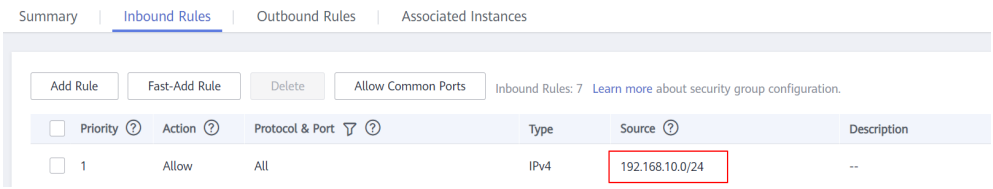
  In the Cloud Native Network 2.0 model, a container is assigned an IP address from the CIDR block of a VPC. The container CIDR block is the subnet of the VPC where the node is located. The container can naturally communicate with other addresses in the VPC. If the access fails, check whether the security group of peer resources allows the access from the container CIDR block.

- **VPC network**

  The VPC network model uses VPC routes to forward container traffic. The container CIDR block and the node VPC are not in the same CIDR block. When a container accesses other resources in the same VPC, **the security group of the peer resource must allow access of the container CIDR block**.

  For example, the CIDR block where the cluster node resides is 192.168.10.0/24, and the container CIDR block is 172.16.0.0/16.

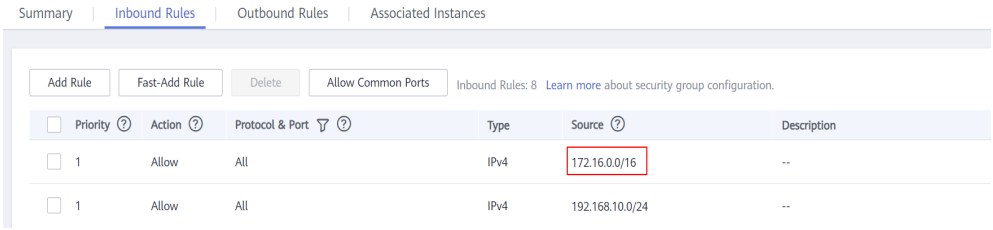
  There is an ECS whose IP address is 192.168.10.52 in the VPC (outside the cluster). The security group of the ECS allows access of only the CIDR block of the cluster node.

  In this case, 192.168.10.52 cannot be pinged from the container.


```
kubectl exec test01-6cbbf97b78-krj6h  -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
^C
--- 192.168.10.25 ping statistics ---
104 packets transmitted, 0 packets received, 100% packet loss
```

Configure the security group to allow access from the container CIDR block 172.16.0.0/16.



In this case, 192.168.10.52 can be pinged from the container.

```
$ kubectl exec test01-6cbbf97b78-krj6h  -it -- /bin/sh
/ # ping 192.168.10.25
PING 192.168.10.25 (192.168.10.25): 56 data bytes
64 bytes from 192.168.10.25: seq=0 ttl=64 time=1.412 ms
64 bytes from 192.168.10.25: seq=1 ttl=64 time=1.400 ms
64 bytes from 192.168.10.25: seq=2 ttl=64 time=1.299 ms
64 bytes from 192.168.10.25: seq=3 ttl=64 time=1.283 ms
^C
--- 192.168.10.25 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

Cross-VPC Access

Cross-VPC access is implemented by establishing a peering connection between VPCs.

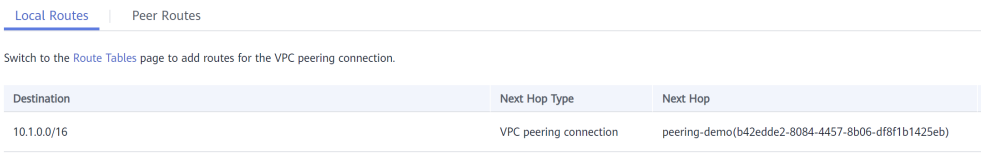
- In the container tunnel network model, a container can access the peer VPC only when the communication is enabled between the node network and the peer VPC.
- Cloud Native Network 2.0 is similar to the container tunnel network. You only need to enable the communication between the subnet where the container is located and the peer VPC.
- Each VPC network has an independent container CIDR block. In addition to the VPC CIDR block, the container CIDR block also needs to be connected.

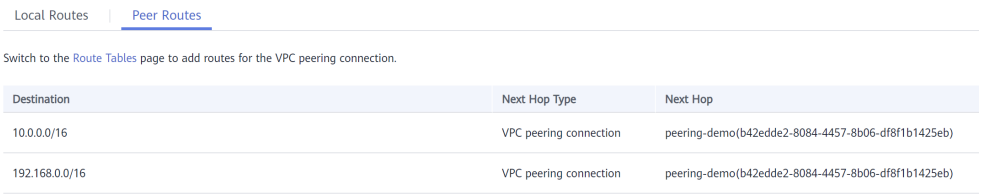
  Assume that there are two VPCs.

  – vpc-demo: Its CIDR block is 192.168.0.0/16, the cluster is in vpc-demo, and the container CIDR block is 10.0.0.0/16.

  – vpc-demo2: Its CIDR block is 10.1.0.0/16.

  Create a peering connection named **peering-demo** (the local VPC is vpc-demo and the peer VPC is vpc-demo2). Add the container CIDR block to the route of the peer VPC.





After this configuration, you can access the container CIDR block 10.0.0.0/16 in vpc-demo2. During the access, pay attention to the security group configuration and enable the port configuration.

Accessing Other Cloud Services

Common services that communicate with CCE through an intranet include RDS, DCS, Kafka, ModelArts, and RabbitMQ.

In addition to the network configurations described in **Intra-VPC Access** and **Cross-VPC Access**, you also need to check **whether these cloud services allow external access**. For example, the DCS Redis instance can be accessed only by the IP addresses in its trustlist. Generally, these cloud services can be accessed by IP addresses in the same VPC. However, the container CIDR block in the VPC network model is different from the CIDR block of the VPC. Therefore, you must add the container CIDR block to the trustlist.

What If a Container Fails to Access an Intranet?

If an intranet cannot be accessed from a container, perform the following operations:

1. View the security group rule of the peer server to check whether the container is allowed to access the peer server.
   - The container tunnel network model needs to allow the IP address of the node where the container is located.
   - The VPC network model needs to allow the container CIDR block.
   - The Cloud Native Network 2.0 model needs to allow the subnet where the container is located.
2. Check whether a trustlist is configured for the peer server. For example, the DCS Redis instance can be accessed only by the IP addresses in its trustlist. Add the container and node CIDR blocks to the trustlist.
3. Check whether the container engine is installed on the peer server and whether it conflicts with the container CIDR block in CCE. If a network conflict occurs, the access fails.

7.8 Accessing the Internet from a Container

Containers can access the Internet in either of the following ways:

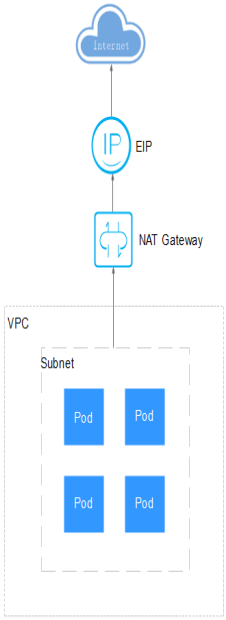
- Bind an EIP to the node where the container is located if the network model is VPC or tunnel.
- Bind an EIP to the pod. (This function applies only to Cloud Native 2.0 clusters. To do so, manually bind an EIP to the ENI or sub-ENI of the pod on the VPC console. This method is not recommended because the IP address of

a pod changes after the pod is rescheduled. As a result, the new pod cannot access the Internet.)

- Configure SNAT rules through NAT Gateway.

You can use NAT Gateway to enable container pods in a VPC to access the Internet. NAT Gateway provides source network address translation (SNAT), which translates private IP addresses to a public IP address by binding an elastic IP address (EIP) to the gateway, providing secure and efficient access to the Internet. **Figure 7-95** shows the SNAT architecture. The SNAT function allows the container pods in a VPC to access the Internet without being bound to an EIP. SNAT supports a large number of concurrent connections, which makes it suitable for applications involving a large number of requests and connections.

Figure 7-95 SNAT



To enable a container to access the Internet, perform the following steps:

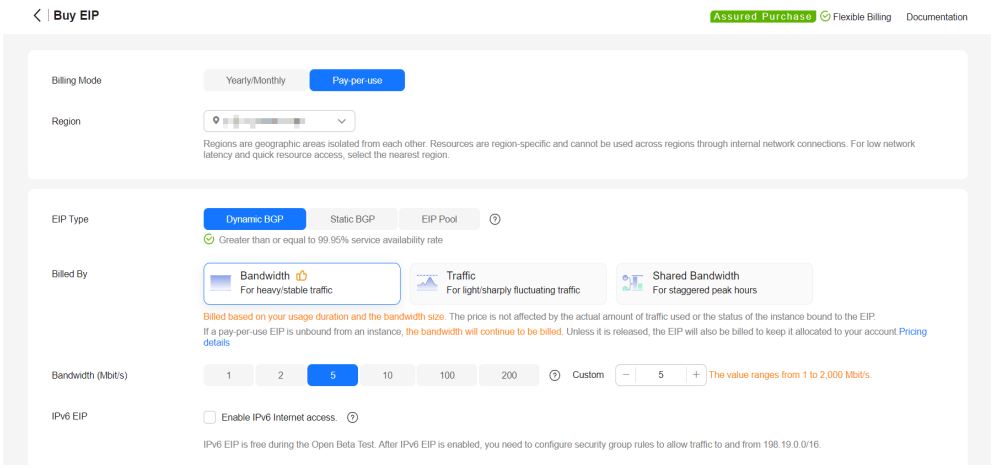
Step 1 Obtain an EIP. For details, see **Assigning an EIP**

1. Log in to the management console.
2. Click in the upper left corner of the management console and select a region and a project.
3. Click in the upper left corner and choose **Networking** > **Elastic IP** in the expanded list.
4. On the **EIPs** page, click **Buy EIP**.
5. Configure parameters as required.

NOTE

Set **Region** to the region where container pods are located.

Figure 7-96 Buying an EIP



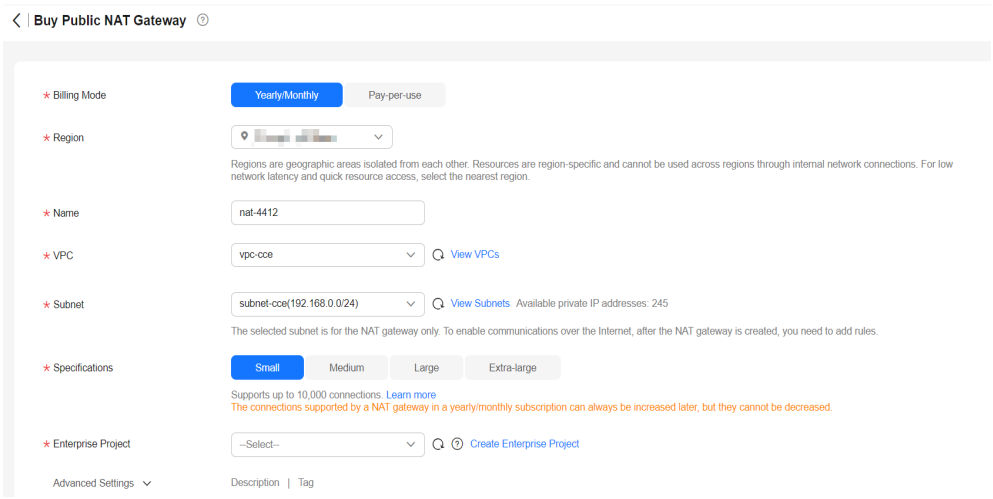
Step 2 Create a NAT gateway. For details, see **Using a Public NAT Gateway to Enable Servers to Share One or More EIPs to Access the Internet**.

1. Click ☰ in the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
2. On the **Public Network Gateways** page, click **Buy Public NAT Gateway** in the upper right corner.
3. Configure parameters as required.

   NOTE

   Select the same VPC.

Figure 7-97 Buying a NAT gateway



Step 3 Configure an SNAT rule and bind the EIP to the subnet. For details, see **Using a Public NAT Gateway to Enable Servers to Share One or More EIPs to Access the Internet**.

1. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
2. On the **SNAT Rules** tab page, click **Add SNAT Rule**.

3. Configure parameters as required.

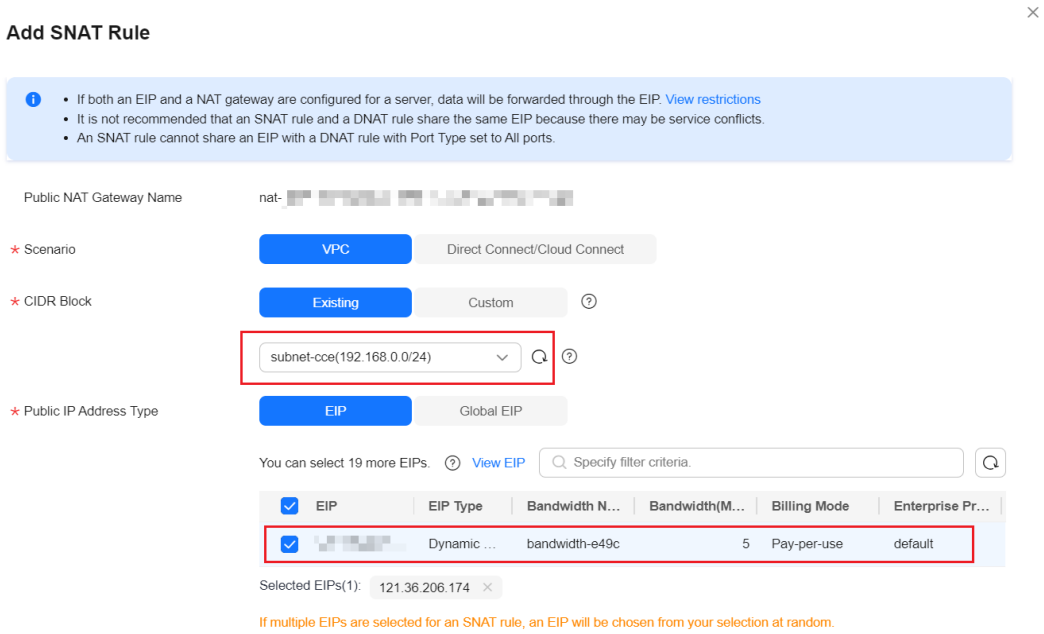
NOTE

SNAT rules take effect by CIDR block. As different container network models use different communication modes, the subnet needs to be selected according to the following rules:

- Tunnel network and VPC network: Select the subnet where the node is located, that is, the subnet selected during node creation.

If there are multiple CIDR blocks, you can create multiple SNAT rules or customize a CIDR block as long as the CIDR block contains the node subnet.

Figure 7-98 Adding an SNAT rule



After the SNAT rule is configured, workloads can access the Internet from the container. The Internet can be pinged from the container.

----End

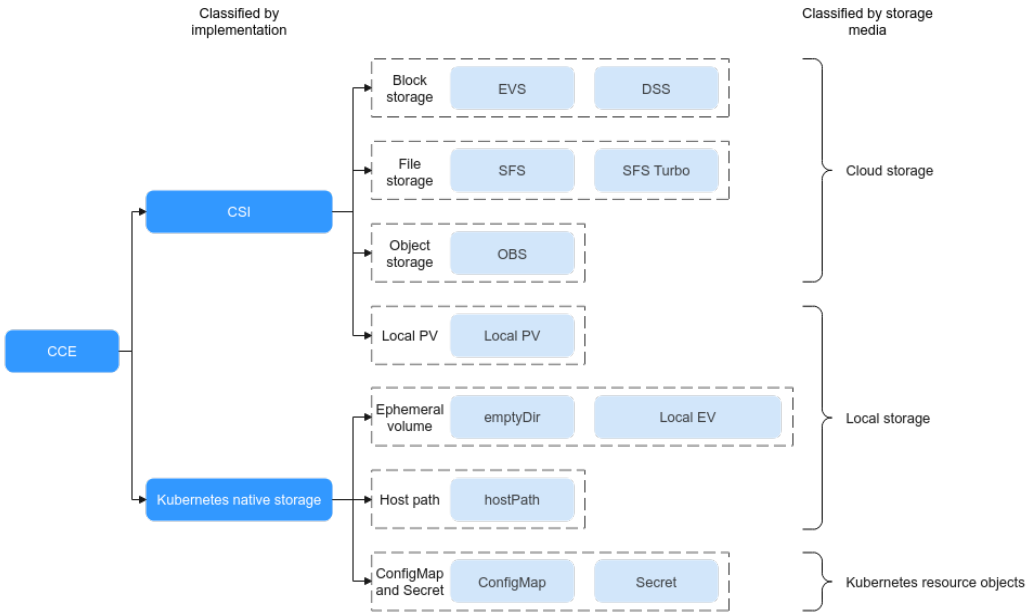
8 Storage

8.1 Overview

Container Storage

CCE container storage is implemented based on Kubernetes container storage APIs (**CSI**). CCE integrates multiple types of cloud storage and covers different application scenarios. CCE is fully compatible with Kubernetes native storage services, such as emptyDir, hostPath, secret, and ConfigMap.

Figure 8-1 Container storage types



CCE allows workload pods to use multiple types of storage:

- In terms of implementation, storage supports Container Storage Interface (CSI) and Kubernetes native storage.

Type	Description
CSI	An **out-of-tree** volume add-on, which specifies the standard container storage API and allows storage vendors to use standard custom storage plugins that are mounted using PVCs and PVs without the need to add their plugin source code to the Kubernetes repository for unified build, compilation, and release. CSI is a recommended in Kubernetes 1.13 and later versions.
Kubernetes native storage	An "in-tree" volume add-on that is built, compiled, and released with the Kubernetes repository.

- In terms of storage media, storage can be classified as cloud storage, local storage, and Kubernetes resource objects.

Type	Description	Application Scenario
Cloud storage	The storage media is provided by storage vendors. Storage volumes of this type are mounted using PVCs and PVs.	Data requires high availability or needs to be shared, for example, logs and media resources. Select a proper cloud storage type based on the application scenario. For details, see **Cloud Storage Comparison**.
Local storage	The storage media is the local data disk or memory of the node. The local PV is a customized storage type provided by CCE and mounted using PVCs and PVs through the CSI. Other storage types are Kubernetes native storage.	Non-HA data requires high I/O and low latency. Select a proper local storage type based on the application scenario. For details, see **Local Storage Comparison**.
Kubernetes resource objects	ConfigMaps and secrets are resources created in clusters. They are special storage types and are provided by tmpfs (RAM-based file system) on the Kubernetes API server.	ConfigMaps are used to inject configuration data to pods. Secrets are used to transmit sensitive information such as passwords to pods.

Cloud Storage Comparison

Item	EVS	SFS	SFS Turbo	OBS	DSS
Definition	EVS offers scalable block storage for cloud servers. With high reliability, high performance, and rich specifications, EVS disks can be used for distributed file systems, dev/test environments, data warehouses, and high-performance computing (HPC) applications.	Expandable to petabytes, SFS provides fully hosted shared file storage, highly available and stable to handle data- and bandwidth-intensive applications in HPC, media processing, file sharing, content management, and web services.	Expandable to 320 TB, SFS Turbo provides fully hosted shared file storage, which is highly available and stable, to support small files and applications requiring low latency and high IOPS. You can use SFS Turbo in high-traffic websites, log storage, compression/decompression, DevOps, enterprise OA, and containerized applications.	Object Storage Service (OBS) provides massive, secure, and cost-effective data storage for you to store data of any type and size. You can use it in enterprise backup/archiving, video on demand (VoD), video surveillance, and many other scenarios.	Dedicated Distributed Storage Service (DSS) provides dedicated physical storage resources. With data redundancy and cache acceleration technologies, DSS delivers highly reliable, durable, low-latency, and stable storage resources.
Data storage logic	Stores binary data and cannot directly store files. To store files, format the file system first.	Stores files and sorts and displays data in the hierarchy of files and folders.	Stores files and sorts and displays data in the hierarchy of files and folders.	Stores objects. Files directly stored automatically generate the system metadata, which can also be customized by users.	Stores binary data and cannot directly store files. To store files, format the file system first.

Item	EVS	SFS	SFS Turbo	OBS	DSS
Access mode	Accessible only after being mounted to ECSs or BMSs and initialized.	Mounted to ECSs or BMSs using network protocols. A network address must be specified or mapped to a local directory for access.	Supports the Network File System (NFS) protocol (NFSv3 only). You can seamlessly integrate existing applications and tools with SFS Turbo.	Accessible through the Internet or Direct Connect (DC). Specify the bucket address and use transmission protocols such as HTTP or HTTPS.	Accessible only after being mounted to ECSs or BMSs and initialized.
Static storage volumes	Supported. For details, see **Using an Existing EVS Disk Through a Static PV**.	Supported. For details, see **Using an Existing SFS File System Through a Static PV**.	Supported. For details, see **Using an Existing SFS Turbo File System Through a Static PV**.	Supported. For details, see **Using an Existing OBS Bucket Through a Static PV**.	Supported. For details, see **Using DSS Through a Static PV**.
Dynamic storage volumes	Supported. For details, see **Using an EVS Disk Through a Dynamic PV**.	Supported. For details, see **Using an SFS File System Through a Dynamic PV**.	Not supported	Supported. For details, see **Using an OBS Bucket Through a Dynamic PV**.	Supported. For details, see **Using DSS Through a Dynamic PV**.
Features	Non-shared storage. Each volume can be mounted to only one node.	Shared storage featuring high performance and throughput	Shared storage featuring high performance and bandwidth	Shared, user-mode file system	Non-shared storage. Each volume can be mounted to only one node.

Item	EVS	SFS	SFS Turbo	OBS	DSS
Application scenarios	HPC, enterprise core cluster applications, enterprise application systems, and dev/test **NOTE** HPC apps here require high-speed and high-IOPS storage, such as industrial design and energy exploration.	HPC, media processing, content management, web services, big data, and analysis applications **NOTE** HPC apps here require high bandwidth and shared file storage, such as gene sequencing and image rendering.	High-traffic websites, log storage, DevOps, and enterprise OA	Big data analytics, static website hosting, online video on demand (VoD), gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks)	• Hybrid deployment of multiple applications, including HPC, database, email, office applications, and web applications • HPC • OLAP applications
Capacity	TB	SFS 1.0: PB	General-purpose: TB	EB	TB
Latency	1–2 ms	SFS 1.0: 3–20 ms	General-purpose: 1–5 ms	10 ms	1–3 ms
Max. IOPS	2200–256000, depending on flavors	SFS 1.0: 2000	General-purpose: up to 100,000	Tens of millions	1500–8000, depending on flavors
Bandwidth	MB/s	SFS 1.0: GB/s	General-purpose: up to GB/s	TB/s	MB/s

Local Storage Comparison

Item	Local PV	Local Ephemeral Volume	emptyDir	hostPath
Definition	Node's local disks form a storage pool (VolumeGroup) through LVM. LVM divides them into logical volumes (LVs) and mounts them to pods.	Kubernetes native emptyDir, where node's local disks form a storage pool (VolumeGroup) through LVM. LVs are created as the storage medium of emptyDir and mounted to pods. LVs deliver better performance than the default storage medium of emptyDir.	Kubernetes native emptyDir. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.	Used to mount a file directory of the host where a pod is located to a specified mount point of the pod.
Features	Low-latency, high-I/O, and non-HA PV. Storage volumes are non-shared storage and bound to nodes through labels. Therefore, storage volumes can be mounted only to a single pod.	Local ephemeral volume. The storage space is from local LVs.	Local ephemeral volume. The storage space comes from the local kubelet root directory or memory.	Used to mount files or directories of the host file system. Host directories can be automatically created. Pods can be migrated (not bound to nodes).
Storage volume mounting	Static storage volumes are not supported. **Using a Local PV Through a Dynamic PV** is supported.	For details, see **Using a Local EV**.	For details, see **Using a Temporary Path**.	For details, see **hostPath**.

Item	Local PV	Local Ephemeral Volume	emptyDir	hostPath
Application scenarios	High I/O requirements and built-in HA solutions of applications, for example, deploying MySQL in HA mode.	• Scratch space, such as for a disk-based merge sort • Checkpointing a long computation for recovery from crashes • Saving the files obtained by the content manager container when web server container data is used	• Scratch space, such as for a disk-based merge sort • Checkpointing a long computation for recovery from crashes • Saving the files obtained by the content manager container when web server container data is used	Requiring a node file, for example, if Docker is used, you can use hostPath to mount the **/var/lib/docker** path of the node. **NOTICE** Avoid using hostPath volumes as much as possible, as they are prone to security risks. If hostPath volumes must be used, they can only be applied to files or directories and mounted in read-only mode.

Enterprise Project Support

NOTE

To use this function, the Everest add-on must be upgraded to v1.2.33 or later.

- Automatically creating storage:

  When creating EVS or OBS PVCs using a StorageClass in CCE, you can specify an enterprise project to assign the created storage resources (EVS disks and OBS) to. **This enterprise project can either be the default one or the same one as the cluster belongs to.**

  If no enterprise project is specified, the enterprise project specified in StorageClass will be used by default for creating storage resources.

  - For a custom StorageClass, you can specify an enterprise project in StorageClass. For details, see **Scenario 3: Specifying an Enterprise Project for a StorageClass**. If no enterprise project is specified in StorageClass, the default enterprise project is used.
  - For the csi-disk and csi-obs storage classes provided by CCE, the created storage resources belong to the default enterprise project.

- Use existing storage:

  When you create a PVC using a PV, ensure that **everest.io/enterprise-project-id** specified in the PVC and PV are the same because an enterprise

project has been specified during storage resource creation. Otherwise, the PVC and PV cannot be bound.

Documentation

- Storage Basics
- Elastic Volume Service
- Scalable File Service
- SFS Turbo
- Object Storage Service

8.2 Storage Basics

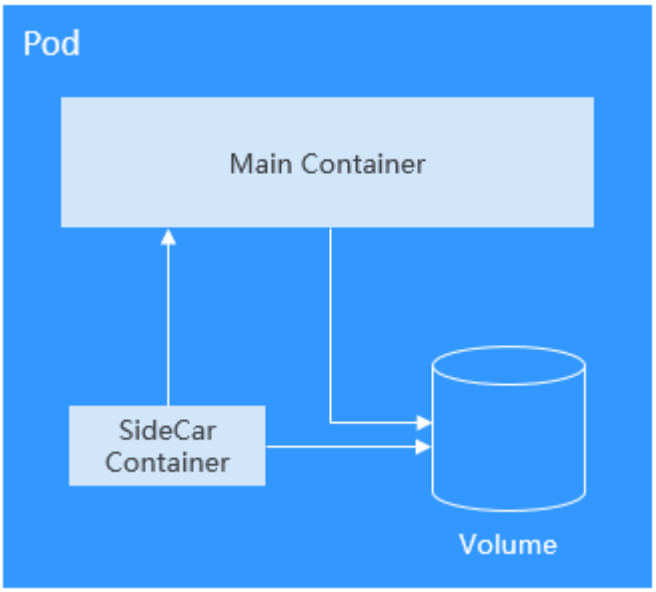
Volumes

On-disk files in a container are ephemeral, which presents the following problems to important applications running in the container:

1. When a container is rebuilt, files in the container will be lost.
2. When multiple containers run in a pod at the same time, files need to be shared among the containers.

Kubernetes volumes resolve both of these problems. Volumes, as part of a pod, cannot be created independently and can only be defined in pods. All containers in a pod can access its volumes, but the volumes must have been mounted to any directory in a container.

The following figure shows how a storage volume is used between containers in a pod.



The basic principles for using volumes are as follows:

- Multiple volumes can be mounted to a pod. However, do not mount too many volumes to a pod.

- Multiple types of volumes can be mounted to a pod.
- Each volume mounted to a pod can be shared among containers in the pod.
- You are advised to use PVCs and PVs to mount volumes for Kubernetes.

NOTE

The lifecycle of a volume is the same as that of the pod to which the volume is mounted. When the pod is deleted, the volume is also deleted. However, files in the volume may outlive the volume, depending on the volume type.

Kubernetes provides various volume types, which can be classified as in-tree and out-of-tree.

Volume Classification	Description
In-tree	Maintained through the Kubernetes code repository and built, edited, and released with Kubernetes binary files. Kubernetes does not accept this volume type anymore. Kubernetes-native volumes such as hostPath, emptyDir, Secret, and ConfigMap are all the in-tree type. PVCs are a special in-tree volume. Kubernetes uses this type of volume to convert from in-tree to out-of-tree. PVCs allow you to request for PVs created using the underlying storage resources provided by different storage vendors.
Out-of-tree	Out-of-tree volumes include container storage interfaces (CSIs) and FlexVolumes (deprecated). Storage vendors only need to comply with certain specifications to create custom storage add-ons and PVs that can be used by Kubernetes, without adding add-on source code to the Kubernetes code repository. Cloud storage such as SFS and OBS is used by installing storage drivers in a cluster. You need to create PVs in the cluster and mount the PVs to pods using PVCs.

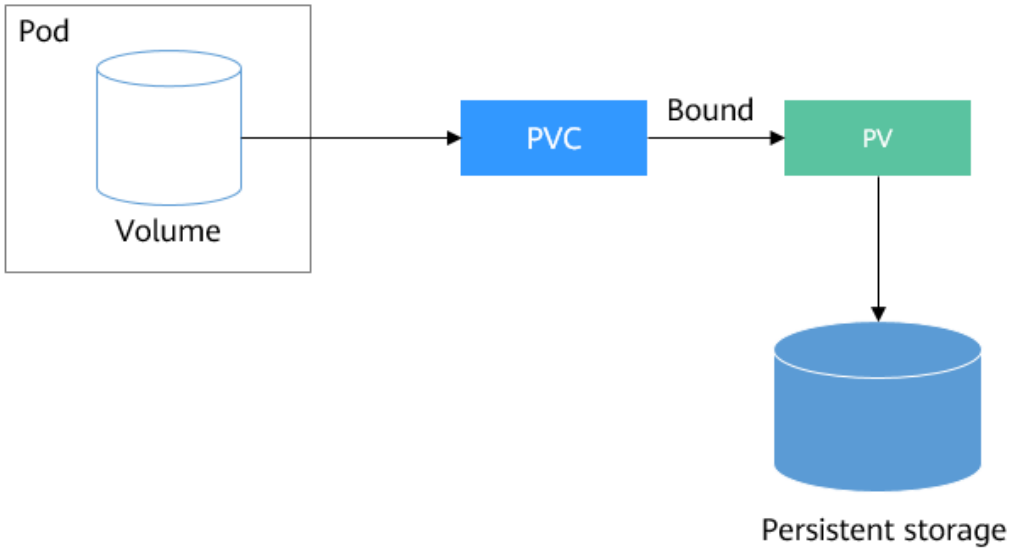
PV and PVC

Kubernetes provides PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) to abstract details of how storage is provided from how it is consumed. You can request specific size of storage when needed, just like pods can request specific levels of resources (CPU and memory).

- PV: describes a persistent storage volume in a cluster. A PV is a cluster-level resource just like a node. It applies to the entire Kubernetes cluster. A PV has a lifecycle independent of any individual Pod that uses the PV.
- PVC: describes a request for storage by a user. When configuring storage for an application, claim a storage request (that is, PVC). Kubernetes selects a PV that best meets the request and binds the PV to the PVC. A PVC to PV binding is a one-to-one mapping. When creating a PVC, describe the attributes of the requested persistent storage, such as the storage size and read/write permission.

You can bind PVCs to PVs in a pod so that the pod can use storage resources. The following figure shows the relationship between PVs and PVCs.

Figure 8-2 PVC-to-PV binding



CSI

CSI is a standard for container storage interfaces and a storage plugin implementation solution recommended by the Kubernetes community. **CCE Container Storage (Everest)** is a storage add-on developed based on CSI. It provides different types of persistent storage for containers.

Volume Access Modes

Storage volumes can be mounted to the host system only in the mode supported by underlying storage resources. For example, a file storage system can be read and written by multiple nodes, but an EVS disk can be read and written by only one node.

- **ReadWriteOnce**: A storage volume can be mounted to a single node in read-write mode.
- **ReadWriteMany**: A storage volume can be mounted to multiple nodes in read-write mode.

Table 8-1 Access modes supported by storage volumes

Volume Type	ReadWriteOnce	ReadWriteMany
EVS	√	x
SFS	x	√
OBS	x	√
SFS Turbo	x	√

Volume Type	ReadWriteOnce	ReadWriteMany
Local PV	√	x
DSS	√	x

Mounting a Storage Volume

You can mount volumes in the following ways:

Use PVs to describe existing storage resources, and then create PVCs to use the storage resources in pods. You can also use the dynamic creation mode. That is, specify the **StorageClass** when creating a PVC and use the provisioner in the StorageClass to automatically create a PV and bind the PV to the PVC.

Table 8-2 Modes of mounting volumes

Mount Mode	Description	Supported Volume Type	Other Constraints
Statically creating storage volume (using existing storage)	Use existing storage (such as EVS disks and SFS file systems) to create PVs and mount the PVs to the workload through PVCs. Kubernetes binds PVCs to the matching PVs so that workloads can access storage services.	All volumes	None
Dynamically creating storage volumes (automatically creating storage)	Specify a **StorageClass** for a PVC. The storage provisioner creates underlying storage media as required to automatically create PVs and directly bind the PV to the PVC.	EVS, DSS, OBS, SFS, and local PV	None
Dynamic mounting (VolumeClaimTemplate)	Achieved by using the **volumeClaimTemplates** field and depends on the dynamic PV creation capability of StorageClass. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name.	EVS, DSS, and local PV	Supported only by StatefulSets

PV Reclaim Policy

A PV reclaim policy is used to delete or reclaim underlying volumes when a PVC is deleted. The value can be **Delete** or **Retain**.

- **Delete**: Deleting a PVC will remove the PV from Kubernetes, and the associated underlying storage assets will also be removed from the external infrastructure.

  NOTE

  Yearly/Monthly resources cannot be deleted using the **Delete** reclaim policy.

- **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.

  You can manually delete and reclaim volumes by performing the following operations:

  a. Delete the PV.
  b. Clear data on the associated underlying storage resources as required.
  c. Delete the associated underlying storage resources.

  To reuse the underlying storage resources, create a PV.

CCE also allows you to delete a PVC without deleting underlying storage resources. This function can be achieved only by using a YAML file: Set the PV reclaim policy to **Delete** and add **everest.io/reclaim-policy: retain-volume-only** to **annotations**. In this way, when the PVC is deleted, the PV is deleted, but the underlying storage resources are retained.

The following YAML file takes EVS as an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/disk-volume-type: SAS
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>      # AZ of the node where the application is to be deployed
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  volumeName: pv-evs-test

---
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only
  name: pv-evs-test
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is
```

```
to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>      # AZ of the node where the application is to be
deployed
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeHandle: 2af98016-6082-4ad6-bedc-1a9c673aef20
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SAS
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```

Documentation

- For more information about Kubernetes storage, see **Storage**.
- For more information about CCE container storage, see **Overview**.

8.3 Elastic Volume Service

8.3.1 Overview

To achieve persistent storage, CCE allows you to mount the storage volumes created from Elastic Volume Service (EVS) disks to a path of a container. When the container is migrated within an AZ, the mounted EVS volumes are also migrated. By using EVS volumes, you can mount the remote file directory of a storage system to a container so that data in the data volume is permanently preserved. Even if the container is deleted, the data in the data volume is still stored in the storage system.

EVS Disk Performance Specifications

EVS performance metrics include:

- IOPS: the number of input/output operations performed by an EVS disk per second
- Throughput: the amount of data read from and written into an EVS disk per second
- I/O latency: the minimum interval between two consecutive I/O operations on an EVS disk

Table 8-3 EVS disk performance specifications

Parameter	Extreme SSD	General Purpose SSD	Ultra-high I/O	High I/O
Max. capacity (GiB)	• System disk: 1,024 • Data disk: 32,768	• System disk: 1,024 • Data disk: 32,768	• System disk: 1,024 • Data disk: 32,768	• System disk: 1,024 • Data disk: 32,768
Max. IOPS	128,000	20,000	50,000	5000
Max. throughput (MiB/s)	1000	250	350	150
Burst IOPS limit	64,000	8000	16,000	5000
Disk IOPS	Min. (128,000, 1800 + 50 x Capacity)	Min. (20,000, 1800 + 12 x Capacity)	Min. (50,000, 1800 + 50 x Capacity)	Min. (5000, 1800 + 8 x Capacity)
Disk throughput (MiB/s)	Min. (1000, 120 + 0.5 x Capacity)	Min. (250, 100 + 0.5 x Capacity)	Min. (350, 120 + 0.5 x Capacity)	Min. (150, 100 + 0.15 x Capacity)
Single-queue access latency (ms)	Sub-millisecond	1	1	1–3
API name	ESSD	GPSSD	SSD	SAS

For details about EVS disks, see **Disk Types and Performance**.

Application Scenarios

EVS disks can be mounted in the following modes based on application scenarios:

- **Using an Existing EVS Disk Through a Static PV**: static creation mode, where you use an existing EVS disk to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an EVS Disk Through a Dynamic PV**: dynamic creation mode, in which you do not need to create EVS volumes beforehand. Instead, specify a StorageClass when creating a PVC. Then, an EVS volume and PV will be created automatically. This mode applies to scenarios where no underlying storage is available.
- **Dynamically Mounting an EVS Disk to a StatefulSet**: available only for StatefulSets. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. This mode applies to StatefulSets with multiple pods.

Billing

- To mount storage volumes of the EVS type, the billing mode of EVS disks **automatically created** by specifying the StorageClass is pay-per-use by default and cannot be changed to yearly/monthly. To use a yearly/monthly-billed EVS disk, **use an existing one**.
- For details about the EVS disk pricing, see **Billing for Disks**.

8.3.2 Using an Existing EVS Disk Through a Static PV

CCE allows you to create a PV using an existing EVS disk. After the PV is created, you can create a PVC and bind it to the PV. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- You have created an EVS disk that meets the following requirements:
  - The EVS disk cannot be a system disk, DSS disk, or shared disk.
  - The EVS disk must be of the **SCSI** type (the default disk type is **VBD** when you purchase an EVS disk).
  - The EVS disk must be available and not used by other resources.
  - The AZ of the EVS disk must be the same as that of the cluster node. Otherwise, the EVS disk cannot be mounted and the pod cannot start.
  - If the EVS disk is encrypted, the key must be available.
  - The EVS disk must be in the default enterprise project or the enterprise project to which the cluster belongs.
  - EVS disks that have been partitioned are not supported.
  - Only ext4 EVS disks are supported.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If an EVS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses EVS disks.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

  For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.

Using an Existing EVS Disk on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **EVS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using an EVS Disk Through a Dynamic PV**. In this example, select **Create new** to create both a PV and PVC on the console.
PV[a]	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in **Related Operations**. You do not need to specify this parameter in this example.
EVS[b]	Click **Select EVS**. On the displayed page, select the EVS volume that meets your requirements and click **OK**.
PV Name[b]	Enter the PV name, which must be unique in the same cluster.
Access Mode[b]	EVS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Reclaim Policy[b]	You can select **Delete** or **Retain** to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see **PV Reclaim Policy**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-4**. For details about other parameters, see **Workloads**.

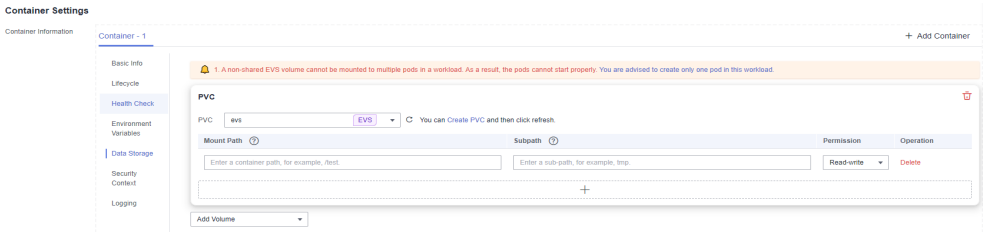
   Table 8-4 Mounting a storage volume

Parameter	Description
PVC	Select an existing EVS volume. An EVS volume can be mounted to only one workload.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.

NOTE

A non-shared EVS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if an EVS disk is attached.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Using an Existing EVS Disk Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV. If a PV has been created in your cluster, skip this step. When using a YAML file to create a PV, do not use parameters that are not declared in the file, such as **status**, **spec.claimRef**, and **annotation.everest.io/set-disk-metadata**. Otherwise, the PV may be abnormal.

1. Create the **pv-evs.yaml** file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only         # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-evs    # PV name
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
spec:
  accessModes:
    - ReadWriteOnce     # Access mode, which must be ReadWriteOnce for EVS disks
  capacity:
    storage: 10Gi       # EVS disk capacity, in the unit of GiB. The value ranges from 1 to 32768.
  csi:
    driver: disk.csi.everest.io     # Dependent storage driver for the mounting
    fsType: ext4     # Must be the same as that of the original file system of the disk.
    volumeHandle: <your_volume_id>   # EVS volume ID
    volumeAttributes:
      everest.io/disk-mode: SCSI          # Device type of the EVS disk. Only SCSI is supported.
      everest.io/disk-volume-type: SAS     # EVS disk type
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
      everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
      everest.io/disk-iops: '3000'      # (Optional) IOPS of only a GPSSD2 EVS disk
      everest.io/disk-throughput: '125' # (Optional) Throughput of only a GPSSD2 EVS disk

      everest.io/csi.charging-mode: prePaid  # Mandatory if the billing mode is set to Yearly/Monthly
      everest.io/csi.period-type: month      # Mandatory if the billing mode is set to Yearly/Monthly. month indicates that resources will be billed on a monthly basis.
      everest.io/csi.period-num: '1'         # Mandatory if the billing mode is set to Yearly/Monthly. '1' indicates that the required duration is one month.
```



```
persistentVolumeReclaimPolicy: Delete    # Reclaim policy
storageClassName: csi-disk               # StorageClass name. The value must be csi-disk for EVS disks.
```

Table 8-5 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Only **retain-volume-only** is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is **Delete**. If the reclaim policy is **Delete** and the current value is **retain-volume-only**, the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
fsType	Yes	File system type, which defaults to **ext4**.
volumeHandle	Yes	Volume ID of the EVS disk. To obtain a volume ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Elastic Volume Service** > **Disks**. Click the name of the target EVS disk to go to its details page. On the **Summary** tab page, click the copy button after **ID**.
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. – **SAS**: high I/O – **SSD**: ultra-high I/O – **GPSSD**: general-purpose SSD – **ESSD**: extreme SSD – **GPSSD2**: general-purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the **everest.io/disk-iops** and **everest.io/disk-throughput** annotations are configured

Parameter	Mandatory	Description
everest.io/disk-iops	No	Preconfigured IOPS, which is supported only by general-purpose SSD v2 EVS disks. – The IOPS of general-purpose SSD v2 EVS disks ranges from 3000 to 128000, and the maximum value is 500 times of the capacity (GiB). If the IOPS of general-purpose SSD v2 disks is greater than 3000, extra IOPS will be billed. For details, see **Price Calculator**.
everest.io/disk-throughput	No	Preconfigured throughput, which is supported only by general-purpose SSD v2 EVS disks. The value ranges from 125 MiB/s to 1000 MiB/s. The maximum value is a quarter of IOPS. If the throughput is greater than 125 MiB/s, extra throughput will be billed. For details, see **Price Calculator**.
everest.io/crypt-key-id	No	Mandatory when the EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. To obtain an encryption key ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Elastic Volume Service** > **Disks**. Click the name of the target EVS disk to go to its details page. On the **Summary** tab page, copy the value of **KMS Key ID** in the **Configuration Information** area.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Elastic Volume Service** > **Disks**. Click the name of the target EVS disk to go to its details page. On the **Summary** tab page, click the enterprise project in **Management Information** to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the EVS disk belongs.

Parameter	Mandatory	Description
everest.io/csi.charging-mode	No	This parameter is mandatory when the yearly/monthly billing mode is used. If this parameter is not specified, the pay-per-use billing mode will be used by default. This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: – **prePaid**: EVS disks are billed on a yearly/monthly basis, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. – **postPaid**: EVS disks are billed on a pay-per-use basis. **NOTE** Before creating an EVS disk billed on a yearly/monthly basis, you need to grant the payment permission (**bss:order:pay**) to **cce_cluster_agency**. Use your tenant account on the CCE console to perform this operation for the first time. After confirmation, CCE will automatically authorize the agency.
everest.io/csi.period-type	No	Billing cycle, which is mandatory only when the yearly/monthly billing mode is used. Options: – **month**: Resources are billed on a monthly basis. – **year**: Resources are billed on a yearly basis.
everest.io/csi.period-num	No	Required duration, which is mandatory only when the yearly/monthly billing mode is used. Options: – For monthly billing, the value of this parameter ranges from **1** to **9**. – For yearly billing, the value of this parameter is **1**.

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If high data security is required, select **Retain** to prevent data from being deleted by mistake. **Delete**: – If **everest.io/reclaim-policy** is not specified, both the PV and EVS disk will be deleted when a PVC is deleted. – If **everest.io/reclaim-policy** is set to **retain-volume-only**, when a PVC is deleted, the PV will be deleted but the EVS disk will be retained. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storageClassName	Yes	StorageClass name, which is **csi-disk** for an EVS disk.

2. Run the following command to create a PV:

```
kubectl apply -f pv-evs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-evs.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # EVS disk type
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
    everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
    everest.io/csi.charging-mode: prePaid  # Mandatory if the billing mode is set to Yearly/Monthly
    everest.io/csi.period-type: month      # Mandatory if the billing mode is set to Yearly/Monthly. month indicates that resources will be billed on a monthly basis.
    everest.io/csi.period-num: '1'         # Mandatory if the billing mode is set to Yearly/Monthly. '1' indicates that the required duration is one month.
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
spec:
  accessModes:
```



```
- ReadWriteOnce              # The value must be ReadWriteOnce for EVS disks.
 resources:
   requests:
     storage: 10Gi           # EVS disk capacity, ranging from 1 to 32768. The value must be the same as the storage size of the existing PV.
 storageClassName: csi-disk  # The StorageClass is EVS.
 volumeName: pv-evs          # PV name
```

Table 8-6 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/csi.charging-mode	No	This parameter is mandatory when the yearly/monthly billing mode is used. If this parameter is not specified, the pay-per-use billing mode will be used by default. This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: – **prePaid**: EVS disks are billed on a yearly/monthly basis, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. – **postPaid**: EVS disks are billed on a pay-per-use basis. **NOTE** Before creating an EVS disk billed on a yearly/monthly basis, you need to grant the payment permission (**bss:order:pay**) to **cce_cluster_agency**. Use your tenant account on the CCE console to perform this operation for the first time. After confirmation, CCE will automatically authorize the agency.

Parameter	Mandatory	Description
everest.io/csi.period-type	No	Billing cycle, which is mandatory only when the yearly/monthly billing mode is used. Options: – **month**: Resources are billed on a monthly basis. – **year**: Resources are billed on a yearly basis.
everest.io/csi.period-num	No	Required duration, which is mandatory only when the yearly/monthly billing mode is used. Options: – For monthly billing, the value of this parameter ranges from **1** to **9**. – For yearly billing, the value of this parameter is **1**.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Yes	PV name, which must be the same as the PV name in 1.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in 1. The StorageClass for EVS volumes is **csi-disk**.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-evs.yaml
```

Step 4 Create an application.

1. Create a file named **web-evs.yaml**. In this example, the EVS volume is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs
  namespace: default
spec:
 replicas: 1          # The number of workload replicas that use the EVS volume must be 1.
 selector:
   matchLabels:
     app: web-evs
 serviceName: web-evs   # Headless Service name
 template:
   metadata:
     labels:
       app: web-evs
   spec:
     containers:
     - name: container-1
       image: nginx:latest
       volumeMounts:
       - name: pvc-disk    # Volume name, which must be the same as the volume name in the
```



```
volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-evs    # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs   # Headless Service name
  namespace: default
  labels:
    app: web-evs
spec:
  selector:
    app: web-evs
  clusterIP: None
  ports:
    - name: web-evs
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f web-evs.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-evs
```

Expected output:

```
web-evs-0               1/1     Running   0             38s
```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

```
kubectl exec web-evs-0 -- df | grep data
```

Expected output:

```
/dev/sdc            10255636     36888  10202364   0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-0 -- ls /data
```

Expected output:

```
lost+found
```

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-evs-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-0 -- ls /data
```

Expected output:


```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-evs-0**:

```
kubectl delete pod web-evs-0
```

Expected output:

```
pod "web-evs-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-7**.

Table 8-7 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Click **Create PersistentVolume** in the upper right corner. In the dialog box displayed, configure parameters. • **Volume Type**: Select **EVS**. • **EVS**: Click **Select EVS**. On the displayed page, select the EVS volume that meets your requirements and click **OK**. • **PV Name**: Enter the PV name, which must be unique in a cluster. • **Access Mode**: EVS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**. • **Reclaim Policy**: **Delete** or **Retain** is supported. For details, see **PV Reclaim Policy**. 2. Click **Create**.

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of an attached EVS disk on the CCE console. Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.3.3 Using an EVS Disk Through a Dynamic PV

CCE allows you to specify a StorageClass to automatically create an EVS disk and the corresponding PV. This function is applicable when no underlying storage volume is available.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- EVS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If an EVS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses EVS disks.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

  For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.
- Resource tags can be added to dynamically created EVS disks. After the EVS disks are created, the resource tags cannot be updated on CCE. To update them, go to the EVS console. If you use an existing EVS disk to create a PV, you also need to add or update resource tags on the EVS console.

Automatically Creating an EVS Disk on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **EVS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If no underlying storage is available, select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode. – If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see **Using an Existing EVS Disk Through a Static PV**. In this example, select **Dynamically provision**.

Parameter	Description
Storage Classes	The default StorageClasses for EVS disks are **csi-disk** and **csi-disk-topology**. **NOTE** If you use the **csi-disk** (EVS) StorageClass, a PVC and PV will be created immediately. The EVS disk is created with the PV, and then the PVC is bound to the PV. If you use the **csi-disk-topology** (EVS created with a delay) StorageClass, a PV will not be immediately created when a PVC is created. Instead, the pods that will be associated with the PVC are scheduled first, and then the EVS disk and PV are created, and finally the PV is bound to the PVC. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. **NOTE** An EVS disk can only be mounted to a node in the same AZ. After an EVS disk is created, its AZ cannot be changed.
Disk Type	Select an EVS disk type. EVS disk types vary depending on regions. Obtain the available EVS types on the console. **NOTE** If the Everest version is 2.4.4 or later, general-purpose SSD V2 EVS disks are supported. General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**.
Capacity (GiB)	Capacity of the requested storage volume.

Parameter	Description
Billing Mode	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: – **Yearly/Monthly** Resources will be billed on a yearly/monthly basis. To use the yearly/monthly billing mode, you need to select the required duration, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. – **Pay-per-use** Resources will be billed based on usage duration. You can provision or delete resources at any time.
Access Mode	EVS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Encryption	Configure whether to encrypt underlying storage. If you select **Enabled (key)**, an encryption key must be configured. Before using encryption, check whether the region where the EVS disk is located supports disk encryption.
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.
Resource Tag	You can add resource tags to classify resources, which is supported only when the Everest version in the cluster is 2.1.39 or later. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=*{Cluster ID}***, **CCE-Cluster-Name=*{Cluster name}***, and **CCE-Namespace=*{Namespace name}***. These tags cannot be modified. **NOTE** After a dynamic PV of the EVS type is created, the resource tags cannot be updated on the CCE console. To update these resource tags, go to the EVS console.

2. Click **Create**.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-8**. For details about other parameters, see **Workloads**.

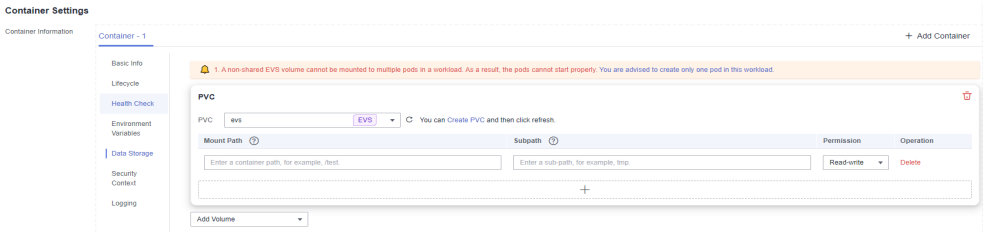
   Table 8-8 Mounting a storage volume

Parameter	Description
PVC	Select an existing EVS volume. An EVS volume can be mounted to only one workload.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.

NOTE

A non-shared EVS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if an EVS disk is attached.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Automatically Creating an EVS Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-evs-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # EVS disk type
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
    everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
    everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # (Optional) Custom resource tags
    everest.io/disk-iops: '3000'      # (Optional) IOPS of only a GPSSD2 EVS disk
    everest.io/disk-throughput: '125' # (Optional) Throughput of only a GPSSD2 EVS disk
    everest.io/csi.charging-mode: prePaid  # Mandatory if the billing mode is set to Yearly/Monthly
    everest.io/csi.period-type: month      # Mandatory if the billing mode is set to Yearly/Monthly. month indicates that resources will be billed on a monthly basis.
    everest.io/csi.period-num: '1'         # Mandatory if the billing mode is set to Yearly/Monthly. '1' indicates that the required duration is one month.
    everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
spec:
  accessModes:
  - ReadWriteOnce              # The value must be ReadWriteOnce for EVS disks.
  resources:
    requests:
      storage: 10Gi            # EVS disk capacity, ranging from 1 to 32768
  storageClassName: csi-disk    # The StorageClass is EVS.
```


Table 8-9 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. – **SAS**: high I/O – **SSD**: ultra-high I/O – **GPSSD**: general-purpose SSD – **ESSD**: extreme SSD – **GPSSD2**: general-purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the **everest.io/disk-iops** and **everest.io/disk-throughput** annotations are configured
everest.io/disk-iops	No	Preconfigured IOPS, which is supported only by general-purpose SSD v2 EVS disks. – The IOPS of general-purpose SSD v2 EVS disks ranges from 3000 to 128000, and the maximum value is 500 times of the capacity (GiB). If the IOPS of general-purpose SSD v2 disks is greater than 3000, extra IOPS will be billed. For details, see **Price Calculator**.
everest.io/disk-throughput	No	Preconfigured throughput, which is supported only by general-purpose SSD v2 EVS disks. The value ranges from 125 MiB/s to 1000 MiB/s. The maximum value is a quarter of IOPS. If the throughput is greater than 125 MiB/s, extra throughput will be billed. For details, see **Price Calculator**.

Parameter	Mandatory	Description
everest.io/crypt-key-id	No	This parameter is mandatory when an EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. You can use a custom key or the default key named **evs/default**. To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
everest.io/disk-volume-tags	No	This field is optional. It is supported when the Everest version in the cluster is 2.1.39 or later. You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}*, and **CCE-Namespace=***{Namespace name}*. These tags cannot be modified.

Parameter	Mandatory	Description
everest.io/csi.charging-mode	No	This parameter is mandatory when the yearly/monthly billing mode is used. If this parameter is not specified, the pay-per-use billing mode will be used by default. This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: – **prePaid**: EVS disks are billed on a yearly/monthly basis, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. – **postPaid**: EVS disks are billed on a pay-per-use basis. **NOTE** Before creating an EVS disk billed on a yearly/monthly basis, you need to grant the payment permission (**bss:order:pay**) to **cce_cluster_agency**. Use your tenant account on the CCE console to perform this operation for the first time. After confirmation, CCE will automatically authorize the agency.
everest.io/csi.period-type	No	Billing cycle, which is mandatory only when the yearly/monthly billing mode is used. Options: – **month**: Resources are billed on a monthly basis. – **year**: Resources are billed on a yearly basis.
everest.io/csi.period-num	No	Required duration, which is mandatory only when the yearly/monthly billing mode is used. Options: – For monthly billing, the value of this parameter ranges from **1** to **9**. – For yearly billing, the value of this parameter is **1**.

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	StorageClass name, which is **csi-disk** for an EVS disk.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-evs-auto.yaml
```

Step 3 Create an application.

1. Create a file named **web-evs-auto.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-evs-auto
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-evs-auto
  serviceName: web-evs-auto   # Headless Service name
  template:
    metadata:
      labels:
        app: web-evs-auto
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
```



```
      volumes:
        - name: pvc-disk    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-evs-auto    # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-evs-auto   # Headless Service name
  namespace: default
  labels:
    app: web-evs-auto
spec:
  selector:
    app: web-evs-auto
  clusterIP: None
  ports:
    - name: web-evs-auto
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f web-evs-auto.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-evs-auto
```

Expected output:

```
web-evs-auto-0                1/1     Running   0             38s
```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

```
kubectl exec web-evs-auto-0 -- df | grep data
```

Expected output:

```
/dev/sdc            10255636     36888  10202364   0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Expected output:

```
lost+found
```

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-evs-auto-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```


Step 4 Run the following command to delete the pod named **web-evs-auto-0**:

```
kubectl delete pod web-evs-auto-0
```

Expected output:

```
pod "web-evs-auto-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-evs-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-10**.

Table 8-10 Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of an attached EVS disk on the CCE console. Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).

Operation	Description	Procedure
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.3.4 Dynamically Mounting an EVS Disk to a StatefulSet

Application Scenarios

Dynamic mounting is available only for creating a **StatefulSet**. It is implemented through a volume claim template (**volumeClaimTemplates** field) and depends on dynamic creation of PVs through StorageClass. In this mode, each pod in a multi-pod StatefulSet is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. In the common mounting mode for a Deployment, if ReadWriteMany is supported, multiple pods of the Deployment will be mounted to the same underlying storage.

NOTE

When updating a StatefulSet in Kubernetes, it is not allowed to add or delete the **volumeClaimTemplates** field. This field can only be configured during the creation of the StatefulSet.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Dynamically Mounting an EVS Disk on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.

Step 3 Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **VolumeClaimTemplate**.

Step 4 Click **Create PVC**. In the dialog box displayed, configure PVC parameters.

Click **Create**.

Parameter	Description
PVC Type	In this example, select **EVS**.
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <*Custom PVC name*>-<*Serial number*>, for example, example-0.
Creation Method	You can select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClasses for EVS disks are **csi-disk** and **csi-disk-topology**. **NOTE** If you use the **csi-disk** (EVS) StorageClass, a PVC and PV will be created immediately. The EVS disk is created with the PV, and then the PVC is bound to the PV. If you use the **csi-disk-topology** (EVS created with a delay) StorageClass, a PV will not be immediately created when a PVC is created. Instead, the pods that will be associated with the PVC are scheduled first, and then the EVS disk and PV are created, and finally the PV is bound to the PVC. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-***{UID}*.
AZ	Select the AZ of the EVS disk. The AZ must be the same as that of the cluster node. **NOTE** An EVS disk can only be mounted to a node in the same AZ. After an EVS disk is created, its AZ cannot be changed.
Disk Type	Select an EVS disk type. EVS disk types vary depending on regions. Obtain the available EVS types on the console. **NOTE** If the Everest version is 2.4.4 or later, general-purpose SSD V2 EVS disks are supported. General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**.
Capacity (GiB)	Capacity of the requested storage volume.

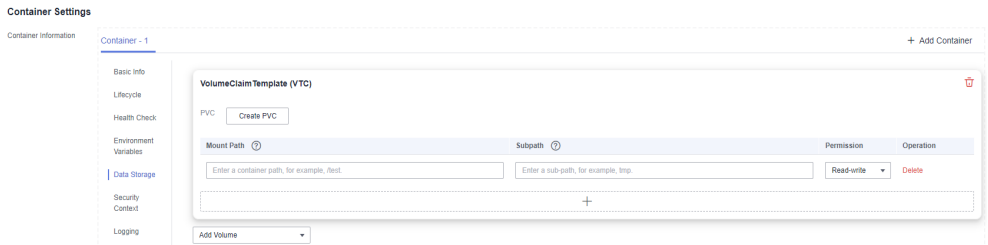
Parameter	Description
Billing Mode	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: • **Yearly/Monthly** Resources will be billed on a yearly/monthly basis. To use the yearly/monthly billing mode, you need to select the required duration, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. • **Pay-per-use** Resources will be billed based on usage duration. You can provision or delete resources at any time.
Access Mode	EVS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Encryption	Configure whether to encrypt underlying storage. If you select **Enabled (key)**, an encryption key must be configured. Before using encryption, check whether the region where the EVS disk is located supports disk encryption.
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.
Resource Tag	You can add resource tags to classify resources, which is supported only when the Everest version in the cluster is 2.1.39 or later. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}*, and **CCE-Namespace=***{Namespace name}*. These tags cannot be modified. **NOTE** After a dynamic PV of the EVS type is created, the resource tags cannot be updated on the CCE console. To update these resource tags, go to the EVS console.

Step 5 Enter the path to which the volume is mounted.

Table 8-11 Mounting a storage volume

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the EVS disk.



Step 6 Dynamically mount and use storage volumes. For details about other parameters, see **Creating a StatefulSet**. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Dynamically Mounting an EVS Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a file named **statefulset-evs.yaml**. In this example, the EVS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-evs
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-evs
  template:
    metadata:
      labels:
        app: statefulset-evs
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk          # The value must be the same as that in the volumeClaimTemplates field.
              mountPath: /data        # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-evs        # Headless Service name
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-disk
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS    # EVS disk type
          everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
          everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
          everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # (Optional) Custom resource tags
          everest.io/disk-iops: '3000'      # (Optional) IOPS of only a GPSSD2 EVS disk
          everest.io/disk-throughput: '125' # (Optional) Throughput of only a GPSSD2 EVS disk
          everest.io/csi.charging-mode: prePaid  # Mandatory if the billing mode is set to Yearly/Monthly
          everest.io/csi.period-type: month      # Mandatory if the billing mode is set to Yearly/Monthly. month indicates that resources will be billed on a monthly basis.
          everest.io/csi.period-num: '1'         # Mandatory if the billing mode is set to Yearly/Monthly. '1' indicates that the required duration is one month.
          everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
        labels:
          failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
          failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
      spec:
        accessModes:
          - ReadWriteOnce             # The value must be ReadWriteOnce for EVS disks.
        resources:
          requests:
            storage: 10Gi             # EVS disk capacity, ranging from 1 to 32768
        storageClassName: csi-disk    # The StorageClass is EVS.
---
apiVersion: v1
kind: Service
```



```
metadata:
  name: statefulset-evs   # Headless Service name
  namespace: default
  labels:
    app: statefulset-evs
spec:
  selector:
    app: statefulset-evs
  clusterIP: None
  ports:
    - name: statefulset-evs
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Table 8-12 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. • **SAS**: high I/O • **SSD**: ultra-high I/O • **GPSSD**: general-purpose SSD • **ESSD**: extreme SSD • **GPSSD2**: general-purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the **everest.io/disk-iops** and **everest.io/disk-throughput** annotations are configured
everest.io/disk-iops	No	Preconfigured IOPS, which is supported only by general-purpose SSD v2 EVS disks. • The IOPS of general-purpose SSD v2 EVS disks ranges from 3000 to 128000, and the maximum value is 500 times of the capacity (GiB). If the IOPS of general-purpose SSD v2 disks is greater than 3000, extra IOPS will be billed. For details, see **Price Calculator**.

Parameter	Mandatory	Description
everest.io/disk-throughput	No	Preconfigured throughput, which is supported only by general-purpose SSD v2 EVS disks. The value ranges from 125 MiB/s to 1000 MiB/s. The maximum value is a quarter of IOPS. If the throughput is greater than 125 MiB/s, extra throughput will be billed. For details, see **Price Calculator**.
everest.io/crypt-key-id	No	Mandatory when the EVS disk is encrypted. Enter the encryption key ID selected during EVS disk creation. To obtain an encryption key ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Elastic Volume Service** > **Disks**. Click the name of the target EVS disk to go to its details page. On the **Summary** tab page, copy the value of **KMS Key ID** in the **Configuration Information** area.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of the EVS disk. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Elastic Volume Service** > **Disks**. Click the name of the target EVS disk to go to its details page. On the **Summary** tab page, click the enterprise project in **Management Information** to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the EVS disk belongs.

Parameter	Mandatory	Description
everest.io/csi.charging-mode	No	This parameter is mandatory when the yearly/monthly billing mode is used. If this parameter is not specified, the pay-per-use billing mode will be used by default. This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.16 or later is installed in the cluster. Options: • **prePaid**: EVS disks are billed on a yearly/monthly basis, and auto-renewal is enabled by default. Your order will automatically renew on a monthly or yearly basis, depending on if you purchased by month or by year. • **postPaid**: EVS disks are billed on a pay-per-use basis. **NOTE** Before creating an EVS disk billed on a yearly/monthly basis, you need to grant the payment permission (**bss:order:pay**) to **cce_cluster_agency**. Use your tenant account on the CCE console to perform this operation for the first time. After confirmation, CCE will automatically authorize the agency.
everest.io/csi.period-type	No	Billing cycle, which is mandatory only when the yearly/monthly billing mode is used. Options: • **month**: Resources are billed on a monthly basis. • **year**: Resources are billed on a yearly basis.
everest.io/csi.period-num	No	Required duration, which is mandatory only when the yearly/monthly billing mode is used. Options: • For monthly billing, the value of this parameter ranges from **1** to **9**. • For yearly billing, the value of this parameter is **1**.

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	StorageClass name, which is **csi-disk** for an EVS disk.

Step 3 Run the following command to create a workload to which the EVS volume is mounted:

```
kubectl apply -f statefulset-evs.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and EVS volume files.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep statefulset-evs
   ```

   Expected output:

   ```
   statefulset-evs-0       1/1     Running   0          45s
   statefulset-evs-1       1/1     Running   0          28s
   ```

2. Run the following command to check whether the EVS volume has been mounted to the **/data** path:

   ```
   kubectl exec statefulset-evs-0 -- df | grep data
   ```

   Expected output:

   ```
   /dev/sdd        10255636     36888  10202364   0% /data
   ```

3. Run the following command to check the files in the **/data** path:

   ```
   kubectl exec statefulset-evs-0 -- ls /data
   ```

   Expected output:

   ```
   lost+found
   ```


Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec statefulset-evs-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **statefulset-evs-0**:

```
kubectl delete pod statefulset-evs-0
```

Expected output:

```
pod "statefulset-evs-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec statefulset-evs-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the EVS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-13**.

Table 8-13 Related operations

Operation	Description	Procedure
Expanding the capacity of an EVS disk	Quickly expand the capacity of an attached EVS disk on the CCE console. Only the capacity of pay-per-use EVS disks can be expanded on the CCE console. To expand the capacity of yearly/monthly EVS disks, click the volume name to go to the EVS console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.3.5 Encrypting EVS Disks

Encrypting cloud disks ensures data privacy and control, making it ideal for scenarios that demand high security or compliance standards. This section describes how to use the keys managed by Data Encryption Workshop (DEW) to encrypt EVS disks.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- An available key has been created in DEW. For details, see **Creating a Key**.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

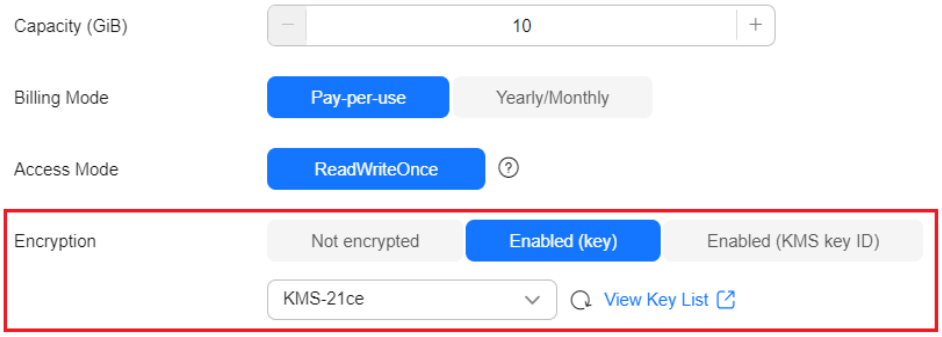
Using the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane and click the **PersistentVolumeClaims (PVCs)** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.
2. Select EVS for the storage type, enable encryption, and choose a key. Configure other parameters based on service requirements. For details, see **Using an EVS Disk Through a Dynamic PV**.

Figure 8-3 Encrypted storage volume



3. Click **Create**.

Step 3 Go to the **PersistentVolumeClaims (PVCs)** tab and check whether the PVC of the encrypted EVS disk is created and whether the disk is encrypted.

Figure 8-4 Encrypted PVC

Step 4 The method of using an encrypted PVC is the same as that of using a regular PVC.

----End

Automatically Creating an Encrypted EVS Disk Using kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create the **pvc-evs-auto.yaml** file. For details, see **Automatically Creating an EVS Volume Through kubectl**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # EVS disk type
    everest.io/crypt-key-id: 37f202db-a970-4ac1-a506-e5c4f2d7ce69   # Encryption key ID, which can be obtained from DEW
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
spec:
  accessModes:
  - ReadWriteOnce              # The value must be ReadWriteOnce for EVS disks.
  resources:
    requests:
      storage: 10Gi            # EVS disk capacity, ranging from 1 to 32768
  storageClassName: csi-disk    # The StorageClass is EVS.
```

Step 3 Run the following command to create a PVC:

```
kubectl apply -f pvc-evs-auto.yaml
```


Step 4 Go to the **PersistentVolumeClaims (PVCs)** tab and check whether the PVC of the encrypted EVS disk is created and whether the disk is encrypted.

----End

8.3.6 Expanding the Capacity of an EVS Disk

If the EVS disk attached to a workload does not have enough space, you can increase its capacity by expanding it. This section describes how to expand the capacity of an EVS disk through the console.

Prerequisites

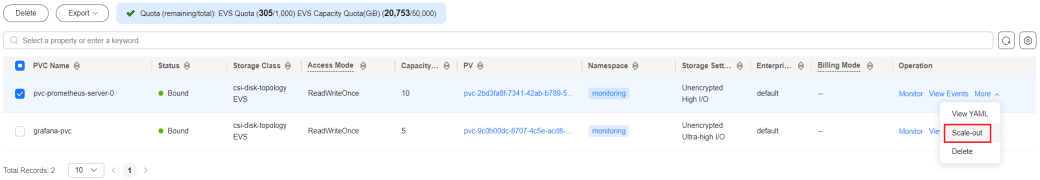
You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.

Pay-per-Use EVS Disk

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Storage** in the navigation pane and click the **PersistentVolumeClaims (PVCs)** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**.

Figure 8-5 Expanding an EVS disk capacity

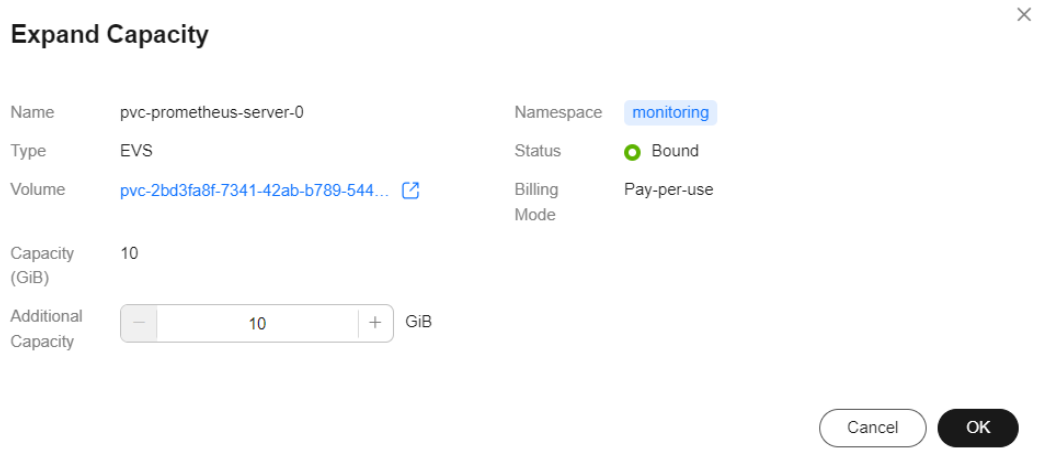


Step 3 Enter the capacity to be added and click **OK**.

NOTE

The disk size can only be increased, not decreased.

Figure 8-6 Configuring the capacity to be expanded



----End

Yearly/Monthly EVS Disk

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Storage** in the navigation pane and click the **PersistentVolumeClaims (PVCs)** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**.

Step 3 For a yearly/monthly disk, click the disk name to go to the EVS console and expand the disk capacity.

Step 4 Enter the target capacity and click **Next**.

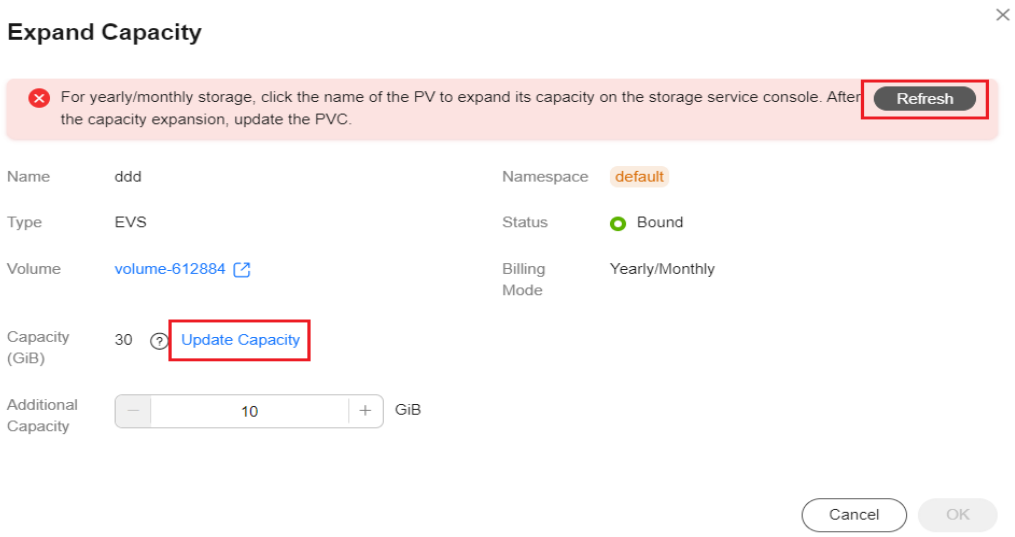
NOTE

The disk size can only be increased, not decreased.

Step 5 Read the precautions carefully and submit the yearly/monthly order again.

Step 6 After the EVS disk is expanded, go back to the CCE console, refresh the capacity expansion page, and update the capacity for the PVC.

Figure 8-7 Update Capacity



----End

8.3.7 Snapshots and Backups

CCE works with EVS to support snapshots. A snapshot is a complete copy or image of EVS disk data at a certain point of time, which can be used for data DR.

You can create snapshots to rapidly save the disk data at a certain point of time. In addition, you can use snapshots to create disks so that the created disks will contain the snapshot data in the beginning.

Precautions

- The snapshot function is available **only for clusters of v1.15 or later** and requires the CSI-based Everest add-on.

- The subtype (common I/O, high I/O, or ultra-high I/O), disk mode (VBD or SCSI), data encryption, sharing status, and capacity of an EVS disk created from a snapshot must be the same as those of the disk associated with the snapshot. These attributes cannot be modified after being checked or configured.
- Snapshots can be created only for EVS disks that are available or in use, and a maximum of seven snapshots can be created for a single EVS disk.
- Snapshots can be created only for PVCs created using the storage class (whose name starts with csi) provided by the Everest add-on. Snapshots cannot be created for PVCs created using the FlexVolume storage class whose name is ssd, sas, or sata.
- Snapshot data of encrypted disks is stored encrypted, and that of non-encrypted disks is stored non-encrypted.

Application Scenarios

The snapshot feature helps address your following needs:

- **Routine data backup**

  You can create snapshots for EVS disks regularly and use snapshots to recover your data in case that data loss or data inconsistency occurred due to misoperations, viruses, or attacks.

- **Rapid data restoration**

  You can create a snapshot or multiple snapshots before an OS change, application software upgrade, or a service data migration. If an exception occurs during the upgrade or migration, service data can be rapidly restored to the time point when the snapshot was created.

  For example, a fault occurred on system disk A of ECS A, and therefore ECS A cannot be started. Because system disk A is already faulty, the data on system disk A cannot be restored by rolling back snapshots. In this case, you can use an existing snapshot of system disk A to create EVS disk B and attach it to ECS B that is running properly. Then, ECS B can read data from system disk A using EVS disk B.

  NOTE

  The snapshot capability provided by CCE is the same as the CSI snapshot function provided by the Kubernetes community. EVS disks can be created only based on snapshots, and snapshots cannot be rolled back to source EVS disks.

- **Rapid deployment of multiple services**

  You can use a snapshot to create multiple EVS disks containing the same initial data, and these disks can be used as data resources for various services, for example, data mining, report query, and development and testing. This method protects the initial data and creates disks rapidly, meeting the diversified service data requirements.

Creating a Snapshot

Using the CCE console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console. Choose **Storage** in the navigation pane. In the right pane, click the **Snapshots and Backups** tab.

Step 3 Click **Create Snapshot** in the upper right corner. In the dialog box displayed, set related parameters.

- **Snapshot Name**: Enter a snapshot name.
- **Storage**: Select an EVS PVC.

Step 4 Click **Create**.

----End

Using YAML

```
kind: VolumeSnapshot
apiVersion: snapshot.storage.k8s.io/v1beta1
metadata:
  name: cce-disksnap-test   # Snapshot name
  namespace: default
spec:
  source:
    persistentVolumeClaimName: pvc-evs-test     # PVC name. Only an EVS PVC can be selected.
  volumeSnapshotClassName: csi-disk-snapclass
```

Using a Snapshot to Create a PVC

The disk type, encryption setting, and disk mode of the created EVS PVC are consistent with those of the snapshot's source EVS disk.

Using the CCE console

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to go to the cluster console. Choose **Storage** in the navigation pane. In the right pane, click the **Snapshots and Backups** tab.

Step 3 Locate the snapshot that you want to use for creating a PVC, click **Create PVC**, and configure PVC parameters in the displayed dialog box.

- **PVC Name**: Enter a PVC name.
- **Resource Tag**: Resource tags can be added to classify resources, which is supported only when the Everest version in the cluster is 2.1.39 or later.

  You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**.

  CCE automatically creates system tags **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}*, and **CCE-Namespace=***{Namespace name}*. These tags cannot be modified.

Step 4 Click **Create**.

----End

Using YAML

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
```



```
  namespace: default
  annotations:
    everest.io/disk-volume-type: SSD     # EVS disk type, which must be the same as that of the snapshot's source EVS disk.
    everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # (Optional) Custom resource tags
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Replace the region with the one where the EVS disk is located.
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # Replace the AZ with the one where the EVS disk is located.
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
  dataSource:
    name: cce-disksnap-test            # Snapshot name
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

8.4 Scalable File Service

8.4.1 Overview

Introduction

CCE allows you to mount a volume created from a Scalable File Service (SFS) file system to a container to store data persistently. SFS volumes are commonly used in ReadWriteMany scenarios for large-capacity expansion and cost-sensitive services, such as media processing, content management, big data analysis, and workload process analysis. For services with massive volume of small files, SFS Turbo file systems are recommended.

Expandable to petabytes, SFS provides fully hosted shared file storage, highly available and stable to handle data- and bandwidth-intensive applications

- **Standard file protocols**: You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing**: The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network**: Users can access data only in private networks of data centers.
- **Capacity and performance**: The capacity of a single file system is high (PB level) and the performance is excellent (ms-level I/O latency).
- **Use cases**: Deployments/StatefulSets in the ReadWriteMany mode and jobs created for high-performance computing (HPC), media processing, content management, web services, big data analysis, and workload process analysis

Performance

CCE supports SFS Capacity-Oriented and SFS 3.0. For details about file system types, see **File System Type**.

NOTE

- SFS Capacity-Oriented file systems are sold out and cannot be used to create PVs on the CCE console. However, you can still create PVs using existing SFS Capacity-Oriented file systems through **kubectl**.
- SFS 3.0 is being rolled out in different regions, but it may not be available in all regions. If you encounter any issues, contact SFS customer support or wait for further updates. If the region where your application is located already has SFS 3.0 available, use it for new applications and migrate existing SFS Capacity-Oriented file systems to SFS 3.0 as soon as possible to prevent any service disruptions caused by insufficient capacity.

Table 8-14 Performance

Parameter	SFS Capacity-Oriented	SFS 3.0
Maximum bandwidth	2 GB/s	1.25 TB/s
Maximum IOPS	2000	Million
Latency	3–20 ms	10 ms
Maximum capacity	4 PB	EB

Application Scenarios

SFS supports the following mounting modes based on application scenarios:

- **Using an Existing SFS File System Through a Static PV**: static creation mode, where you use an existing SFS volume to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an SFS File System Through a Dynamic PV**: dynamic creation mode, in which you do not need to create SFS file systems beforehand. Instead, specify a StorageClass when creating a PVC. Then, a file system and PV will be created automatically. This mode applies to scenarios where no underlying storage is available.

Billing

- The default billing mode of an SFS volume **automatically created** using StorageClass is **Pay-per-use**, indicating that it is billed based on the used storage capacity and duration. For details about SFS pricing, see **Billing**.
- To use a yearly/monthly-billed SFS volume, **use an existing one**.

8.4.2 Using an Existing SFS File System Through a Static PV

SFS is a network-attached storage (NAS) that provides shared, scalable, and high-performance file storage. It applies to large-capacity expansion and cost-sensitive services. This section describes how to use an existing SFS file system to statically create PVs and PVCs for data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created an SFS file system that is in the same VPC as the cluster.
- Before using SFS 3.0 for storage, ensure a VPC endpoint has been created in the VPC where the cluster is located for the cluster to access the SFS 3.0 file system. For details, see **Configuring a VPC Endpoint**.

Notes and Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying SFS or SFS Turbo volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** value.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.
  - When the underlying volume is repeatedly used, enable isolation and protection for ReadWriteMany at the application layer to prevent data overwriting and loss.
- If SFS 3.0 is used, the Everest add-on of v2.0.9 or later must be installed in the cluster.
- If SFS 3.0 is used, the owner group and permission of the mount point cannot be modified. The default owner of the mount point is user **root**.
- If SFS 3.0 is used, there may be a latency during the creation or deletion of PVCs and PVs. The billing duration is determined by the time when the SFS system is created or deleted on the SFS console.
- If the reclamation policy of SFS 3.0 is set to **Delete**, PVs and PVCs can be properly deleted only after all files are manually deleted from the mounted SFS system.

Using an Existing SFS File System on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS**.

Parameter	Description
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using an SFS File System Through a Dynamic PV**. In this example, select **Create new** to create both a PV and PVC on the console.
PV[a]	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in **Related Operations**. You do not need to specify this parameter in this example.
SFS[b]	Click **Select SFS**. On the displayed page, select the SFS file system that meets your requirements and click **OK**. **NOTE** Only SFS 3.0 is supported.
Subdirectory[b]	Determine whether to use subdirectories to create PVs. Enter the absolute path of a subdirectory, for example, **/a/b**. Ensure that the subdirectory is available.
PV Name[b]	Enter the PV name, which must be unique in the same cluster.
Access Mode[b]	SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
Reclaim Policy[b]	You can select **Delete** or **Retain** to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see **PV Reclaim Policy**. **NOTE** If multiple PVs use the same underlying storage volume, use **Retain** to prevent the underlying volume from being deleted with a PV.
Subdirectory Reclaim Policy[b]	Determine whether to retain subdirectories when a PVC is deleted. This parameter must be used with **PV Reclaim Policy** and can be configured when PVs are created using subdirectories and **PV Reclaim Policy** is set to **Delete**. – **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. – **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**.

Parameter	Description
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring SFS Volume Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

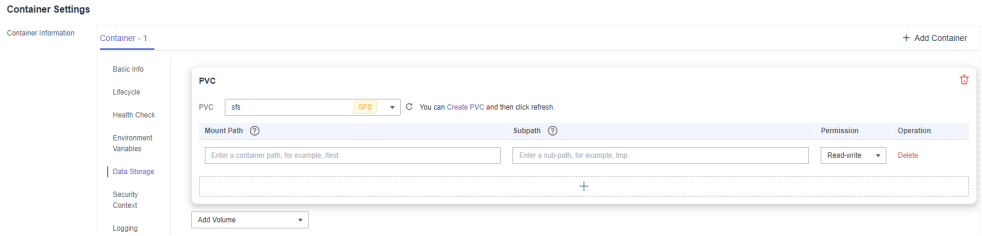
   Mount and use storage volumes, as shown in **Table 8-15**. For details about other parameters, see **Workloads**.

Table 8-15 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.

Parameter	Description
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS file system.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Using an Existing SFS File System Through kubectl

You can select a mode that meets the requirements of your service.

Using an Existing SFS 3.0 File System

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfs.yaml** file.

   Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-sfs    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi     # SFS volume capacity
  csi:
    driver: nas.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <sfs30_name>   # Enter the file system name when SFS 3.0 is used.
```



```
  volumeAttributes:
    everest.io/share-export-location: <your_location>  # Shared path of the SFS volume
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    everest.io/sfs-version: sfs3.0       # SFS 3.0 is used.
persistentVolumeReclaimPolicy: Retain    # Reclaim policy
storageClassName: csi-sfs                # StorageClass name, where csi-sfs indicates SFS 3.0
mountOptions: []                         # Mount options
```

Table 8-16 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy	No	Only **retain-volume-only** is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is **Delete**. If the reclaim policy is **Delete** and the current value is **retain-volume-only**, the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
volumeHandle	Yes	If SFS 3.0 is used, enter the name of the file system.
everest.io/share-export-location	Yes	Shared path of SFS 3.0. A shared path is in the following format: *{your_sfs30_name}*.sfs3.*{region}*.myhuaweicloud.com:/*{your_sfs30_name}*
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see **Configuring SFS Volume Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If multiple PVs use the same SFS volume, use **Retain** to prevent the underlying volume from being deleted with a PV. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again. **Delete**: When a PVC is deleted, its PV will also be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for SFS file systems.
storageClassName	Yes	StorageClass name. Enter **csi-sfs**, indicating that SFS 3.0 is used.

2. Run the following command to create a PV:
```
kubectl apply -f pv-sfs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-sfs.yaml** file.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany              # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi             # SFS volume capacity
  storageClassName: csi-sfs    # StorageClass name, which must be the same as that of the PV
  volumeName: pv-sfs    # PV name
```


Table 8-17 Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name. Enter **csi-sfs**, indicating that SFS 3.0 is used. The value of this parameter must be the same as the PV StorageClass specified in **1**.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-sfs.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfs    # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Using Subdirectories of an Existing SFS 3.0 File System

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfs.yaml** file.

   Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfs    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi     # SFS volume capacity
  csi:
    driver: nas.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: pv-sfs    # Enter the PV name when using a subdirectory.
    volumeAttributes:
      everest.io/share-export-location: <your_location>  # Shared path of the SFS volume
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/sfs-version: sfs3.0       # SFS 3.0 is used.
      everest.io/csi.volume-as: absolute-path    # An SFS subdirectory is used.
      everest.io/csi.path: /a  # An automatically created subdirectory, which must be an absolute path
      everest.io/csi.sfs30-name: <sfs_name>     # SFS instance name. This parameter is available only
when subdirectories are used.
      everest.io/csi.reclaim-policy: retain-volume-only    # When a PVC is deleted, the PV will be deleted
but the subdirectories associated with the PV will be retained. This parameter is available only
when subdirectories are used and the reclaim policy is Delete.
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy
  storageClassName: csi-sfs              # StorageClass name, where csi-sfs indicates SFS 3.0
  mountOptions: []                       # Mount options
```

Table 8-18 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	PV name when an SFS subdirectory is used.
everest.io/share-export-location	Yes	Shared path of SFS 3.0. A shared path is in the following format: *{your_sfs30_name}*.sfs3.*{region}*.myhuaweicloud.com:/*{your_sfs30_name}*/*{absolute_path}*
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see **Configuring SFS Volume Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If multiple PVs use the same SFS volume, use **Retain** to prevent the underlying volume from being deleted with a PV. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again. **Delete**: When a PVC is deleted, its PV will also be deleted.
everest.io/csi.volume-as	No	The value is fixed at **absolute-path**, indicating that a dynamically created SFS subdirectory is used. Ensure Everest of v2.4.41 or later has been installed in the cluster.
everest.io/csi.path	No	Subdirectory that is automatically created, which must be an absolute path.
everest.io/csi.sfs30-name	No	An SFS instance name. This parameter is used only when an SFS subdirectory is dynamically created.

Parameter	Mandatory	Description
everest.io/csi.reclaim-policy	No	Whether to retain the subdirectory when deleting a PVC. Use this parameter only when creating a dynamic SFS subdirectory. This parameter must be used with **PV Reclaim Policy**. This parameter is available only when the PV reclaim policy is **Delete**. Options: – **retain-volume-only**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. – **delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for SFS file systems.
storageClassName	Yes	StorageClass name. Enter **csi-sfs**, indicating that SFS 3.0 is used.

2. Run the following command to create a PV:
```
kubectl apply -f pv-sfs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-sfs.yaml** file.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/csi.volume-as: absolute-path # (Optional) An SFS subdirectory is used.
    everest.io/csi.path: /a  # An automatically created subdirectory, which must be an absolute path
    everest.io/csi.sfs30-name: <sfs_name>     # SFS instance name. This parameter is available only when subdirectories are used.
    everest.io/csi.reclaim-policy: retain-volume-only
spec:
  accessModes:
  - ReadWriteMany              # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi             # SFS volume capacity
  storageClassName: csi-sfs    # StorageClass name, which must be the same as that of the PV
  volumeName: pv-sfs    # PV name
```


Table 8-19 Key parameters

Parameter	Mandatory	Description
everest.io/csi.volume-as	Yes	The value is fixed at **absolute-path**, indicating that a dynamically created SFS subdirectory is used. Ensure Everest of v2.4.41 or later has been installed in the cluster.
everest.io/csi.path	Yes	Subdirectory that is automatically created, which must be an absolute path.
everest.io/csi.sfs30-name	Yes	An SFS instance name. This parameter is used only when an SFS subdirectory is dynamically created.
everest.io/csi.reclaim-policy	Yes	Whether to retain the subdirectory when deleting a PVC. Use this parameter only when creating a dynamic SFS subdirectory. This parameter must be used with **PV Reclaim Policy**. This parameter is available only when the PV reclaim policy is **Delete**. Options: – **retain-volume-only**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. – **delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name. Enter **csi-sfs**, indicating that SFS 3.0 is used. The value of this parameter must be the same as the PV StorageClass specified in **1**.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-sfs.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfs    # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Using an Existing SFS Capacity-Oriented File System

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfs.yaml** file.

Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-sfs    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi    # SFS volume capacity
  csi:
    driver: nas.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <your_volume_id>   # SFS Capacity-Oriented volume ID
```



```
  volumeAttributes:
    everest.io/share-export-location: <your_location>  # Shared path of the SFS volume
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain     # Reclaim policy
  storageClassName: csi-nas                 # StorageClass name. csi-nas indicates that SFS Capacity-Oriented is used.
  mountOptions: []                          # Mount options
```

Table 8-20 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy	No	Only **retain-volume-only** is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is **Delete**. If the reclaim policy is **Delete** and the current value is **retain-volume-only**, the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
volumeHandle	Yes	Volume ID if SFS Capacity-Oriented is used. Log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Capacity-Oriented**. In the list, click the name of the target SFS file system. On the details page, copy the content following **ID**.
everest.io/share-export-location	Yes	Shared path of the file system. On the management console, choose **Service List** > **Storage** > **Scalable File Service**. You can obtain the shared path of the file system from the **Mount Address** column.
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see **Configuring SFS Volume Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If multiple PVs use the same SFS volume, use **Retain** to prevent the underlying volume from being deleted with a PV. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again. **Delete**: When a PVC is deleted, its PV will also be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for SFS file systems.
storageClassName	Yes	StorageClass name **csi-nas**, indicating that SFS 1.0 Capacity-Oriented is used for storage.

2. Run the following command to create a PV:
```
kubectl apply -f pv-sfs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-sfs.yaml** file.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany               # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi              # SFS volume capacity
  storageClassName: csi-nas     # StorageClass name, which must be the same as that of the PV
  volumeName: pv-sfs    # PV name
```


Table 8-21 Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name **csi-nas**, which must be the same as the StorageClass of the PV specified in **1**. This indicates that SFS 1.0 Capacity-Oriented is used for storage.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-sfs.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfs    # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0          46s
   web-demo-846b489584-wvv5s   1/1     Running   0          46s
   ```
2. Run the following commands in sequence to view the files in the **/data** path of the pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Expected output:
```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Expected output:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   After the deletion, the Deployment controller automatically creates a replica.
2. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          110s
   web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
   ```
3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   static
   ```
   The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 **Verify data sharing.**

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          7m
   web-demo-846b489584-wvv5s   1/1     Running   0          13m
   ```
2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.


```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in **Table 8-22**.

Table 8-22 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Click **Create PersistentVolume** in the upper right corner. In the dialog box displayed, configure parameters. • **Volume Type**: Select **SFS**. • **SFS**: Click **Select SFS**. On the displayed page, select the SFS file system that meets your requirements and click **OK**. • **Subdirectory**: Determine whether to use subdirectories to create PVs. Enter the absolute path of a subdirectory, for example, **/a/b**. Ensure that the subdirectory is available. • PV Name: Enter the PV name, which must be unique in a cluster. • **Access Mode**: SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**. • **Reclaim Policy**: **Delete** or **Retain** is supported. For details, see **PV Reclaim Policy**. **NOTE** If multiple PVs use the same underlying storage volume, use **Retain** to prevent the underlying volume from being deleted with a PV. • **Subdirectory Reclaim Policy**: Determine whether to retain subdirectories when a PVC is deleted. This parameter must be used with **PV Reclaim Policy** and can be configured when PVs are created using subdirectories and **PV Reclaim Policy** is set to **Delete**. **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. • **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **Configuring SFS Volume Mount Options**. 2. Click **Create**.

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.4.3 Using an SFS File System Through a Dynamic PV

This section describes how to use storage classes to dynamically create PVs and PVCs for data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- Before using SFS 3.0 for storage, ensure a VPC endpoint has been created in the VPC where the cluster is located for the cluster to access the SFS 3.0 file system. For details, see **Configuring a VPC Endpoint**.

Notes and Constraints

- If SFS 3.0 is used, the **CCE Container Storage (Everest)** add-on of v2.0.9 or later must be installed in the cluster.
- If SFS 3.0 is used, the owner group and permission of the mount point cannot be modified. The default owner of the mount point is user **root**.
- If SFS 3.0 is used, there may be a latency during the creation or deletion of PVCs and PVs. The billing duration is determined by the time when the SFS system is created or deleted on the SFS console.
- If the reclamation policy of SFS 3.0 is set to **Delete**, PVs and PVCs can be properly deleted only after all files are manually deleted from the mounted SFS system.

Automatically Creating an SFS File System on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If no underlying storage is available, select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode. – If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see **Using an Existing SFS File System Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The default StorageClass for SFS volumes is **csi-sfs** or **csi-nas**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
Access Mode	SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
Encryption	Configure whether to encrypt underlying storage if the storage class is **csi-nas**. If you select **Enabled (key)**, an encryption key must be configured.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

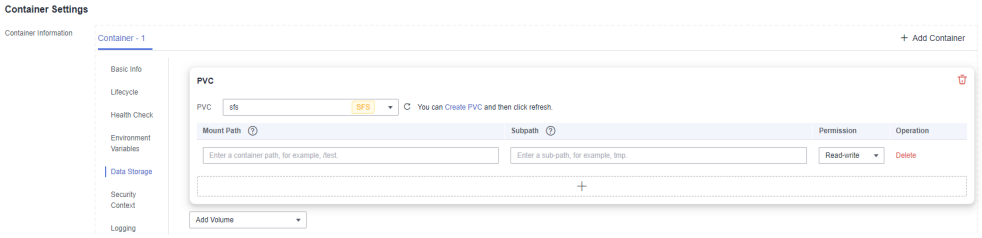
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-23**. For details about other parameters, see **Workloads**.

   Table 8-23 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS file system.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Automatically Creating an SFS File System Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-sfs-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-auto
  namespace: default
  annotations:
    everest.io/crypt-key-id: <your_key_id>      # (Optional) ID of the key for encrypting file systems

    everest.io/crypt-alias: sfs/default         # (Optional) Key name. Mandatory for encrypting volumes.

    everest.io/crypt-domain-id: <your_domain_id>  # (Optional) ID of the tenant to which an encrypted volume belongs. Mandatory for encrypting volumes.

    everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi              # SFS volume capacity
  storageClassName: csi-nas     # StorageClass is SFS. csi-nas indicates that SFS Capacity-Oriented is used, and csi-sfs indicates that SFS 3.0 is used.
```

Table 8-24 Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. For SFS, this field is used only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for SFS file systems.

Parameter	Mandatory	Description
storageClassName	Yes	Name of a storage class. – **csi-sfs** (recommended): indicates that SFS 3.0 is used. – **csi-nas**: indicates that an SFS 1.0 Capacity-Oriented file system is used.
everest.io/crypt-key-id	No	If StorageClass is **csi-nas**, you can determine whether to encrypt the underlying storage. This parameter is mandatory when an SFS system is encrypted. Enter the encryption key ID selected during SFS system creation. You can use a custom key or the default key named **sfs/default**. To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/crypt-alias	No	Key name, which is mandatory when you create an encrypted volume. To obtain a key name, log in to the DEW console, locate the key to be encrypted, and copy the key name.
everest.io/crypt-domain-id	No	ID of the tenant to which the encrypted volume belongs. This parameter is mandatory for creating an encrypted volume. To obtain a tenant ID, hover the cursor over the username in the upper right corner of the ECS console, choose **My Credentials**, and copy the account ID.

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-sfs-auto.yaml
```

Step 3 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS volume is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfs-auto    # Name of the created PVC
```
2. Run the following command to create a workload to which the SFS volume is mounted:
```
kubectl apply -f web-demo.yaml
```


After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0          46s
   web-demo-846b489584-wvv5s   1/1     Running   0          46s
   ```
2. Run the following commands in sequence to view the files in the **/data** path of the pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Expected output:
```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Expected output:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   After the deletion, the Deployment controller automatically creates a replica.
2. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          110s
   web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
   ```
3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   static
   ```
   The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 **Verify data sharing.**

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```


Expected output:

```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in **Table 8-25**.

Table 8-25 Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

Troubleshooting

Mounting the SFS 3.0 storage timed out, and the error message is as follows:

```
MountVolume.SetUp failed for volume "***" : rpc error: code = Internal desc = [30834707-b8fc-11ee-ba7a-fa163eaacb17] failed to execute cmd: "systemd-run --scope  mount -t nfs -o proto=tcp -o vers=3 -o
```


timeo=600 -o noresvport -o nolock ***.sfs3.cn-east-3.myhuaweicloud.com:/*** /mnt/paas/kubernetes/kubelet/pods/add9a323-10e2-434f-b151-42675f83860e/volumes/kubernetes.io~csi/***/mount". outputs: Running scope as unit run-1597979.scope.; error: signal: killed. **please check whether the volumeAttributes of related PersistentVolume of the volume is correct and whether it can be mounted.**

Solution

Before using SFS 3.0, create a VPC endpoint. Otherwise, the cluster cannot access the SFS 3.0 file system. As a result, the mounting times out and fails. For details, see **Configuring a VPC Endpoint**.

8.4.4 Creating an SFS Subdirectory Using a Dynamic PV

When an SFS volume is mounted to a workload container, the root directory is mounted to the container by default. To properly use the storage capacity, CCE allows you to dynamically create an SFS subdirectory when creating a PVC so that different workloads can share one SFS volume.

NOTICE

Only SFS 3.0 supports dynamic creation of subdirectories.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on of v2.4.41 or later in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created an available SFS file system, and the SFS file system and the cluster are in the same VPC. Before using SFS 3.0 for storage, ensure a VPC endpoint has been created in the VPC where the cluster is located for the cluster to access the SFS 3.0 file system. For details, see **Configuring a VPC Endpoint**.

Dynamically Creating an SFS Subdirectory on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	Select **New subdirectory**.

Parameter	Description
Storage Classes	Select **csi-sfs** for SFS. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
Access Mode	SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
File Storage	Click **Select SFS**. On the displayed page, select the SFS file system that meets your requirements and click **OK**.
Subdirectory	Enter the absolute path of a subdirectory, for example, **/a/b**.
Subdirectory Reclaim Policy	Whether to retain subdirectories when a PVC is deleted. • **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. • **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.

Step 3 Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

----End

Dynamically Creating an SFS Subdirectory Using kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create the **pvc-sfs-subpath.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-subpath    # PVC name
  namespace: default
  annotations:
    everest.io/csi.volume-as: absolute-path              # An SFS subdirectory is used.
    everest.io/csi.sfs30-name: <sfs_name>                # Name of the SFS instance
    everest.io/csi.path: /a                              # Subdirectory that is automatically created, which must be an absolute path
    everest.io/csi.reclaim-policy: retain-volume-only    # When a PVC is deleted, the PV will be deleted, but the subdirectories associated with the PV will be retained.
spec:
  accessModes:
    - ReadWriteMany      # ReadWriteMany must be selected for SFS.
  resources:
    requests:
      storage: 1Gi       # This parameter is only used for verification for the PVCs of the SFS subdirectory type. The value cannot be empty or 0.
  storageClassName: csi-sfs     # StorageClass name of SFS 3.0
```


Table 8-26 Key parameters

Parameter	Mandatory	Description
everest.io/csi.volume-as	Yes	When a dynamically created SFS subdirectory is used, the value of this parameter is consistently **absolute-path**.
everest.io/csi.sfs30-name	Yes	SFS instance name.
everest.io/csi.path	Yes	Subdirectory that is automatically created, which must be an absolute path.
everest.io/csi.reclaim-policy	Yes	Whether to retain subdirectories when deleting a PVC. This parameter must be used with **PV Reclaim Policy**. This parameter is available only when the PV reclaim policy is **Delete**. Options: • **retain-volume-only**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. • **delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. This parameter is only used for verification for the PVCs of the SFS subdirectory type. The value cannot be empty or **0**, and can be fixed at **1** because any value you set does not take effect.

Step 3 Run the following command to create a PVC:

```
kubectl apply -f pvc-sfs-subpath.yaml
```

----End

8.4.5 Configuring SFS Volume Mount Options

This section describes how to configure SFS mount options. You can configure mount options in a PV and bind the PV to a PVC. Alternatively, configure mount options in a StorageClass and use the StorageClass to create a PVC. In this way, PVs can be dynamically created and inherit mount options configured in the StorageClass by default.

Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

Notes and Constraints

- Mount options cannot be configured for Kata containers.
- Due to the restrictions of the NFS protocol, if an SFS volume is mounted to a node for multiple times, link-related mounting parameters (such as **timeo**) take effect only when the SFS volume is mounted for the first time by default. For example, if the same SFS file system is mounted to multiple pods running on a node, the mounting parameter set later does not overwrite the existing parameter value. If you want to configure different mounting parameters in the preceding scenario, additionally configure the **nosharecache** parameter.

SFS Volume Mount Options

The Everest add-on in CCE presets the options described in **Table 8-27** for mounting SFS volumes.

Table 8-27 SFS volume mount options

Parameter	Value	Description
keep-original-ownership	Blank	Whether to retain the ownership of the file mount point. If this option is used, the Everest add-on must be v1.2.63 or v2.1.2 or later. • By default, this option is not added, and the mount point ownership is **root:root** when SFS is mounted. • If this option is added, the original ownership of the file system is retained when SFS is mounted.
vers	3	File system version. Currently, only NFSv3 is supported. Value: **3**
nolock	Blank	Whether to lock files on the server using the NLM protocol. If **nolock** is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid.
timeo	600	Waiting time before the NFS client retransmits a request. The unit is 0.1 seconds. Recommended value: **600**

Parameter	Value	Description
hard/soft	Blank	Mount mode. • **hard**: If the NFS request times out, the client keeps resending the request until the request is successful. • **soft**: If the NFS request times out, the client returns an error to the invoking program. The default value is **hard**.
sharecache/ nosharecache	Blank	How the data cache and attribute cache are shared when one file system is concurrently mounted to different clients. If this parameter is set to **sharecache**, the caches are shared between the mountings. If this parameter is set to **nosharecache**, the caches are not shared, and one cache is configured for each client mounting. The default value is **sharecache**. **NOTE** The **nosharecache** setting will affect the performance. The mounting information must be obtained for each mounting, which increases the communication overhead with the NFS server and the memory consumption of the NFS clients. In addition, the **nosharecache** setting on the NFS clients may lead to inconsistent caches. Determine whether to use **nosharecache** based on site requirements.

You can configure other mount options if needed. For details, see **Mounting an NFS File System to ECSs (Linux)**.

Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in **SFS Volume Mount Options**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-sfs
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS.
  capacity:
    storage: 1Gi     # SFS volume capacity
  csi:
    driver: nas.csi.everest.io    # Dependent storage driver for the mounting
```



```
  fsType: nfs
  volumeHandle: <your_volume_id>  # ID of the SFS Capacity-Oriented volume
  volumeAttributes:
    everest.io/share-export-location: <your_location>  # Shared path of the SFS volume
    storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
persistentVolumeReclaimPolicy: Retain    # Reclaim policy
storageClassName: csi-nas              # StorageClass name.
mountOptions:                          # Mount options
- vers=3
- nolock
- timeo=600
- hard
```

Step 3 After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see **Using an Existing SFS File System Through a Static PV**.

Step 4 Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS volume has been mounted. In this example, the workload name is **web-sfs**.

   ```
   kubectl get pod | grep web-sfs
   ```

   Command output:

   ```
   web-sfs-***  1/1     Running   0          23m
   ```

2. Run the following command to check the mount options (**web-sfs-*** ** is an example pod):

   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```

   If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.

   ```
   <Your shared path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
   timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=2050,mountproto=tcp
   ,local_lock=all,addr=**.**.**.**)
   ```

----End

Configuring Mount Options in a StorageClass

You can use the **mountOptions** field to configure mount options in a StorageClass. The options you can configure in **mountOptions** are listed in **SFS Volume Mount Options**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a custom StorageClass. Example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sfs-mount-option
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: nas.csi.everest.io
  csi.storage.k8s.io/fstype: nfs
everest.io/share-access-to: <your_vpc_id> # VPC ID of the cluster
reclaimPolicy: Delete
volumeBindingMode: Immediate
```



```
mountOptions:                 # Mount options
- vers=3
- nolock
- timeo=600
- hard
```

Step 3 After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options configured in the StorageClass. For details, see **Using an SFS File System Through a Dynamic PV**.

Step 4 Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS volume has been mounted. In this example, the workload name is **web-sfs**.
   ```
   kubectl get pod | grep web-sfs
   ```
   Command output:
   ```
   web-sfs-***  1/1     Running   0          23m
   ```
2. Run the following command to check the mount options (**web-sfs-*** ** is an example pod):
   ```
   kubectl exec -it web-sfs-*** -- mount -l | grep nfs
   ```
   If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.
   ```
   <Your shared path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
   timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=2050,mountproto=tcp
   ,local_lock=all,addr=**.**.**.**)
   ```

----End

8.4.6 Migrating Containerized Application Data from SFS 1.0 to SFS 3.0 or SFS Turbo

Scalable File Service (SFS) provides the following types of file systems: SFS 1.0 (Capacity-Oriented), SFS 3.0 (General Purpose), and SFS Turbo. For details, see **File System Types**.

In earlier versions, CCE supports mounting SFS 1.0 to workloads. You are advised to migrate workloads to SFS 3.0 or SFS Turbo.

You can select a proper storage mounting mode according to your workload types. Dynamic mounting and static mounting are distinguished by mounting storage volumes to workloads.

- Dynamic mounting applies to only StatefulSets. This function is implemented using the **volumeClaimTemplates** field and depends on dynamic creation of PVs through StorageClass. A StatefulSet associates each pod with a PVC using the **volumeClaimTemplates** field, and the PVC is bound to the corresponding PV. Therefore, after the pod is rescheduled, the original data can still be mounted based on the PVC name.
- Static mounting applies to all types of workloads. This function is implemented using the **volumes** field.

NOTICE

To migrate containerized applications from SFS 1.0 to SFS 3.0 or SFS Turbo, the procedure is the same. However, the only difference is that SFS Turbo does not allow dynamic creation, which affects StatefulSets that rely on it.

Notes and Constraints

- You need to migrate data from SFS 1.0 to SFS 3.0 or SFS Turbo beforehand. For details, see **Migrating Data from SFS Capacity-Oriented to Another Type of File Systems**. If necessary, contact SFS customer service.
- The SFS file system and the cluster must be in the same VPC.
- Before using SFS 3.0 for storage, ensure a VPC endpoint has been created in the VPC where the cluster is located for the cluster to access the SFS 3.0 file system. For details, see **Configuring a VPC Endpoint**.

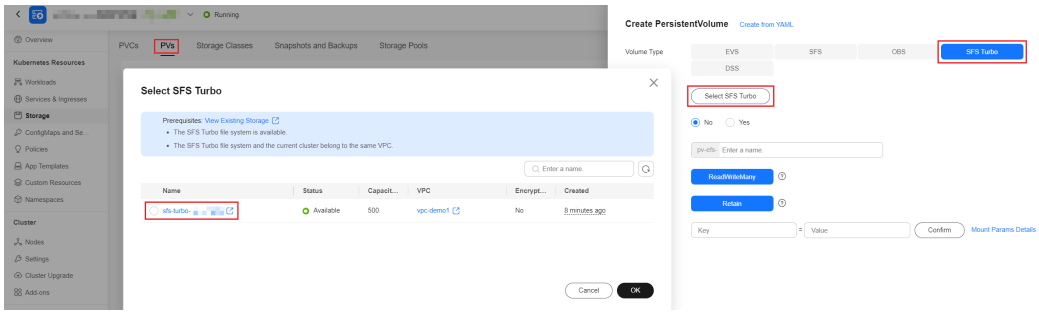
Migrating Data from Statically Mounted Storage

Static mounting applies to all types of workloads. This function is implemented using the **volumes** field. The procedure for migrating storage data using this mounting method from SFS 1.0 to SFS 3.0 or SFS Turbo is the same. This section uses SFS Turbo as an example.

Step 1 Access the cluster console and choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Then, click **Create PersistentVolume** in the upper right corner.

Configure the following key parameters:

- **Volume Type**: Select **SFS Turbo**.
- **SFS Turbo**: Select the target SFS Turbo volume for data migration.
- **PV Name**: Enter a custom name.
- **Access Mode**: Select **ReadWriteMany**.
- **Reclaim Policy**: Select **Retain**.

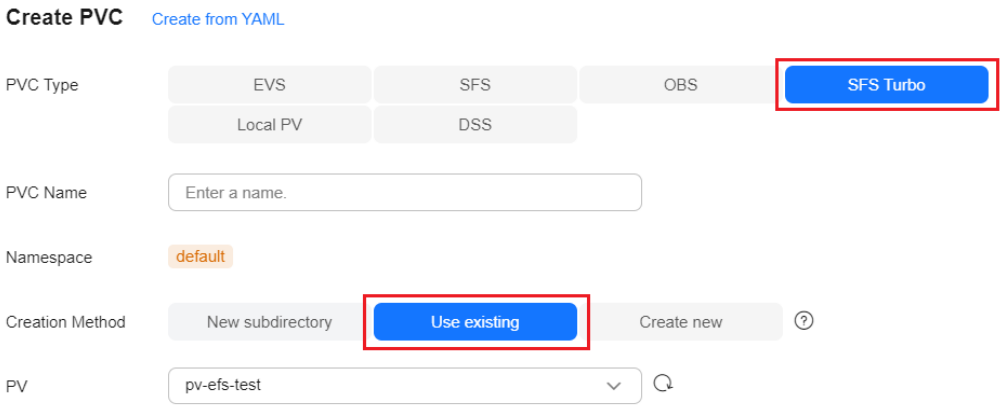


Step 2 After the PV is created, switch to the **PVCs** and click **Create PVC** in the upper right corner.

Configure the following parameters:

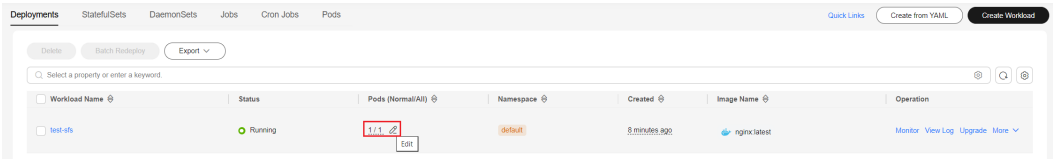
- **PVC Type**: Select **SFS Turbo**.

- **PVC Name**: Enter a custom name, which must be different from the original SFS 1.0 PVC name.
- **Creation Method**: Select **Use existing**.
- **PV**: Select the volume created in the previous step.



After the PVC is created, the following figure is displayed.

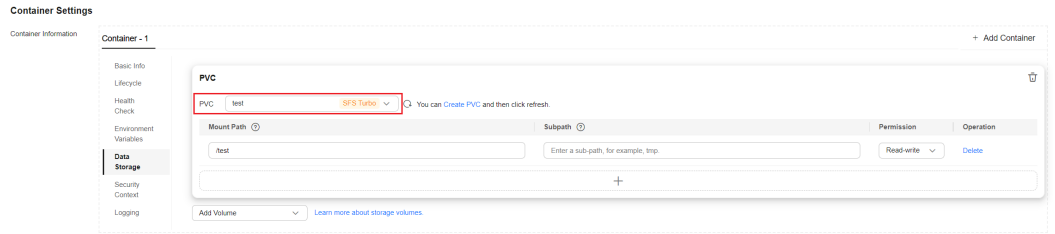
Step 3 Choose **Workloads** in the navigation pane. On the displayed page, locate the target workload and reduce the number of pods to 0.



Step 4 Click **Upgrade** in the **Operation** column of the workload. In **Container Settings**, switch to the **Data Storage** tab page and choose **PVC** from the drop-down list. In the displayed PVC area, select the PVC created in step **Step 2** to replace the PVC used by the workload.

NOTICE

After the migration, ensure that the mount path and subpath in the container are the same as those when SFS 1.0 is mounted.



Step 5 After the migration, you can increase the number of pods.

Check that the new volume works properly, and clear the SFS 1.0 storage volume on CCE.

----End

Migrating Data from Dynamically Mounted Storage Used by a StatefulSet

This section describes how to migrate data from dynamically mounted storage used by a StatefulSet from SFS 1.0 to SFS 3.0 or SFS Turbo.

Migrating to SFS Turbo

NOTICE

- The automatic scale-out capability of dynamic mounting is supported only by StatefulSets.
- SFS Turbo does not support dynamic provision. Therefore, after data of a StatefulSet is migrated from SFS 1.0 to SFS Turbo, the StatefulSet does not support the automatic scale-out capability of dynamic mounting.

Step 1 Choose **Workloads** in the navigation pane of the cluster console. Switch to the **StatefulSets** tab page, record the number of pods of the target workload, and reduce the number of pods to 0.

NOTE

Perform steps **Step 2** to **Step 6** for the PVC used by each pod.

Step 2 Check the PVC mounting mode of the StatefulSet.

```
kubectl get statefulset {statefulset-name} -n {namespace} -ojsonpath='{range .items[*]}{.spec.volumeClaimTemplates}{"\n"}{end}'
```

- If any command output is displayed, dynamic mounting is used. In this case, go to **Step 3** through **Step 7**.
- If there is no command output, dynamic mounting is not used. In this case, skip these steps and refer to **Migrating Data from Statically Mounted Storage**.

Step 3 Run the following command to change the **persistentVolumeReclaimPolicy** value of the PV corresponding to the PVC used by the pod from **Delete** to **Retain**:

```
kubectl edit pv {pv-name} -n {namespace}
```

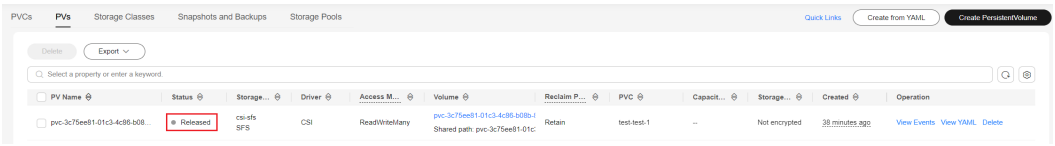
Check the result.


```
kubectl get pv {pv-name} -n {namespace} -o yaml |grep persistentVolumeReclaimPolicy
```

Example:

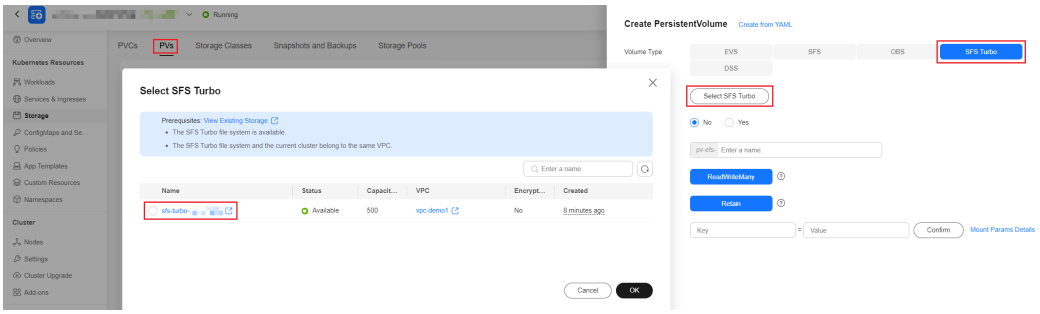
```
# kubectl get pv pvc-29467e4a-0120-4698-a147-5b75f0ae9a43 -o yaml |grep
persistentVolumeReclaimPolicy
  persistentVolumeReclaimPolicy: Retain
```

Step 4 On the **Storage** page, click the **PVs** tab. Then, record the PVC name corresponding to the SFS 1.0 PV and delete the PVC. The PV is in the **Released** state.

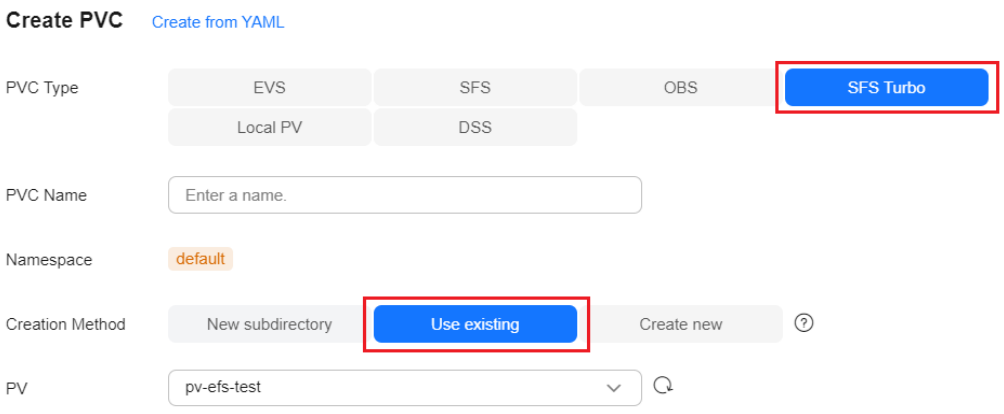


Step 5 Click **Create PersistentVolume** in the upper right corner and configure the following parameters:

- **Volume Type**: Select **SFS Turbo**.
- **SFS Turbo**: Select the target SFS Turbo volume for data migration.
- **PV Name**: Enter a custom name.
- **Access Mode**: Select **ReadWriteMany**.
- **Reclaim Policy**: Select **Retain**.



Step 6 Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner, create a PVC with the same name as that in **Step 4**, and bind the PVC to the SFS Turbo volume created in the previous step.



Step 7 After the PVCs corresponding to all pods are migrated, expand the number of pods to the original number.

After confirmation, go to the SFS console to delete the corresponding SFS 1.0 volume and delete the PV corresponding to SFS 1.0 on the CCE console.

----End

Migrating to SFS 3.0

NOTICE

- The automatic scale-out capability of dynamic mounting is supported only by StatefulSets.
- After a StatefulSet is migrated from SFS 1.0 to SFS 3.0, it still supports automatic scale-out.

To enable StatefulSets to support dynamic scale-out after the migration, change the StorageClass used by **volumeClaimTemplates** in StatefulSets from **csi-nas** to **csi-sfs**. The **volumeClaimTemplates** of stateful applications that use dynamic mounting cannot be modified. Therefore, delete the stateful applications and then rebuild them. During the process, ensure that the configurations, including the number of pods, are the same as those before the migration.

Step 1 Choose **Workloads** in the navigation pane. On the displayed page, locate the target workload, record the number of pods of the workload, and reduce the number of pods to 0.

NOTE

Perform steps **Step 2** to **Step 6** for the PVC used by each pod.

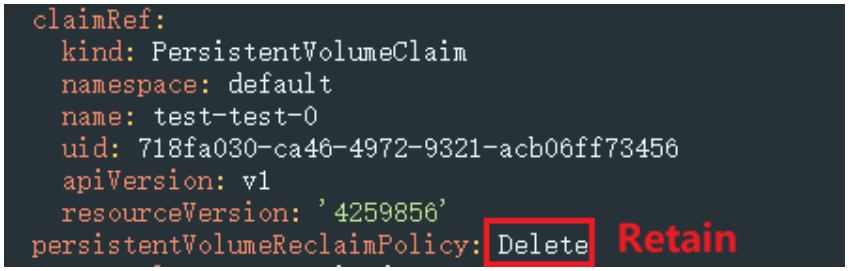
Step 2 Check the PVC mounting mode of the StatefulSet.

```
kubectl get statefulset {statefulset-name} -n {namespace} -ojsonpath='{range .items[*]}
{.spec.volumeClaimTemplates}{"\n"}{end}'
```

- If any command output is displayed, dynamic mounting is used. In this case, go to **Step 3** through **Step 7**.
- If there is no command output, dynamic mounting is not used. In this case, skip these steps and refer to **Migrating Data from Statically Mounted Storage**.

Step 3 Run the following command to change the **persistentVolumeReclaimPolicy** value of the SFS 1.0 PV from **Delete** to **Retain**:

```
kubectl edit pv {pv-name} -n {namespace}
```



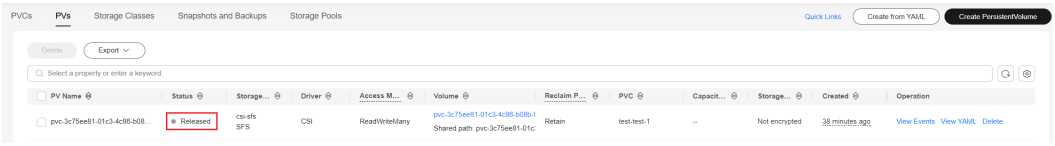
Check the result.

```
kubectl get pv {pv-name} -n {namespace} -o yaml |grep persistentVolumeReclaimPolicy
```

Example:

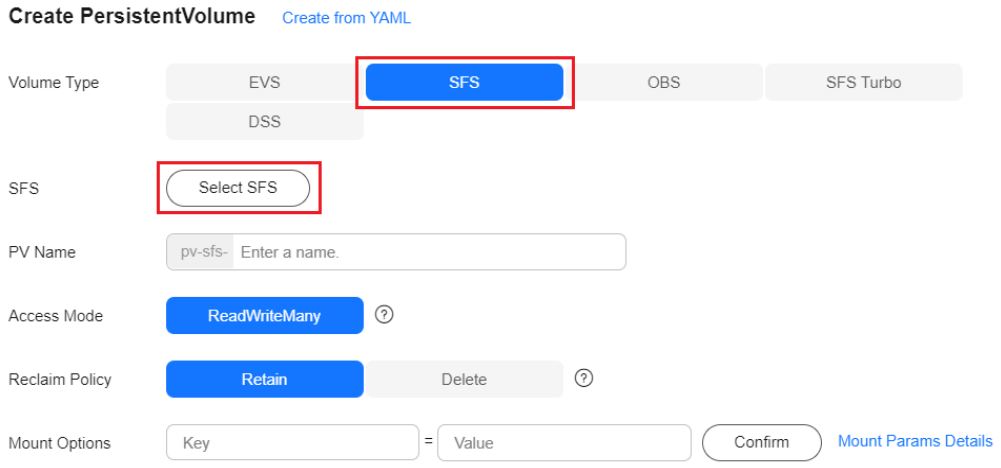
```
# kubectl get pv pvc-29467e4a-0120-4698-a147-5b75f0ae9a43 -o yaml |grep
persistentVolumeReclaimPolicy
  persistentVolumeReclaimPolicy: Retain
```

Step 4 On the **Storage** page, click the **PVs** tab. Then, record the PVC name corresponding to the SFS 1.0 PV and delete the PVC. The PV is in the **Released** state.



Step 5 Click **Create PersistentVolume** in the upper right corner and configure the following parameters:

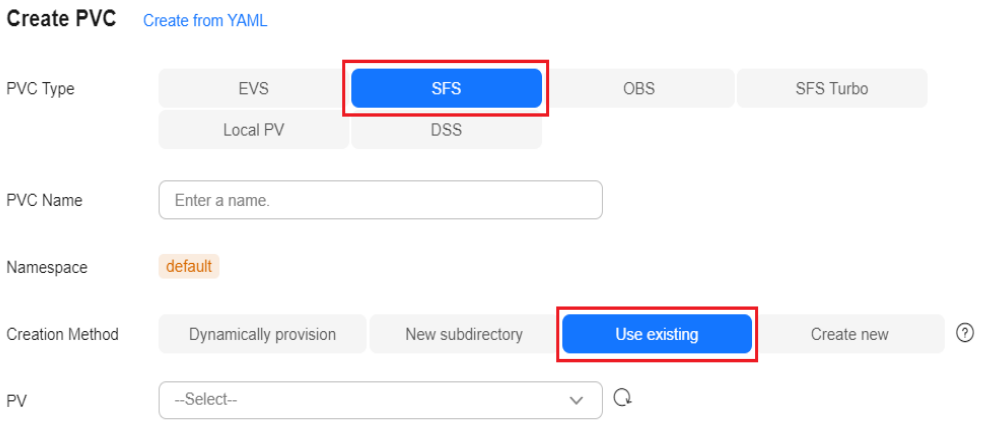
- **Volume Type**: Select **SFS**.
- **SFS**: Select the SFS 3.0 storage volume after data migration.
- **PV Name**: Enter a custom name.
- **Access Mode**: Select **ReadWriteMany**.
- **Reclaim Policy**: Set this parameter as required.
  - **Delete**: The PV will be removed from Kubernetes, and the associated storage assets will also be removed from the external infrastructure.
  - **Retain**: When the PVC is deleted, the PV will be retained, and the target data volume is marked released.



Step 6 Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner, create a PVC with the same name, and bind the PVC to the SFS 3.0 volume created in the previous step.

- **PVC Type**: Select **SFS**.
- **PVC Name**: Change it to the PVC with the same name in step **Step 4**.
- **Creation Method**: Select **Use existing**.

- **PV**: Select the volume created in the previous step.



Step 7 Go to the **Workloads** page to view the original stateful application. Choose **More** > **Edit YAML**, and click **Download** or copy all content of the YAML file to back up the file locally.

Step 8 Delete the original stateful application and modify the copied YAML configuration as follows:

- Change the value of **storageClassName** from **csi-nas** to **csi-sfs**.
- The **resourceVersion** field and its parameters are deleted because this field cannot be specified during creation.

```
volumeClaimTemplates:
  - apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      name: tets
      namespace: default
      labels:
        failure-domain.beta.kubernetes.io/region: cn-east-3
        failure-domain.beta.kubernetes.io/zone: cn-east-3a
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-sfs
podManagementPolicy: OrderedReady
```

Step 9 Click **Create from YAML** in the upper right corner, click **Import** or paste the modified YAML file content, and click **OK**.

Step 10 After the workload is created, scale-out the number of pods to the original number.

After confirmation, go to the SFS console to delete the corresponding SFS 1.0 volume and delete the PV corresponding to SFS 1.0 on the CCE console.

----End

8.5 SFS Turbo

8.5.1 Overview

Introduction

CCE allows you to mount storage volumes created by SFS Turbo file systems to a path of a container to meet data persistence requirements. SFS Turbo file systems are fast, on-demand, and scalable, which are suitable for scenarios with a massive number of small files, such as DevOps, containerized microservices, and enterprise office applications.

Expandable to 320 TB, SFS Turbo provides fully hosted shared file storage, which is highly available and stable, to support small files and applications requiring low latency and high IOPS.

- **Standard file protocols**: You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing**: The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network**: Users can access data only in private networks of data centers.
- **Data isolation**: The on-cloud storage service provides exclusive cloud file storage, which delivers data isolation and ensures IOPS performance.
- **Use cases**: Deployments/StatefulSets in the ReadWriteMany mode, DaemonSets, and jobs created for high-traffic websites, log storage, DevOps, and enterprise OA applications

SFS Turbo Performance

For details about the performance parameters of SFS Turbo, see **File System Types**.

Application Scenarios

SFS Turbo supports the following mounting modes:

- **Using an Existing SFS Turbo File System Through a Static PV**: static creation mode, where you use an existing SFS volume to create a PV and then mount storage to the workload through a PVC.
- **Dynamically Creating an SFS Turbo Subdirectory Using StorageClass**: SFS Turbo allows you to dynamically create subdirectories and mount them to containers so that SFS Turbo can be shared and the SFS Turbo storage capacity can be used more economically and properly.

Billing

SFS Turbo does not support dynamic creation. Only created SFS Turbo volumes can be mounted. You can select the pay-per-use billing mode or yearly/monthly package as required. For pricing details about SFS Turbo pricing, see **Billed Items**.

8.5.2 Using an Existing SFS Turbo File System Through a Static PV

SFS Turbo is a shared file system with high availability and durability. It is suitable for applications that contain massive small files and require low latency, and high IOPS. This section describes how to use an existing SFS Turbo file system to statically create PVs and PVCs for data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created an available SFS Turbo file system, and the SFS Turbo file system and the cluster are in the same VPC.

Notes and Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying SFS or SFS Turbo volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** value.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.
  - When the underlying volume is repeatedly used, enable isolation and protection for ReadWriteMany at the application layer to prevent data overwriting and loss.
- For SFS Turbo storage, the yearly/monthly SFS Turbo resources will not be reclaimed when the cluster or PVC is deleted. Reclaim the resources on the SFS Turbo console.

Using an Existing SFS Turbo File System on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS Turbo**.
PVC Name	Enter the PVC name, which must be unique in a namespace.

Parameter	Description
Creation Method	You can create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV has been created. In this example, select **Create new** to create both a PV and PVC on the console.
PV[a]	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in **Related Operations**. You do not need to specify this parameter in this example.
SFS Turbo[b]	Click **Select SFS Turbo**. On the displayed page, select the SFS Turbo file system that meets your requirements and click **OK**.
Subdirectory[b]	Determine whether to use subdirectories to create PVs. Enter the absolute path of a subdirectory, for example, **/a/b**. Ensure that the subdirectory is available.
PV Name[b]	Enter the PV name, which must be unique in the same cluster.
Access Mode[b]	SFS Turbo volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
Reclaim Policy[b]	Only **Retain** is available if you do not use subdirectories to create PVs. This indicates that the PV is not deleted when the PVC is deleted. For details, see **PV Reclaim Policy**. If you choose to use a subdirectory to create a PV, the value of this parameter can be **Delete**.
Subdirectory Reclaim Policy[b]	Determine whether to retain subdirectories when a PVC is deleted. This parameter must be used with **PV Reclaim Policy** and can be configured when **PV Reclaim Policy** is set to **Delete**. – **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. – **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring SFS Turbo Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

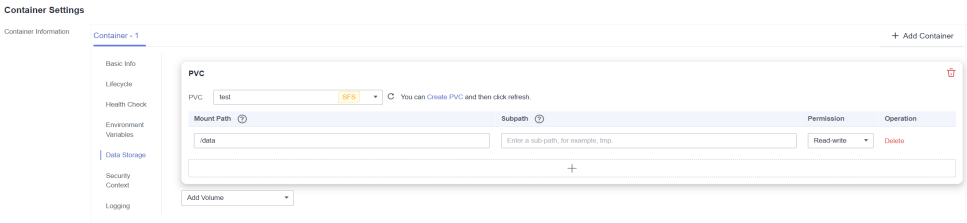
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-28**. For details about other parameters, see **Workloads**.

Table 8-28 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS Turbo volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the SFS Turbo file system.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Using an Existing SFS Turbo File System Through kubectl

You can select a mode that meets the requirements of your service.

Using an Existing SFS Turbo File System

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

   Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi     # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: <your_volume_id>   # SFS Turbo volume ID
    volumeAttributes:
      everest.io/share-export-location: <your_location>   # Shared path of the SFS Turbo volume
      everest.io/enterprise-project-id: <your_project_id>  # Project ID of the SFS Turbo volume
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy
  storageClassName: csi-sfsturbo         # StorageClass name of the SFS Turbo file system
  mountOptions: []                       # Mount options
```


Table 8-29 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	ID of an SFS Turbo file system for creating a PV. How to obtain: Log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. In the list, click the name of the target SFS Turbo file system. On the details page, copy the content following **ID**.
everest.io/share-export-location	Yes	Shared path of the SFS Turbo volume. Log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. You can obtain the shared path of the file system.
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, choose **SFS Turbo** in the navigation pane. Click the name of the SFS Turbo file system to interconnect. On the **Basic Info** tab, find and click the enterprise project to go to the console, and copy the ID.

mountOptions	No	Mount options.
		If not specified, the following configurations are used by default. For details, see **Configuring SFS Turbo Mount Options**.
		`mountOptions:`
		`- vers=3`
		`- timeo=600`
		`- nolock`
		`- hard`

persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. For details, see **PV Reclaim Policy**. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storage	Yes	Requested capacity in the PVC, in Gi.
storageClassName	Yes	The StorageClass name of SFS Turbo volumes is **csi-sfsturbo**.

2. Run the following command to create a PV:
```
kubectl apply -f pv-sfsturbo.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-sfsturbo.yaml** file.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/enterprise-project-id: <your_project_id>  # Project ID of the SFS Turbo volume
spec:
  accessModes:
  - ReadWriteMany               # The value must be ReadWriteMany for SFS Turbo.
  resources:
    requests:
      storage: 500Gi            # SFS Turbo volume capacity.
  storageClassName: csi-sfsturbo      # StorageClass name of the SFS Turbo file system, which must be the same as that of the PV
  volumeName: pv-sfsturbo    # PV name
```

Table 8-30 Key parameters

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, choose **SFS Turbo** in the navigation pane. Click the name of the SFS Turbo file system to interconnect. On the **Basic Info** tab, find and click the enterprise project to go to the console, and copy the ID.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in **1**. The StorageClass name of SFS Turbo volumes is **csi-sfsturbo**.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-sfsturbo.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS Turbo volume is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
```



```
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfsturbo-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfsturbo    # Name of the created PVC
```

2. Run the following command to create a workload to which the SFS Turbo volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Using Subdirectories of an Existing SFS Turbo File System

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only    # When a PVC is deleted, the PV will be deleted but the subdirectories associated with the PV will be retained. This parameter is available only when subdirectories are used and the reclaim policy is Delete.
  name: pv-sfsturbo    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi      # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: pv-sfsturbo   # SFS Turbo volume ID
    volumeAttributes:
      everest.io/share-export-location: <sfsturbo_path>:/<absolute_path>   # Shared path and subdirectory of the SFS Turbo file system
      everest.io/enterprise-project-id: <your_project_id>  # Project ID of the SFS Turbo volume
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/volume-as: absolute-path   # (Optional) An SFS Turbo subdirectory is used.
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy, which can be set to Delete when
```



```
subdirectories are automatically created
  storageClassName: csi-sfsturbo        # StorageClass name of the SFS Turbo file system
  mountOptions: []                      # Mount options
```

Table 8-31 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	PV name when an SFS Turbo subdirectory is used to create the PV.
everest.io/share-export-location	Yes	Shared path of the SFS Turbo subdirectory. Format: *{sfsturbo_path}:/{absolute_path}* How to obtain: Log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. You can obtain the shared path of the file system.
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, choose **SFS Turbo** in the navigation pane. Click the name of the SFS Turbo file system to interconnect. On the **Basic Info** tab, find and click the enterprise project to go to the console, and copy the ID.
mountOptions	No	Mount options. If not specified, the following configurations are used by default. For details, see **Configuring SFS Turbo Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. For details, see **PV Reclaim Policy**. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again. **Delete**: This parameter can be configured when subdirectories are automatically created, indicating that the PV is deleted when a PVC is deleted.

Parameter	Mandatory	Description
everest.io/reclaim-policy	No	Whether to retain subdirectories when deleting a PVC. This parameter must be used with **PV Reclaim Policy**. This parameter is available only when the PV reclaim policy is **Delete**. Options: – **retain-volume-only**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. – **delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
everest.io/volume-as	No	The value is fixed at **absolute-path**, indicating that a dynamically created SFS Turbo subdirectory is used. Ensure Everest of v2.3.23 or later has been installed in the cluster.
storage	Yes	Requested capacity in the PVC, in Gi. If a subdirectory is used, this parameter serves no purpose other than for verification and must have a non-empty, non-zero value.
storageClassName	Yes	The StorageClass name of SFS Turbo volumes is **csi-sfsturbo**.

2. Run the following command to create a PV:

```
kubectl apply -f pv-sfsturbo.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-sfsturbo.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/enterprise-project-id: <your_project_id>  # Project ID of the SFS Turbo volume
spec:
  accessModes:
  - ReadWriteMany               # The value must be ReadWriteMany for SFS Turbo.
  resources:
    requests:
      storage: 500Gi            # SFS Turbo volume capacity.
  storageClassName: csi-sfsturbo      # StorageClass name of the SFS Turbo file system, which must be the same as that of the PV
  volumeName: pv-sfsturbo    # PV name
```


Table 8-32 Key parameters

Parameter	Mandatory	Description
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, choose **SFS Turbo** in the navigation pane. Click the name of the SFS Turbo file system to interconnect. On the **Basic Info** tab, find and click the enterprise project to go to the console, and copy the ID.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in **1**. The StorageClass name of SFS Turbo volumes is **csi-sfsturbo**.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-sfsturbo.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the SFS Turbo volume is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-sfsturbo-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-sfsturbo-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-sfsturbo    # Name of the created PVC
```


2. Run the following command to create a workload to which the SFS Turbo volume is mounted:
```
kubectl apply -f web-demo.yaml
```
After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:
```
kubectl get pod | grep web-demo
```
Expected output:
```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```
2. Run the following commands in sequence to view the files in the **/data** path of the pods:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```
If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:
```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```
Expected output:
```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:
```
kubectl delete pod web-demo-846b489584-mjhm9
```
Expected output:
```
pod "web-demo-846b489584-mjhm9" deleted
```
After the deletion, the Deployment controller automatically creates a replica.
2. Run the following command to view the created pod:
```
kubectl get pod | grep web-demo
```
The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:
```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```
3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:
```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```
Expected output:
```
static
```
The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 Verify data sharing.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0          7m
   web-demo-846b489584-wvv5s   1/1     Running   0          13m
   ```
2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Check the files in the **/data** path of the pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   share
   static
   ```
3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Expected output:
   ```
   share
   static
   ```
   After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in **Table 8-33**.

Table 8-33 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Click **Create PersistentVolume** in the upper right corner. In the dialog box displayed, configure parameters. • **Volume Type**: Select **SFS Turbo**. • **SFS Turbo**: Click **Select SFS Turbo**. On the page displayed, select the SFS Turbo file system that meets your requirements and click **OK**. • **Subdirectory**: Determine whether to use subdirectories to create PVs. Enter the absolute path of a subdirectory, for example, **/a/b**. Ensure that the subdirectory is available. • **PV Name**: Enter the PV name, which must be unique in a cluster. • **Access Mode**: SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**. • **Reclaim Policy**: Only **Retain** is supported if you do not use subdirectories to create PVs. For details, see **PV Reclaim Policy**. If you choose to use a subdirectory to create a PV, the value of this parameter can be **Delete**. • **Subdirectory Reclaim Policy**: Determine whether to retain subdirectories when a PVC is deleted. This parameter must be used with **PV Reclaim Policy** and can be configured when **PV Reclaim Policy** is set to **Delete**. **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. • **Mount Options**: Enter the mounting parameter key-value pairs. For details,

Operation	Description	Procedure
		see **Configuring SFS Turbo Mount Options**. 2. Click **Create**.
Expanding the capacity of an SFS Turbo volume	Quickly expand the capacity of a mounted SFS Turbo volume on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.5.3 Configuring SFS Turbo Mount Options

This section describes how to configure SFS Turbo mount options. For SFS Turbo, you can only set mount options in a PV and bind the PV by creating a PVC.

Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

Notes and Constraints

- Mount options cannot be configured for Kata containers.
- Due to the restrictions of the NFS protocol, if an SFS volume is mounted to a node for multiple times, link-related mounting parameters (such as **timeo**) take effect only when the SFS volume is mounted for the first time by default. For example, if the same SFS file system is mounted to multiple pods running on a node, the mounting parameter set later does not overwrite the existing parameter value. If you want to configure different mounting parameters in the preceding scenario, additionally configure the **nosharecache** parameter.

SFS Turbo Mount Options

The Everest add-on in CCE presets the options described in **Table 8-34** for mounting SFS Turbo volumes.

Table 8-34 SFS Turbo mount options

Parameter	Value	Description
vers	3	File system version. Currently, only NFSv3 is supported. Value: **3**
nolock	Blank	Whether to lock files on the server using the NLM protocol. If **nolock** is selected, the lock is valid for applications on one host. For applications on another host, the lock is invalid.
timeo	600	Waiting time before the NFS client retransmits a request. The unit is 0.1 seconds. Recommended value: **600**
hard/soft	Blank	Mount mode. • **hard**: If the NFS request times out, the client keeps resending the request until the request is successful. • **soft**: If the NFS request times out, the client returns an error to the invoking program. The default value is **hard**.
sharecache/ nosharecache	Blank	How the data cache and attribute cache are shared when one file system is concurrently mounted to different clients. If this parameter is set to **sharecache**, the caches are shared between the mountings. If this parameter is set to **nosharecache**, the caches are not shared, and one cache is configured for each client mounting. The default value is **sharecache**. **NOTE** The **nosharecache** setting will affect the performance. The mounting information must be obtained for each mounting, which increases the communication overhead with the NFS server and the memory consumption of the NFS clients. In addition, the **nosharecache** setting on the NFS clients may lead to inconsistent caches. Determine whether to use **nosharecache** based on site requirements.

You can configure other mount options if needed. For details, see **Mounting an NFS File System to ECSs (Linux)**.

Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in **SFS Turbo Mount Options**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-sfsturbo    # PV name
spec:
  accessModes:
  - ReadWriteMany      # Access mode. The value must be ReadWriteMany for SFS Turbo.
  capacity:
    storage: 500Gi       # SFS Turbo volume capacity
  csi:
    driver: sfsturbo.csi.everest.io    # Dependent storage driver for the mounting
    fsType: nfs
    volumeHandle: {your_volume_id}   # SFS Turbo volume ID
    volumeAttributes:
      everest.io/share-export-location: {your_location}   # Shared path of the SFS Turbo volume
      everest.io/enterprise-project-id: {your_project_id} # Project ID of the SFS Turbo volume
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy
  storageClassName: csi-sfsturbo           # StorageClass name of the SFS Turbo file system
  mountOptions:                            # Mount options
  - vers=3
  - nolock
  - timeo=600
  - hard
```

Step 3 After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see **Using an Existing SFS Turbo File System Through a Static PV**.

Step 4 Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can run the **mount -l** command to check whether the mount options take effect.

1. View the pod to which the SFS Turbo volume has been mounted. In this example, the workload name is **web-sfsturbo**.

   ```
   kubectl get pod | grep web-sfsturbo
   ```

   Command output:

   ```
   web-sfsturbo-***  1/1     Running   0          23m
   ```

2. Run the following command to check the mount options (**web-sfsturbo-******** is an example pod):

   ```
   kubectl exec -it web-sfsturbo-*** -- mount -l | grep nfs
   ```

   If the mounting information in the command output is consistent with the configured mount options, the mount options have been configured.

   ```
   <Your mount path> on /data type nfs
   (rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,nolock,noresvport,proto=tcp,
   timeo=600,retrans=2,sec=sys,mountaddr=**.**.**.**,mountvers=3,mountport=20048,mountproto=tc
   p,local_lock=all,addr=**.**.**.**)
   ```

----End

8.5.4 (Recommended) Creating an SFS Turbo Subdirectory Using a Dynamic PV

When an SFS Turbo volume is mounted to a workload container, the root directory is mounted to the container by default. However, the minimum capacity of an SFS Turbo volume is 500 GiB, which exceeds the capacity required by most workloads, leading to a waste of storage resources. CCE enables efficient utilization of storage capacity by creating SFS Turbo subdirectories dynamically when you create a PVC. This allows multiple workloads to share the SFS Turbo file system.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on of v2.3.23 or later in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created an available SFS Turbo file system, and the SFS Turbo file system and the cluster are in the same VPC.

Dynamically Creating an SFS Turbo Subdirectory on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS Turbo**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	Select **New subdirectory**.
Storage Classes	Choose **csi-sfsturbo**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
Access Mode	SFS Turbo volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
SFS Turbo	Click **Select SFS Turbo**. On the displayed page, select the SFS Turbo file system that meets your requirements and click **OK**.
Subdirectory	Enter the absolute path of a subdirectory, for example, **/a/b**.

Parameter	Description
Subdirectory Reclaim Policy	Determine whether to retain subdirectories when a PVC is deleted. • **Retain**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. • **Delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.

Step 3 Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

----End

Dynamically Creating an SFS Turbo Subdirectory Using kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create the **pvc-sfsturbo-subpath.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfsturbo-subpath    # PVC name
  namespace: default
  annotations:
    everest.io/volume-as: absolute-path               # An SFS Turbo subdirectory is used.
    everest.io/sfsturbo-share-id: <sfsturbo_id>       # SFS Turbo ID
    everest.io/path: /a                               # Subdirectory that is automatically created, which must be an absolute path
    everest.io/reclaim-policy: retain-volume-only     # When a PVC is deleted, the PV will be deleted, but the subdirectories associated with the PV will be retained.
spec:
  accessModes:
    - ReadWriteMany      # ReadWriteMany must be selected for SFS Turbo.
  resources:
    requests:
      storage: 10Gi      # For SFS Turbo subdirectory PVCs, this configuration is meaningless and only used for verification (the value cannot be empty or 0).
  storageClassName: csi-sfsturbo     # StorageClass name of the SFS Turbo file system
```

Table 8-35 Key parameters

Parameter	Mandatory	Description
everest.io/volume-as	No	The value is fixed at **absolute-path**, indicating that a dynamically created SFS Turbo subdirectory is used.

Parameter	Mandatory	Description
everest.io/sfsturbo-share-id	No	SFS Turbo ID How to obtain: Log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. In the list, click the name of the target SFS Turbo file system. On the details page, copy the content following **ID**.
everest.io/path	No	Subdirectory that is automatically created, which must be an absolute path.
everest.io/reclaim-policy	No	Whether to retain subdirectories when deleting a PVC. This parameter must be used with **PV Reclaim Policy**. This parameter is available only when the PV reclaim policy is **Delete**. Options: • **retain-volume-only**: If a PVC is deleted, the PV will be deleted, but **the subdirectories associated with the PV will be retained**. • **delete**: After a PVC is deleted, **the PV and its associated subdirectories will also be deleted**. **NOTE** When a subdirectory is deleted, only the absolute path of the subdirectory configured in the PVC will be deleted. The upper-layer directory will not be deleted.
storage	Yes	Requested capacity in the PVC, in Gi. This parameter is only used for verification for SFS Turbo subdirectory PVCs. The value cannot be empty or **0**, and can be fixed at **10** because any value you set does not take effect.

Step 3 Run the following command to create a PVC:

```
kubectl apply -f pvc-sfsturbo-subpath.yaml
```

----End

8.5.5 Dynamically Creating an SFS Turbo Subdirectory Using StorageClass

Background

The minimum capacity of an SFS Turbo file system is 500 GiB, and the SFS Turbo file system cannot be billed by usage. By default, the root directory of an SFS Turbo file system is mounted to a container which, in most case, does not require such a large capacity.

The everest add-on allows you to dynamically create subdirectories in an SFS Turbo file system and mount these subdirectories to containers. In this way, an SFS Turbo file system can be shared by multiple containers to increase storage efficiency.

Notes and Constraints

- Only clusters of v1.15 or later are supported.
- The cluster must use the everest add-on of version 1.1.13 or later.
- Kata containers are not supported.
- When the everest add-on earlier than 1.2.69 or 2.1.11 is used, a maximum of 10 PVCs can be created concurrently at a time by using the subdirectory function. everest of 1.2.69 or later or of 2.1.11 or later is recommended.
- A subPath volume is a subdirectory of an SFS Turbo file system. Increasing the capacity of a PVC of this type only changes the resource range specified by the PVC, but does not change the total capacity of the SFS Turbo file system. If the SFS Turbo file system's total resource capacity is not enough, the available capacity of the subPath volume will be restricted. To fix this, you must increase the resource capacity of the SFS Turbo file system on the SFS Turbo console.

  Deleting the subPath volume does not result in the deletion of the resources of the SFS Turbo file system.

Creating an SFS Turbo Volume of the subPath Type

Step 1 Create an SFS Turbo file system in the same VPC and subnet as the cluster.

Step 2 Create a YAML file of StorageClass, for example, **sfsturbo-subpath-sc.yaml**.

The following is an example:

```
apiVersion: storage.k8s.io/v1
allowVolumeExpansion: true
kind: StorageClass
metadata:
  name: sfsturbo-subpath-sc # Storage class name
mountOptions:  #Mount options
- lock
parameters:
  csi.storage.k8s.io/csi-driver-name: sfsturbo.csi.everest.io
  csi.storage.k8s.io/fstype: nfs
  everest.io/archive-on-delete: "true"
  everest.io/share-access-to: 7ca2dba2-1234-1234-1234-626371a8fb3a
  everest.io/share-expand-type: bandwidth
  everest.io/share-export-location: 192.168.1.1:/sfsturbo/  # Mount directory configuration
  everest.io/share-source: sfs-turbo
  everest.io/share-volume-type: STANDARD
  everest.io/volume-as: subpath
  everest.io/volume-id: 0d773f2e-1234-1234-1234-de6a35074696  # ID of an SFS Turbo volume
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

In this example:

- **name**: indicates the name of the StorageClass.
- **mountOptions**: indicates the mount options. This field is optional.
  - In versions later than everest 1.1.13 and earlier than everest 1.2.8, only the **nolock** parameter can be configured. By default, the **nolock**

parameter is used for the mount operation and does not need to be configured. If **nolock** is set to **false**, the **lock** field is used.

– Starting from everest 1.2.8, more mount options are supported. For details, see **Configuring SFS Turbo Mount Options**. **Do not set nolock to true. Otherwise, the mount operation will fail.**

```
mountOptions:
- vers=3
- timeo=600
- nolock
- hard
```

- **everest.io/volume-as**: This parameter is set to **subpath** to use the subPath volume.
- **everest.io/share-access-to**: This parameter is optional. In a subPath volume, set this parameter to the ID of the VPC where the SFS Turbo file system is located.
- **everest.io/share-expand-type**: This parameter is optional. If the type of the SFS Turbo file system is SFS Turbo Standard – Enhanced or SFS Turbo Performance – Enhanced, set this parameter to **bandwidth**.
- **everest.io/share-export-location**: This parameter indicates the mount directory. It consists of the SFS Turbo shared path and sub-directory. The shared path can be obtained on the SFS Turbo console. The sub-directory is user-defined. The PVCs created using the StorageClass are located in this sub-directory.
- **everest.io/share-volume-type**: This parameter is optional. It specifies the SFS Turbo file system type. The value can be **STANDARD** or **PERFORMANCE**. For enhanced types, this parameter must be used together with **everest.io/share-expand-type** (whose value should be **bandwidth**).
- **everest.io/zone**: This parameter is optional. Set it to the AZ where the SFS Turbo file system is located.
- **everest.io/volume-id**: This parameter indicates the ID of the SFS Turbo volume. You can obtain the volume ID on the SFS Turbo page.
- **everest.io/archive-on-delete**: If this parameter is set to **true** and **Delete** is selected for **Reclaim Policy**, the original documents of the PV will be archived to the directory named **archived-***{$PV name.timestamp}* before the PVC is deleted. If this parameter is set to **false**, the SFS Turbo subdirectory of the corresponding PV will be deleted. The default value is **true**, indicating that the original documents of the PV will be archived to the directory named **archived-***{$PV name.timestamp}* before the PVC is deleted.

Step 3 Run **kubectl create -f sfsturbo-subpath-sc.yaml**.

Step 4 Create a PVC YAML file named **sfs-turbo-test.yaml**.

The following is an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sfs-turbo-test  # PVC name
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
```



```
storageClassName: sfsturbo-subpath-sc  # Storage class name
volumeMode: Filesystem
```

In this example:

- **name**: indicates the name of the PVC.
- **storageClassName**: specifies the name of the StorageClass.
- **storage**: In a subPath volume, modifying the value of this parameter does not impact the resource capacity of the SFS Turbo file system. A subPath volume is essentially a file path within an SFS Turbo file system. As a result, increasing the capacity of the subPath volume in a PVC does not lead to an increase in the resources of the SFS Turbo file system.

  NOTE

  The capacity of a subPath volume is restricted by the overall resource capacity of the corresponding SFS Turbo file system. If the resources of the SFS Turbo file system are inadequate, you can adjust the resource capacity via the SFS Turbo console.

Step 5 Run **kubectl create -f sfs-turbo-test.yaml**.

----End

Creating a Deployment and Mounting an Existing Volume

Step 1 Create a YAML file for the Deployment, for example, **deployment-test.yaml**.

The following is an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-turbo-subpath-example   # Name of the created workload
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-turbo-subpath-example
  template:
    metadata:
      labels:
        app: test-turbo-subpath-example
    spec:
      containers:
      - image: nginx:latest  # Image of the workload
        name: container-0
        volumeMounts:
        - mountPath: /tmp   #Mount path in a container
          name: pvc-sfs-turbo-example
      restartPolicy: Always
      imagePullSecrets:
      - name: default-secret
      volumes:
      - name: pvc-sfs-turbo-example
        persistentVolumeClaim:
          claimName: sfs-turbo-test   # Name of an existing PVC
```

In this example:

- **name**: indicates the name of the created workload.

- **image**: specifies the image used by the workload.
- **mountPath**: indicates the mount path of the container. In this example, the volume is mounted to the **/tmp** directory.
- **claimName**: indicates the name of an existing PVC.

Step 2 Create the Deployment.

kubectl create -f deployment-test.yaml

----End

Dynamically Creating a subPath Volume for a StatefulSet

Step 1 Create a YAML file for a StatefulSet, for example, **statefulset-test.yaml**.

The following is an example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: test-turbo-subpath # Name of the created workload
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test-turbo-subpath
  template:
    metadata:
      labels:
        app: test-turbo-subpath
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:latest'  # Image of the workload
          resources: {}
          volumeMounts:
            - name: sfs-turbo-160024548582479676
              mountPath: /tmp  # Mount path in a container
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: sfs-turbo-160024548582479676
        namespace: default
        annotations: {}
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
```



```
          storage: 10Gi
      storageClassName: sfsturbo-subpath-sc # Enter the name of a self-managed storage class.
serviceName: wwww
podManagementPolicy: OrderedReady
updateStrategy:
  type: RollingUpdate
revisionHistoryLimit: 10
```

In this example:

- **name**: indicates the name of the created workload.
- **image**: specifies the image used by the workload.
- **mountPath**: indicates the mount path of the container. In this example, the volume is mounted to the **/tmp** directory.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name**: must be consistent because they have a mapping relationship.
- **storageClassName**: specifies the name of an on-premises StorageClass.

Step 2 Create the StatefulSet.

kubectl create -f statefulset-test.yaml

----End

8.6 Object Storage Service

8.6.1 Overview

Introduction

Object Storage Service (OBS) provides massive, secure, and cost-effective data storage for you to store data of any type and size. You can use it in enterprise backup/archiving, video on demand (VoD), video surveillance, and many other scenarios.

- **Standard APIs**: With HTTP RESTful APIs, OBS allows you to use client tools or third-party tools to access object storage.
- **Data sharing**: Servers, embedded devices, and IoT devices can use the same path to access shared object data in OBS.
- **Public/Private networks**: OBS allows data to be accessed from public networks to meet Internet application requirements.
- **Capacity and performance**: No capacity limit; high performance (I/O latency within 10 ms).
- **Use cases**: Deployments/StatefulSets in the **ReadOnlyMany** mode and jobs created for big data analysis, static website hosting, online VOD, gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks). You can create object storage by using the OBS console, tools, and SDKs.

OBS Specifications

OBS provides multiple storage classes to meet customers' requirements on storage performance and costs.

- Object buckets provide reliable, high-performance, secure, and budget-friendly storage for data. They have no restrictions on the quantity of files or storage capacity.
  - Standard: features low latency and high throughput. It is therefore good for storing frequently (multiple times per month) accessed files or small files (less than 1 MB). Its application scenarios include big data analytics, mobile apps, hot videos, and social apps.
  - OBS Infrequent Access: applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response. Its application scenarios include file synchronization or sharing, and enterprise-level backup. This storage class has the same durability, low latency, and high throughput as the Standard storage class, with a lower cost, but its availability is slightly lower than the Standard storage class.
- Parallel file systems are a sub-product of OBS designed to provide high-performance file semantics for big data scenarios. For details, see **Overview of PFS**.

For details about OBS storage classes, see **Storage Class**.

Performance

Every time an OBS volume is mounted to a container workload, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. The amount of memory required in these scenarios is listed **Table 8-36**. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than** 4.

Table 8-36 Memory required by a single object storage resident process

Test Item	Memory Usage
Long-term stable running	About 50 MiB
Concurrent write to a 10 MB file from two processes	About 110 MiB
Concurrent write to a 10 MB file from four processes	About 220 MiB
Write to a 100 GB file from a single process	About 300 MiB

Application Scenarios

OBS supports the following mounting modes based on application scenarios:

- **Using an Existing OBS Bucket Through a Static PV**: static creation mode, where you use an existing OBS volume to create a PV and then mount storage to the workload through a PVC. This mode applies if the underlying storage is available or billed on a yearly/monthly basis.
- **Using an OBS Bucket Through a Dynamic PV**: dynamic creation mode, in which you do not need to create OBS volumes beforehand. Instead, specify a StorageClass when creating a PVC. Then, an OBS volume and PV will be created automatically. This mode applies to scenarios where no underlying storage is available.

Billing

- The default billing mode of an OBS volume **automatically created** using StorageClass is **pay-per-use**. For details about OBS pricing, see **OBS Pricing Details**.
- To use a yearly/monthly-billed OBS volume, **use an existing one**.

8.6.2 Using an Existing OBS Bucket Through a Static PV

This section describes how to use an existing Object Storage Service (OBS) bucket to statically create PVs and PVCs for data persistence and sharing in workloads.

Prerequisites

- You have created a cluster and installed the **CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- Every time an OBS volume is mounted to a workload through a PVC, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than** 4.
- Secure containers do not support OBS volumes.
- Hard links are not supported when common buckets are mounted.
- Multiple PVs can use the same OBS storage volume with the following restrictions:
  - Do not mount the PVCs/PVs that use the same underlying OBS volume to one pod. This will lead to a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** value.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs must be set to **Retain**. Otherwise, when a PV is deleted, the associated underlying volume may be deleted. In this case, other PVs associated with the underlying volume malfunction.

- If underlying storage is repeatedly used, you are required to maintain data consistency. Enable isolation and protection for ReadWriteMany at the application layer and prevent multiple clients from writing the same file to prevent data overwriting and loss.

Using an Existing OBS Bucket on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **OBS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using an OBS Bucket Through a Dynamic PV**. In this example, select **Create new** to create both a PV and PVC on the console.
PV[a]	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in **Related Operations**. You do not need to specify this parameter in this example.
OBS[b]	Click **Select OBS**. On the displayed page, select the OBS volume that meets your requirements and click **OK**.
PV Name[b]	Enter the PV name, which must be unique in the same cluster.
Access Mode[b]	OBS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
Reclaim Policy[b]	You can select **Delete** or **Retain** to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see **PV Reclaim Policy**. **NOTE** If multiple PVs use the same OBS volume, use **Retain** to prevent the underlying volume from being deleted with a PV.

Parameter	Description
Access Key (AK/SK)[b]	**Custom** (Recommended): Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. – **Name**: Enter a secret name. – **Namespace**: Select the namespace where the secret is. – **Access Key (AK/SK)**: Upload a key file in .csv format. For details, see **Obtaining an Access Key**.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring OBS Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

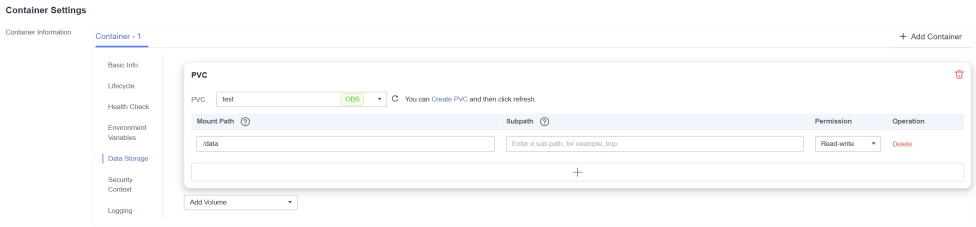
   Mount and use storage volumes, as shown in **Table 8-37**. For details about other parameters, see **Workloads**.

Table 8-37 Mounting a storage volume

Parameter	Description
PVC	Select an existing OBS volume.

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the OBS volume.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Using an Existing OBS Bucket Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-obs.yaml** file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-obs       # PV name
spec:
  accessModes:
  - ReadWriteMany    # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi     # OBS volume capacity
  csi:
    driver: obs.csi.everest.io        # Dependent storage driver for the mounting
    fsType: obsfs                     # Instance type
    volumeHandle: <your_volume_id>    # Name of the OBS volume
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                    # Region where the OBS volume is
      everest.io/enterprise-project-id: <your_project_id>     # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
    nodePublishSecretRef:           # Custom secret of the OBS volume
      name: <your_secret_name>       # Custom secret name
      namespace: <your_namespace>    # Namespace of the custom secret
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy
  storageClassName: csi-obs              # StorageClass name
  mountOptions: []                       # Mount options
```

Table 8-38 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Only **retain-volume-only** is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is **Delete**. If the reclaim policy is **Delete** and the current value is **retain-volume-only**, the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
fsType	Yes	Instance type. The value can be **obsfs** or **s3fs**. – **obsfs**: a parallel file system – **s3fs**: object bucket
volumeHandle	Yes	OBS volume name.

Parameter	Mandatory	Description
everest.io/obs-volume-type	Yes	OBS StorageClass. – If **fsType** is set to **s3fs**, **STANDARD** (standard buckets) and **WARM** (infrequent access buckets) are supported. – This parameter is invalid when **fsType** is set to **obsfs**.
everest.io/region	Yes	Region where the OBS bucket is deployed. For details about its value, see **Regions and Endpoints**.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of OBS. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. **How to obtain**: On the OBS console, choose **Buckets** or **Parallel File Systems** in the navigation pane. Click the name of the OBS bucket to access its details page. In the **Basic Information** area, locate the enterprise project and click it to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the object storage belongs.

nodePublishSecretRef	No	Access key (AK/SK) used for mounting the object storage volume. You can use the AK/SK to create a secret and mount it to the PV. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.
		Example:
		`nodePublishSecretRef:`
		`  name: secret-demo`
		`  namespace: default`

mountOptions	No	Mount options. For details, see **Configuring OBS Mount Options**.

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	A reclaim policy is supported when the cluster version is or later than 1.19.10 and the Everest version is or later than 1.2.9. The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If multiple PVs use the same OBS volume, use **Retain** to prevent the underlying volume from being deleted with a PV. **Delete**: – If **everest.io/reclaim-policy** is not specified, both the PV and storage resources will be deleted when a PVC is deleted. – If **everest.io/reclaim-policy** is set to **retain-volume-only**, when a PVC is deleted, the PV will be deleted but the storage resources will be retained. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storage	Yes	Storage capacity, in Gi. For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at **1**, and any value you set does not take effect for OBS.
storageClassName	Yes	StorageClass name, which is **csi-obs** for an OBS volume.

2. Run the following command to create a PV:
```
kubectl apply -f pv-obs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-obs.yaml** file.
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>  # Custom secret name
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>       # Namespace of the
custom secret
```



```
  everest.io/enterprise-project-id: <your_project_id>     # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
spec:
  accessModes:
  - ReadWriteMany               # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs      # StorageClass name, which must be the same as that of the PV
  volumeName: pv-obs    # PV name
```

Table 8-39 Key parameters

Parameter	Mandatory	Description
csi.storage.k8s.io/node-publish-secret-name	No	Name of the custom secret specified in the PV.
csi.storage.k8s.io/node-publish-secret-namespace	No	Namespace of the custom secret specified in the PV.
everest.io/enterprise-project-id	No	Project ID of OBS. **How to obtain**: On the OBS console, choose **Buckets** or **Parallel File Systems** in the navigation pane. Click the name of the OBS bucket to access its details page. In the **Basic Information** area, locate the enterprise project and click it to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the object storage belongs.
storage	Yes	Requested capacity in the PVC, in Gi. For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at **1**, and any value you set does not take effect for OBS.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in **1**. StorageClass name, which is **csi-obs** for an OBS volume.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-obs.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the OBS volume is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-obs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-obs    # Name of the created PVC
```

2. Run the following command to create a workload to which the OBS volume is mounted:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```


Step 4 Verify data persistence.

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Expected output:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   After the deletion, the Deployment controller automatically creates a replica.
2. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:
   ```
   web-demo-846b489584-d4d4j  1/1   Running  0          110s
   web-demo-846b489584-wvv5s  1/1   Running  0          7m50s
   ```
3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   static
   ```
   The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 Verify data sharing.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-d4d4j  1/1   Running  0          7m
   web-demo-846b489584-wvv5s  1/1   Running  0          13m
   ```
2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.
   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```
   Check the files in the **/data** path of the pod.
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   share
   static
   ```
3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.
   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   Expected output:
   ```
   share
   static
   ```
   After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in **Table 8-40**.

Table 8-40 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Click **Create PersistentVolume** in the upper right corner. In the dialog box displayed, configure parameters. • **Volume Type**: Select **OBS**. • **OBS**: Click **Select OBS**. On the displayed page, select the OBS volume that meets your requirements and click **OK**. • **PV Name**: Enter the PV name, which must be unique in a cluster. • **Access Mode**: SFS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**. • **Reclaim Policy**: **Delete** or **Retain** is supported. For details, see **PV Reclaim Policy**. **NOTE** If multiple PVs use the same underlying storage volume, use **Retain** to prevent the underlying volume from being deleted with a PV. • **Access Key (AK/SK)**: (Recommended) Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. • **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **Configuring OBS Mount Options**. 2. Click **Create**.

Operation	Description	Procedure
Updating an access key	Update the access key of object storage on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Update Access Key**. 2. Upload a key file in .csv format. For details, see **Obtaining an Access Key**. Click **OK**. **NOTE** After a global access key is updated, all pods mounted with the object storage that uses this access key can be accessed only after being restarted.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.6.3 Using an OBS Bucket Through a Dynamic PV

This section describes how to automatically create an OBS bucket. It is applicable when no underlying storage volume is available.

Notes and Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- Every time an OBS volume is mounted to a workload through a PVC, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be **no more than** 4.
- Secure containers do not support OBS volumes.

- Hard links are not supported when common buckets are mounted.
- OBS allows a single user to create a maximum of 100 buckets. If a large number of dynamic PVCs are created, the number of buckets may exceed the upper limit, and no more OBS buckets can be created. In this case, use OBS by calling its API or SDK and do not mount OBS buckets to workloads.

Automatically Creating an OBS Volume on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **OBS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If no underlying storage is available, select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode. – If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see **Using an Existing OBS Bucket Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The default StorageClass for OBS volumes is **csi-obs**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.

Parameter	Description
Instance Type	– **Parallel file system**: a high-performance file system provided by OBS. It provides millisecond-level access latency, TB/s-level bandwidth, and million-level IOPS. – **Object bucket**: provides reliable, high-performance, secure, and budget-friendly storage for data. They have no restrictions on the quantity of files or storage capacity.
OBS Class	You can select the following object bucket types: – **Standard**: Applicable when a large number of hotspot files or small-sized files need to be accessed frequently (multiple times per month on average) and require fast access response. – **Infrequent access**: Applicable when data is not frequently accessed (fewer than 12 times per year on average) but requires fast access response.
Data Redundancy Policy	The Everest version in the cluster must be 2.4.14 or later. – **Multi-AZ storage**: Redundant data is stored in multiple AZs for higher reliability but at relatively high costs. For details, see **Price Calculator**. – **Single-AZ storage**: Data is stored in a single AZ, with lower costs.
Access Mode	OBS volumes support only **ReadWriteMany**, indicating that a storage volume can be mounted to multiple nodes in read/write mode. For details, see **Volume Access Modes**.
Access Key (AK/SK)	**Custom** (Recommended): Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. – **Name**: Enter a secret name. – **Namespace**: Select the namespace where the secret is. – **Access Key (AK/SK)**: Upload a key file in .csv format. For details, see **Obtaining an Access Key**.
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

2. Click **Create** to create a PVC and a PV.

You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

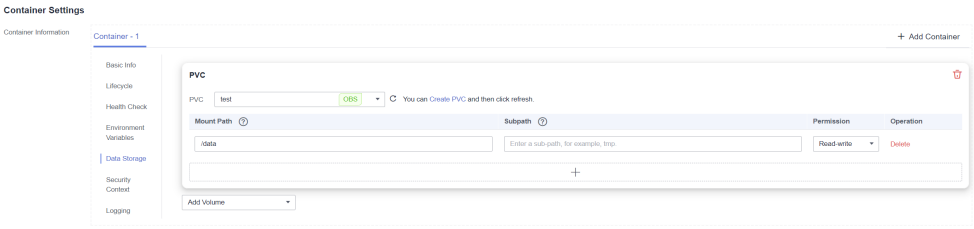
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-41**. For details about other parameters, see **Workloads**.

Table 8-41 Mounting a storage volume

Parameter	Description
PVC	Select an existing OBS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the OBS volume.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Automatically Creating an OBS Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-obs-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs-auto
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD    # OBS
    csi.storage.k8s.io/fstype: obsfs        # Instance type
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>       # Custom secret name
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>    # Namespace of the
custom secret
    everest.io/enterprise-project-id: <your_project_id>     # (Optional) Enterprise project ID. If an
enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the
PVC cannot be bound to a PV.
    everest.io/csi.obs-az-redundancy: 'true'   # (Optional) Multi-AZ object storage
    everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the
automatically created underlying storage
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi              # OBS volume capacity
  storageClassName: csi-obs     # The StorageClass is OBS.
```

Table 8-42 Key parameters

Parameter	Mandatory	Description
everest.io/obs-volume-type	Yes	OBS StorageClass. – If **fsType** is set to **s3fs**, **STANDARD** (standard buckets) and **WARM** (infrequent access buckets) are supported. – This parameter is invalid when **fsType** is set to **obsfs**.

Parameter	Mandatory	Description
csi.storage.k8s.io/fstype	Yes	Instance type. The value can be **obsfs** or **s3fs**. – **obsfs**: a parallel file system – **s3fs**: object bucket
csi.storage.k8s.io/node-publish-secret-name	No	Custom secret name. (Recommended) Select this option if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.
csi.storage.k8s.io/node-publish-secret-namespace	No	Namespace of a custom secret.
everest.io/enterprise-project-id	No	Project ID of OBS. To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
everest.io/csi.obs-az-redundancy	No	Value **true** indicates that redundant data is stored in multiple AZs for higher reliability but at relatively high costs. For details, see **Price Calculator**. The Everest version in the cluster must be 2.4.14 or later.
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-***{UID}*.

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi. For OBS, this field is used only for verification (cannot be empty or 0). Its value is fixed at **1**, and any value you set does not take effect for OBS.
storageClassName	Yes	StorageClass name, which is **csi-obs** for an OBS volume.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-obs-auto.yaml
```

Step 3 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the OBS volume is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-obs-volume    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-obs-volume    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-obs-auto    # Name of the created PVC
```

2. Run the following command to create a workload to which the OBS volume is mounted:
```
kubectl apply -f web-demo.yaml
```
After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:


```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-mjhm9   1/1     Running   0          46s
web-demo-846b489584-wvv5s   1/1     Running   0          46s
```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

Step 4 Verify data persistence.

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

```
kubectl delete pod web-demo-846b489584-mjhm9
```

Expected output:

```
pod "web-demo-846b489584-mjhm9" deleted
```

After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

```
web-demo-846b489584-d4d4j   1/1     Running   0          110s
web-demo-846b489584-wvv5s   1/1     Running   0          7m50s
```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 Verify data sharing.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-demo
```

Expected output:

```
web-demo-846b489584-d4d4j   1/1     Running   0          7m
web-demo-846b489584-wvv5s   1/1     Running   0          13m
```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:


```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations listed in **Table 8-43**.

Table 8-43 Related operations

Operation	Description	Procedure
Updating an access key	Update the access key of object storage on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Update Access Key**. 2. Upload a key file in .csv format. For details, see **Obtaining an Access Key**. Click **OK**. **NOTE** After a global access key is updated, all pods mounted with the object storage that uses this access key can be accessed only after being restarted.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.6.4 Configuring OBS Mount Options

This section describes how to configure OBS volume mount options. You can configure mount options in a PV and bind the PV to a PVC. Alternatively, configure mount options in a StorageClass and use the StorageClass to create a PVC. In this way, PVs can be dynamically created and inherit mount options configured in the StorageClass by default.

Prerequisites

The **CCE Container Storage (Everest)** version must be **1.2.8 or later**. This add-on identifies the mount options and transfers them to the underlying storage resources. The parameter settings take effect only if the underlying storage resources support the specified options.

Notes and Constraints

Mount options cannot be configured for Kata containers.

OBS Mount Options

When mounting an OBS volume, the Everest add-on presets the options described in **Table 8-44** and **Table 8-45** by default. The options in **Table 8-44** are mandatory.

Table 8-44 Mandatory mount options configured by default

Parameter	Supported Object Storage Type	Value	Description
use_ino	Parallel file system	Blank	If enabled, obsfs allocates the **inode** number. Enabled by default in read/write mode.
big_writes	Parallel file system Object bucket	Blank	If enabled, the maximum size of the write cache can be changed.
nonempty	Parallel file system Object bucket	Blank	Allows non-empty mount paths.
allow_other	Parallel file system Object bucket	Blank	Allows other users to access the mount directory.
no_check_certificate	Parallel file system Object bucket	Blank	Disables server certificate verification.

Parameter	Supported Object Storage Type	Value	Description
enable_noobj_cache	Object bucket	Blank	Enables cache entries for objects that do not exist, which can improve performance. Enabled by default in object bucket read/write mode. **This option is no longer configured by default since Everest 1.2.40.**
sigv2	Object bucket	Blank	Specifies the signature version. Used by default in object buckets.
public_bucket	Object bucket	1	If this parameter is set to **1**, public buckets are mounted anonymously. Enabled by default in object bucket read-only mode.
compat_dir	Object bucket	Blank	This parameter is automatically added when s3fs v1.92 is used. Multi-level directory objects in a bucket can be displayed.

Table 8-45 Optional mount options configured by default

Parameter	Supported Object Storage Type	Value	Description
max_write	Parallel file system Object bucket	131072	This parameter is valid only when **big_writes** is configured. The recommended value is **128 KB**.
multipart_size	Object bucket	20	If a file is uploaded in multiple parts, the size of each part, measured in MB, will affect the size of the file that can be uploaded.
ssl_verify_hostname	Parallel file system Object bucket	0	Disables SSL certificate verification based on the host name.
max_background	Parallel file system Object bucket	100	Allows setting the maximum number of waiting requests in the background. Used by default in parallel file systems.

Parameter	Supported Object Storage Type	Value	Description
umask	Parallel file system Object bucket	0	Mask of the configuration file permission. For example, if the umask value is **022**, the directory permission (the maximum permission is **777**) is **755** (777 - 022 = 755, rwxr-xr-x).

Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in **OBS Mount Options**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-obs       # PV name
spec:
  accessModes:
  - ReadWriteMany    # Access mode. The value must be ReadWriteMany for OBS.
  capacity:
    storage: 1Gi     # OBS volume capacity
  csi:
    driver: obs.csi.everest.io        # Dependent storage driver for the mounting
    fsType: obsfs                     # Instance type
    volumeHandle: <your_volume_id>    # Name of the OBS volume
    volumeAttributes:
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <your_region>                    # Region where the OBS volume is
      everest.io/enterprise-project-id: <your_project_id>     # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
    nodePublishSecretRef:           # Custom secret of the OBS volume
      name: <your_secret_name>       # Custom secret name
      namespace: <your_namespace>    # Namespace of the custom secret
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy
  storageClassName: csi-obs              # StorageClass name
  mountOptions:                          # Mount options
  - umask=027
```

Step 3 After a PV is created, you can create a PVC and bind it to the PV, and then mount the PV to the container in the workload. For details, see **Using an Existing OBS Bucket Through a Static PV**.

Step 4 Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can log in to the node where the pod to which the OBS volume is mounted resides and view the progress details.

Run the following command:

- Object bucket: **ps -ef | grep s3fs**

```
root     22142     1  0 Jun03 ?        00:00:00 /usr/bin/s3fs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o no_check_certificate -o ssl_verify_hostname=0 -o umask=027 -o max_write=131072 -o multipart_size=20
```

- Parallel file system: **ps -ef | grep obsfs**

```
root     1355     1  0 Jun03 ?        00:03:16 /usr/bin/obsfs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o umask=027 -o max_write=131072
```

----End

Configuring Mount Options in a StorageClass

You can use the **mountOptions** field to configure mount options in a StorageClass. The options you can configure in **mountOptions** are listed in **OBS Mount Options**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a custom StorageClass. Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-obs-mount-option
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: obs.csi.everest.io
  csi.storage.k8s.io/fstype: s3fs
  everest.io/obs-volume-type: STANDARD
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:                   # Mount options
- umask=027
```

Step 3 After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options configured in the StorageClass. For details, see **Using an OBS Bucket Through a Dynamic PV**.

Step 4 Check whether the mount options take effect.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can log in to the node where the pod to which the OBS volume is mounted resides and view the progress details.

Run the following command:

- Object bucket: **ps -ef | grep s3fs**

```
root     22142     1  0 Jun03 ?        00:00:00 /usr/bin/s3fs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/{your_obs_name} -o nonempty -o big_writes -o sigv2 -o allow_other -o no_check_certificate -o ssl_verify_hostname=0 -o umask=027 -o max_write=131072 -o multipart_size=20
```


- Parallel file system: **ps -ef | grep obsfs**

```
root      1355     1  0 Jun03 ?        00:03:16 /usr/bin/obsfs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o endpoint={region} -o passwd_file=/opt/everest-host-connector/***_obstmpcred/{your_obs_name} -o allow_other -o nonempty -o big_writes -o use_ino -o no_check_certificate -o ssl_verify_hostname=0 -o max_background=100 -o umask=027 -o max_write=131072
```

----End

8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume

Scenario

CCE Container Storage (Everest) of v1.2.8 or later supports custom access keys. In this way, IAM users can use their own custom access keys to mount an OBS volume. For details, see **Differences Between OBS Permissions Control Methods**.

Prerequisites

- The **CCE Container Storage (Everest)** version must be 2.1.46 or later.
- The cluster version must be 1.15.11 or later.

Notes and Constraints

- When an OBS volume is mounted using a custom access key (AK/SK), the access key cannot be deleted or disabled. Otherwise, the service container cannot access the mounted OBS volume.
- Custom access keys cannot be configured for Kata containers.

Disabling Auto Key Mounting

On the earlier version's console, you need to upload the AK/SK, which is then used by default for mounting an OBS volume. As a result, all IAM users within your account will use the same key to mount OBS buckets, and they will have identical permissions on the buckets. However, this setting does not allow you to set different permissions for individual IAM users.

If you have uploaded the AK/SK, disable the automatic mounting of access keys by enabling the **DISABLE_AUTO_MOUNT_SECRET** parameter in the Everest add-on to prevent IAM users from performing unauthorized operations. In this way, the access keys uploaded on the console will not be used when you use OBS volumes.

NOTE

- Before enabling **DISABLE_AUTO_MOUNT_SECRET**, ensure that there are no OBS volumes in the cluster. Workloads using OBS volumes may fail to remount after scaling or restart due to missing access keys, which are blocked by **DISABLE_AUTO_MOUNT_SECRET**.
- If **DISABLE_AUTO_MOUNT_SECRET** is set to **true**, an access key must be specified when a PV or PVC is created. Otherwise, mounting the OBS volume will fail.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Container Storage (Everest)** on the right, and click **Edit**.

Step 2 Configure the add-on parameters. Set **Prohibit Global Secret from Mounting Object Storage (disable_auto_mount_secret)** to **Yes**.

Step 3 Click **OK**.

----End

Obtaining an Access Key

Step 1 Log in to the console.

Step 2 Hover the cursor over the username in the upper right corner and choose **My Credentials** from the drop-down list.

Step 3 In the navigation pane, choose **Access Keys**.

Step 4 Click **Create Access Key**. The **Create Access Key** dialog box is displayed.

Step 5 Click **OK** to download the access key.

----End

Creating a Secret Using an Access Key

Step 1 Obtain an access key.

Step 2 Encode the keys using Base64. (Assume that the AK is xxx and the SK is yyy.)

echo -n xxx|base64

echo -n yyy|base64

Record the encoded AK and SK.

Step 3 Create a YAML file for the secret, for example, **test-user.yaml**.

```
apiVersion: v1
data:
  access.key: WE5WWVhVNU*****
  secret.key: Nnk4emJyZ0*****
kind: Secret
metadata:
  name: test-user
  namespace: default
  labels:
    secret.kubernetes.io/used-by: csi
type: cfe/secure-opaque
```

Specifically:

Parameter	Description
access.key	Base64-encoded AK.
secret.key	Base64-encoded SK.
name	Secret name.
namespace	Namespace of the secret.

Parameter	Description
secret.kubernetes.io/used-by: csi	Add this label in the YAML file if you want to make it available on the CCE console when you create an OBS PV/PVC.
type	Secret type. The value must be **cfe/secure-opaque**. When this type is used, the data entered by users is automatically encrypted.

Step 4 Create the secret.

kubectl create -f test-user.yaml

----End

Mounting a Secret When Statically Creating an OBS Volume

After a secret is created using the AK/SK, you can associate the secret with the PV to be created and then use the AK/SK in the secret to mount an OBS volume.

Step 1 Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class. The parallel file system is used as an example.

Step 2 Create a YAML file for the PV, for example, **pv-example.yaml**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    nodePublishSecretRef:
      name: test-user
      namespace: default
    driver: obs.csi.everest.io
    fsType: obsfs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: ap-southeast-1
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

Parameter	Description
nodePublishSecretRef	Secret specified during the mounting. • **name**: name of the secret • **namespace**: namespace of the secret

Parameter	Description
fsType	File type, which can be **s3fs** or **obsfs**. If the value is **s3fs**, an OBS bucket is created. If the value is **obsfs**, an OBS parallel file system is created.
volumeHandle	OBS bucket name.

Step 3 Create a PV.

kubectl create -f pv-example.yaml

After a PV is created, you can create a PVC and associate it with the PV.

Step 4 Create a YAML file for the PVC, for example, **pvc-example.yaml**.

Example YAML file for the PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

Step 5 Create a PVC.

kubectl create -f pvc-example.yaml

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

Mounting a Secret When Dynamically Creating an OBS Volume

When dynamically creating an OBS volume, you can use the following method to specify a secret:

Step 1 Create a YAML file for the PVC, for example, **pvc-example.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

Step 2 Create a PVC.

kubectl create -f pvc-example.yaml

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

Verification

You can use a secret of an IAM user to mount an OBS volume. Assume that a workload named **obs-secret** is created, the mount path in the container is **/temp**, and the IAM user has the CCE **ReadOnlyAccess** and **Tenant Guest** permissions.

1. Query the name of the workload pod.

   kubectl get po | grep obs-secret

   Expected outputs:

   ```
   obs-secret-5cd558f76f-vxslv        1/1     Running   0          3m22s
   ```

2. Query the objects in the mount path. In this example, the query is successful.

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

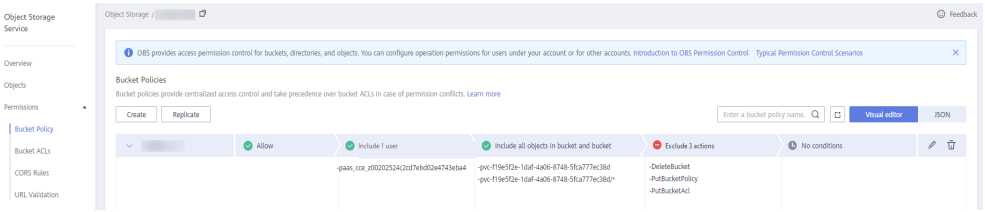
3. Write data into the mount path. In this example, the write operation failed.

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

   Expected outputs:

   ```
   touch: setting times of '/temp/test': No such file or directory
   command terminated with exit code 1
   ```

4. Set the read/write permissions for the IAM user who mounted the OBS volume by referring to the bucket policy configuration.



5. Write data into the mount path again. In this example, the write operation succeeded.

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

6. Check the mount path in the container to see whether the data is successfully written.

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

   Expected outputs:

   ```
   -rwxrwxrwx 1 root root 0 Jun  7 01:52 test
   ```

8.6.6 Using OBS Buckets Across Regions

By default, a pod can use OBS buckets only in the same region. CCE allows a workload to use OBS buckets across regions, which can improve resource utilization in some scenarios, but may also result in a higher latency.

Notes and Constraints

- The **CCE Container Storage (Everest)** add-on version must be **1.2.42 or later**.
- The node to which the storage is mounted must be able to access OBS buckets. Generally, the Internet or Direct Connect is used to access OBS buckets across regions. You can ping the endpoint of OBS on the node where OBS is located to check whether OBS is accessible.
- Only PVs can use OBS buckets across regions, and then are bound to PVCs. The PV reclaim policy must be **Retain**. StorageClass cannot be used to dynamically create PVCs for using OBS buckets across regions.

Procedure

Step 1 Create the paas-obs-endpoint ConfigMap and configure the region and endpoint of OBS.

The ConfigMap name is fixed to **paas-obs-endpoint**, and the namespace is fixed to **kube-system**.

Region names and endpoints are in key-value pairs. Replace *<region_name>* and *<endpoint_address>* with specific values. Use commas (,) to separate multiple values.

For details about its value, see **Regions and Endpoints**.

Example: {"ap-southeast-1": "https://obs.ap-southeast-1.myhuaweicloud.com:443", "ap-southeast-3": "https://obs.ap-southeast-3.myhuaweicloud.com:443"}

```
apiVersion: v1
kind: ConfigMap
metadata:
```



```
  name: paas-obs-endpoint   # The value must be paas-obs-endpoint.
  namespace: kube-system    # The value must be kube-system.
data:
  obs-endpoint: |
    {"<region_name>": "<endpoint_address>"}
```

Step 2 Create a PV.

Set **everest.io/region** to the region where OBS is located.

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: testing-abc
  annotations:
    pv.kubernetes.io/bound-by-controller: 'yes'
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    volumeHandle: testing-abc            # OBS bucket name
    fsType: s3fs                         # obsfs indicates a parallel file system, and s3fs indicates an OBS bucket.
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: <region_name>       # Region where the OBS bucket resides. Replace it with a specific value.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    nodePublishSecretRef:                # AK/SK used for mounting an OBS bucket
      name: test-user
      namespace: default
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain   # The value must be Retain.
  storageClassName: csi-obs
  volumeMode: Filesystem
```

nodePublishSecretRef is the access key (AK/SK) used for mounting the object storage volume. Use the AK/SK to create a secret, which will be used when creating a PV. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.

Step 3 Create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test-abc
  namespace: default
  annotations:
    everest.io/obs-volume-type: STANDARD               # Bucket type, which can be STANDARD or WARM when an OBS bucket is used
    csi.storage.k8s.io/fstype: s3fs                    # File type. obsfs indicates a parallel file system, and s3fs indicates an OBS bucket.
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany            # The value must be ReadWriteMany for object storage.
  resources:
    requests:
      storage: 1Gi           # Storage capacity of a PVC. This parameter is valid only for verification (fixed to 1, cannot be empty or 0). The value setting does not take effect for OBS buckets.
  storageClassName: csi-obs   # StorageClass name. For object storage, the value is fixed at csi-obs.
  volumeName: testing-abc     # PV name
```

Step 4 Create a workload, select the PVC in the data storage option of the container settings, and add the created PVC. If the workload is successfully created, the OBS bucket can be used across regions.


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: obs-deployment-example                  # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-deployment-example
  template:
    metadata:
      labels:
        app: obs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                       # Mount path
          name: pvc-obs-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: pvc-test-abc               # PVC name
```

----End

8.7 DSS

8.7.1 Overview

Dedicated Distributed Storage Service (DSS) provides dedicated physical storage resources. With data redundancy and cache acceleration technologies, DSS delivers highly reliable, durable, low-latency, and stable storage resources. CCE allows you to mount DSS storage volumes to containers.

DSS Performance

The key indicators to measure the performance of DSS storage pools include IOPS, throughput, and I/O latency.

- IOPS: the number of input/output operations per second
- Throughput: the amount of data read from and written into a storage pool per second
- I/O latency: the minimum interval between two consecutive I/O operations

Table 8-46 DSS performance specifications

Parameter	High I/O	Ultra-high I/O
IOPS	1500 IOPS/TB	8000 IOPS/TB
I/O latency (single queue, 4 KiB data blocks)	1 ms to 3 ms	1 ms

Parameter	High I/O	Ultra-high I/O
Typical application scenarios	Common development and testing	• Transcoding services • I/O-intensive services – NoSQL – SQL Server – PostgreSQL • Latency-sensitive services – Redis – Memcache

For more details about DSS performance, see **Storage Pool Types and Performance**.

Application Scenarios

DSS supports the following mounting modes based on application scenarios:

- **Using DSS Through a Static PV**: static creation mode, where you use an existing volume to create a PV and then mount storage to the workload through a PVC. This mode applies to scenarios where disks are available.
- **Using DSS Through a Dynamic PV**: dynamic creation mode, in which you do not need to create disks beforehand. Instead, specify a StorageClass when creating a PVC. Then, a disk and PV will be created automatically. This mode applies to scenarios where no disk is available.
- **Dynamically Mounting a DSS Disk to a StatefulSet**: available only for StatefulSets. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. This mode applies to StatefulSets with multiple pods.

Billing

- A DSS storage pool must be purchased. Then, the DSS storage mounted to containers will use the resources in the purchased resource pool.
- For details about DSS pricing, see **Billing**.

8.7.2 Using DSS Through a Static PV

CCE allows you to create a PV using an existing DSS disk. After the PV is created, you can create a PVC and bind it to the PV. This mode applies to scenarios where underlying storage is available.

Prerequisites

- You have created a cluster of version v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and **CCE Container Storage (Everest)** of v2.4.5 or later has been installed in the cluster.

- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have created a DSS disk that meets the following requirements:
  - The DSS disk cannot be a system disk or shared disk.
  - The DSS disk must be of the **SCSI** type (the default disk type is **VBD** when you create a DSS disk).
  - The DSS disk must be available and not used by other resources.
  - If the DSS disk is encrypted, the key must be available.
  - The DSS disk must be in the default enterprise project or the enterprise project to which the cluster belongs.

Notes and Constraints

- DSS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If a DSS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses DSS.
- If an HPA policy is used to expand the capacity of a workload with a DSS disk attached, new pods cannot be started because the DSS disk cannot attach to them.

Using an Existing DSS Disk on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Statically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **DSS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	– If underlying storage is available, create a PV or use an existing PV to statically create a PVC. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using DSS Through a Dynamic PV**. In this example, select **Create new** to create both a PV and PVC on the console.

Parameter	Description
PV[a]	Select an existing PV in the cluster. For details about how to create a PV, see "Creating a storage volume" in **Related Operations**. You do not need to specify this parameter in this example.
DSS[b]	Click **Select DSS**. On the displayed page, select the DSS volume that meets your requirements and click **OK**.
PV Name[b]	Enter the PV name, which must be unique in the same cluster.
Access Mode[b]	DSS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Reclaim Policy[b]	You can select **Delete** or **Retain** to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see **PV Reclaim Policy**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-47**. For details about other parameters, see **Workloads**.

Table 8-47 Mounting a storage volume

Parameter	Description
PVC	Select an existing DSS volume. A DSS volume can be mounted to only one workload.

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the DSS disk.

NOTE

A non-shared DSS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if a DSS disk is attached.

If multiple workload pods are needed, create a StatefulSet and dynamically mount a PV to each pod. For details, see **Dynamically Mounting a DSS Disk to a StatefulSet**.

3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Using an Existing DSS Disk Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV. If a PV has been created in your cluster, skip this step.

1. Create the **pv-dss.yaml** file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only        # (Optional) The underlying volume is retained when the PV is deleted.
  name: pv-dss    # PV name
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
spec:
  accessModes:
    - ReadWriteOnce     # Access mode, which must be ReadWriteOnce for DSS
  capacity:
    storage: 10Gi       # Disk capacity, in the unit of GiB. The value ranges from 1 to 32768.
  csi:
    driver: disk.csi.everest.io     # Dependent storage driver for the mounting
    fsType: ext4    # Must be the same as that of the original file system of the disk.
    volumeAttributes:
      everest.io/disk-mode: SCSI          # Device type of the DSS disk. Only SCSI is supported.
      everest.io/disk-volume-type: SAS     # Disk type
      everest.io/csi.dedicated-storage-id: <dss_id>     # ID of the DSS storage pool
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
      everest.io/enterprise-project-id: <your_project_id> # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
  persistentVolumeReclaimPolicy: Delete    # Reclaim policy
  storageClassName: csi-disk-dss            # Storage class name of the DSS disk
```

Table 8-48 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Only **retain-volume-only** is supported. This parameter is valid only when the Everest version is 1.2.9 or later and the reclaim policy is **Delete**. If the reclaim policy is **Delete** and the current value is **retain-volume-only**, the associated PV is deleted while the underlying storage volume is retained, when a PVC is deleted.
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the DSS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
fsType	Yes	File system type, which defaults to **ext4**.
everest.io/disk-volume-type	Yes	Disk type. All letters are in uppercase. – **SAS**: high I/O – **SSD**: ultra-high I/O
everest.io/csi.dedicated-storage-id	Yes	ID of the DSS storage pool where the DSS disk resides. To obtain a DSS storage pool ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Storage Pools** and click the name of the target storage pool. On the resource pool details page, copy the pool ID.
everest.io/crypt-key-id	No	Mandatory when the disk is encrypted. Enter the encryption key ID selected during disk creation. To obtain an encryption key ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Disks**. Click the name of the target disk to go to its details page. On the **Summary** tab page, copy the value of **KMS Key ID** in the **Configuration Information** area.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of DSS. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Disks**. Click the name of the target disk to go to its details page. On the **Summary** tab page, click the enterprise project in **Management Information** to access the enterprise project console. Copy the corresponding ID to obtain the ID of the enterprise project to which the DSS disk belongs.

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	The **Delete** and **Retain** reclaim policies are supported. For details, see **PV Reclaim Policy**. If high data security is required, select **Retain** to prevent data from being deleted by mistake. **Delete**: – If **everest.io/reclaim-policy** is not specified, both the PV and DSS disk will be deleted when a PVC is deleted. – If **everest.io/reclaim-policy** is set to **retain-volume-only**, when a PVC is deleted, the PV will be deleted but the DSS disk will be retained. **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storageClassName	Yes	The StorageClass for DSS disks is **csi-disk-dss**.

2. Run the following command to create a PV:

```
kubectl apply -f pv-dss.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-dss.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-dss
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # Disk type
    everest.io/csi.dedicated-storage-id: <dss_id>    # ID of the DSS storage pool
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
    everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>      # AZ of the node where the application is to be deployed
spec:
  accessModes:
  - ReadWriteOnce              # The value must be ReadWriteOnce for DSS.
  resources:
    requests:
      storage: 10Gi            # Disk capacity, ranging from 1 to 32768. The value must be the same as the storage size of the existing PV.
  storageClassName: csi-disk-dss    # The StorageClass is DSS.
  volumeName: pv-dss           # PV name
```


Table 8-49 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the disk is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/csi.dedicated-storage-id	Yes	ID of the DSS storage pool where the DSS disk resides. To obtain a DSS storage pool ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Storage Pools** and click the name of the target storage pool. On the resource pool details page, copy the pool ID.
storage	Yes	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Yes	PV name, which must be the same as the PV name in **1**.
storageClassName	Yes	StorageClass name, which must be the same as the StorageClass of the PV in **1**. The StorageClass for DSS disks is **csi-disk-dss**.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-dss.yaml
```

Step 4 Create an application.

1. Create a file named **web-dss.yaml**. In this example, the disk is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-dss
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-dss
  serviceName: web-dss   # Headless Service name
  template:
    metadata:
      labels:
        app: web-dss
```



```
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk-dss    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk-dss    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-dss    # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-dss   # Headless Service name
  namespace: default
  labels:
    app: web-dss
spec:
  selector:
    app: web-dss
  clusterIP: None
  ports:
    - name: web-dss
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the DSS volume is mounted:

```
kubectl apply -f web-dss.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and DSS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-dss
```

Expected output:

```
web-dss-0               1/1     Running   0             38s
```

2. Run the following command to check whether the DSS volume has been mounted to the **/data** path:

```
kubectl exec web-dss-0 -- df | grep data
```

Expected output:

```
/dev/sdc          10255636     36888  10202364   0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-dss-0 -- ls /data
```

Expected output:

```
lost+found
```

Step 2 Run the following command to create a file named **static** in the **/data** path:


```
kubectl exec web-dss-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-dss-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-dss-0**:

```
kubectl delete pod web-dss-0
```

Expected output:

```
pod "web-dss-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-dss-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the DSS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-50**.

Table 8-50 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVs** tab. Click **Create PersistentVolume** in the upper right corner. In the dialog box displayed, configure parameters. • **Volume Type**: Select **DSS**. • **DSS**: Click **Select DSS**. On the displayed page, select the disk that meets your requirements and click **OK**. • **PV Name**: Enter the PV name, which must be unique in a cluster. • **Access Mode**: DSS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**. • **Reclaim Policy**: **Delete** or **Retain** is supported. For details, see **PV Reclaim Policy**. 2. Click **Create**.
Expanding the capacity of DSS storage	Quickly expand the capacity of an attached DSS disk on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.7.3 Using DSS Through a Dynamic PV

CCE allows you to specify a StorageClass to automatically create a DSS disk and the corresponding PV. This function is applicable when no underlying storage volume is available.

Prerequisites

- You have created a cluster of version v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and **CCE Container Storage (Everest)** of v2.4.5 or later has been installed in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Notes and Constraints

- DSS disks cannot be attached across AZs and cannot be used by multiple workloads, multiple pods of the same workload, or multiple tasks. Data sharing of a shared disk is not supported between nodes in a CCE cluster. If a DSS disk is attached to multiple nodes, I/O conflicts and data cache conflicts may occur. Therefore, select only one pod when creating a Deployment that uses DSS.
- If an HPA policy is used to expand the capacity of a workload with a DSS disk attached, new pods cannot be started because the DSS disk cannot attach to them.
- Resource tags can be added to dynamically created DSS disks. After the DSS disks are created, the resource tags cannot be updated on CCE. To update them, go to the DSS console. If you use an existing DSS disk to create a PV, you also need to add or update resource tags on the DSS console.

Automatically Creating a DSS Volume on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this example, select **DSS**.
PVC Name	Enter the PVC name, which must be unique in a namespace.

Parameter	Description
Creation Method	– If no underlying storage is available, select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode. – If underlying storage is available, create a PV or use an existing PV to statically create a PVC. For details, see **Using DSS Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The default StorageClass for DSS disks is **csi-disk-dss**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
DSS Pool	Select an existing DSS pool.
Capacity (GiB)	Capacity of the requested storage volume.
Access Mode	DSS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Encryption	Configure whether to encrypt underlying storage. If you select **Enabled (key)**, an encryption key must be configured.
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

Parameter	Description
Resource Tag	You can add resource tags to classify resources, which is supported only when the Everest version in the cluster is 2.1.39 or later. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}*, and **CCE-Namespace=***{Namespace name}*. These tags cannot be modified. **NOTE** After a dynamic PV of the DSS type is created, the resource tags cannot be updated on the CCE console. To update DSS resource tags, go to the DSS console.

2. Click **Create**.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

Step 3 Create an application.

1. Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-51**. For details about other parameters, see **Workloads**.

Table 8-51 Mounting a storage volume

Parameter	Description
PVC	Select an existing DSS volume. A DSS volume can be mounted to only one workload.

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the DSS disk.

NOTE

A non-shared DSS disk can be attached to only one workload pod. If there are multiple pods, extra pods cannot start properly. Ensure that the number of workload pods is 1 if a DSS disk is attached.

If multiple workload pods are needed, create a StatefulSet and dynamically mount a PV to each pod. For details, see **Dynamically Mounting a DSS Disk to a StatefulSet**.

3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Automatically Creating a DSS Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Use StorageClass to dynamically create a PVC and PV.

1. Create the **pvc-dss-auto.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-dss-auto
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS    # Disk type
    everest.io/csi.dedicated-storage-id: <dss_id>    # ID of the DSS storage pool
    everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
    everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
    everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # (Optional) Custom resource tags
    everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
  labels:
    failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
    failure-domain.beta.kubernetes.io/zone: <your_zone>      # AZ of the node where the application is to be deployed
spec:
  accessModes:
  - ReadWriteOnce              # The value must be ReadWriteOnce for DSS.
  resources:
    requests:
      storage: 10Gi           # Disk capacity, ranging from 1 to 32768
  storageClassName: csi-disk-dss    # The StorageClass is DSS.
```

Table 8-52 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the disk is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/disk-volume-type	Yes	Disk type. All letters are in uppercase. – **SAS**: high I/O – **SSD**: ultra-high I/O
everest.io/csi.dedicated-storage-id	Yes	ID of the DSS storage pool where the DSS disk resides. To obtain a DSS storage pool ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Storage Pools** and click the name of the target storage pool. On the resource pool details page, copy the pool ID.

Parameter	Mandatory	Description
everest.io/crypt-key-id	No	Mandatory when the DSS disk is encrypted. Enter the encryption key ID selected during disk creation. To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of DSS. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	StorageClass name, which is **csi-disk-dss** for a DSS disk.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-dss-auto.yaml
```

Step 3 Create an application.

1. Create a file named **web-dss-auto.yaml**. In this example, the disk is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: StatefulSet
```



```
metadata:
  name: web-dss-auto
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-dss-auto
  serviceName: web-dss-auto   # Headless Service name
  template:
    metadata:
      labels:
        app: web-dss-auto
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk-dss    # Volume name, which must be the same as the volume name in the volumes field
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk-dss    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-dss-auto    # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-dss-auto   # Headless Service name
  namespace: default
  labels:
    app: web-dss-auto
spec:
  selector:
    app: web-dss-auto
  clusterIP: None
  ports:
    - name: web-dss-auto
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the DSS volume is mounted:

```
kubectl apply -f web-dss-auto.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and DSS volume files.

1. Run the following command to view the created pod:

```
kubectl get pod | grep web-dss-auto
```

Expected output:

```
web-dss-auto-0              1/1     Running   0             38s
```

2. Run the following command to check whether the DSS volume has been mounted to the **/data** path:


```
kubectl exec web-dss-auto-0 -- df | grep data
```

Expected output:

```
/dev/sdc        10255636     36888  10202364   0% /data
```

3. Run the following command to check the files in the **/data** path:

```
kubectl exec web-dss-auto-0 -- ls /data
```

Expected output:

```
lost+found
```

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-dss-auto-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-dss-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-dss-auto-0**:

```
kubectl delete pod web-dss-auto-0
```

Expected output:

```
pod "web-dss-auto-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-dss-auto-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the DSS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-53**.

Table 8-53 Related operations

Operation	Description	Procedure
Expanding the capacity of DSS storage	Quickly expand the capacity of an attached DSS disk on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.7.4 Dynamically Mounting a DSS Disk to a StatefulSet

Application Scenarios

Dynamic mounting is available only for creating a **StatefulSet**. It is implemented through a volume claim template (**volumeClaimTemplates** field) and depends on dynamic creation of PVs through StorageClass. In this mode, each pod in a multi-pod StatefulSet is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. In the common mounting mode for a Deployment, if ReadWriteMany is supported, multiple pods of the Deployment will be mounted to the same underlying storage.

NOTE

When updating a StatefulSet in Kubernetes, it is not allowed to add or delete the **volumeClaimTemplates** field. This field can only be configured during the creation of the StatefulSet.

Prerequisites

- You have created a cluster of version v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and **CCE Container Storage (Everest)** of v2.4.5 or later has been installed in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.

Dynamically Mounting a DSS Disk on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.

Step 3 Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **VolumeClaimTemplate**.

Step 4 Click **Create PVC**. In the dialog box displayed, configure PVC parameters.

Click **Create**.

Parameter	Description
PVC Type	In this example, select **DSS**.
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <*Custom PVC name*>-<*Serial number*>, for example, example-0.
Creation Method	You can select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass for DSS disks is **csi-disk-dss**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-***{UID}*.
DSS Pool	Select an existing DSS pool.
Capacity (GiB)	Capacity of the requested storage volume.
Access Mode	DSS volumes support only **ReadWriteOnce**, indicating that a storage volume can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Encryption	Configure whether to encrypt underlying storage. If you select **Enabled (key)**, an encryption key must be configured.
Enterprise Project	The default enterprise project, the enterprise project to which the cluster belongs, or the enterprise project specified by StorageClass is available.

Parameter	Description
Resource Tag	You can add resource tags to classify resources, which is supported only when the Everest version in the cluster is 2.1.39 or later. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=*{Cluster ID}***, **CCE-Cluster-Name=*{Cluster name}***, and **CCE-Namespace=*{Namespace name}***. These tags cannot be modified. **NOTE** After a dynamic PV of the DSS type is created, the resource tags cannot be updated on the CCE console. To update DSS resource tags, go to the DSS console.

Step 5 Enter the path to which the volume is mounted.

Table 8-54 Mounting a storage volume

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.

Parameter	Description
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the DSS disk.

Step 6 Dynamically mount and use storage volumes. For details about other parameters, see **Creating a StatefulSet**. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Dynamically Mounting a DSS Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a file named **statefulset-dss.yaml**. In this example, the disk is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-dss
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-dss
  template:
    metadata:
      labels:
        app: statefulset-dss
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-disk          # The value must be the same as that in the volumeClaimTemplates field.
              mountPath: /data        # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-dss        # Headless Service name
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-disk
        namespace: default
        annotations:
          everest.io/disk-volume-type: SAS    # Disk type
          everest.io/csi.dedicated-storage-id: <dss_id>    # ID of the DSS storage pool
          everest.io/crypt-key-id: <your_key_id>    # (Optional) Encryption key ID. Mandatory for an encrypted disk.
```



```
        everest.io/enterprise-project-id: <your_project_id>  # (Optional) Enterprise project ID. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV.
        everest.io/disk-volume-tags: '{"key1":"value1","key2":"value2"}' # (Optional) Custom resource tags
        everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
      labels:
        failure-domain.beta.kubernetes.io/region: <your_region>   # Region of the node where the application is to be deployed
        failure-domain.beta.kubernetes.io/zone: <your_zone>       # AZ of the node where the application is to be deployed
    spec:
      accessModes:
        - ReadWriteOnce              # The value must be ReadWriteOnce for DSS.
      resources:
        requests:
          storage: 10Gi              # Disk capacity, ranging from 1 to 32768
      storageClassName: csi-disk-dss      # The StorageClass is DSS.
---
apiVersion: v1
kind: Service
metadata:
  name: statefulset-dss   # Headless Service name
  namespace: default
  labels:
    app: statefulset-dss
spec:
  selector:
    app: statefulset-dss
  clusterIP: None
  ports:
    - name: statefulset-dss
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Table 8-55 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about its value, see **Regions and Endpoints**.
failure-domain.beta.kubernetes.io/zone	Yes	AZ where the DSS volume is created. It must be the same as the AZ planned for the workload. For details about its value, see **Regions and Endpoints**.
everest.io/disk-volume-type	Yes	Disk type. All letters are in uppercase. • **SAS**: high I/O • **SSD**: ultra-high I/O

Parameter	Mandatory	Description
everest.io/csi.dedicated-storage-id	Yes	ID of the DSS storage pool where the DSS disk resides. To obtain a DSS storage pool ID, log in to the **Cloud Server Console**. In the navigation pane, choose **Dedicated Distributed Storage Service** > **Storage Pools** and click the name of the target storage pool. On the resource pool details page, copy the pool ID.
everest.io/crypt-key-id	No	Mandatory when the DSS disk is encrypted. Enter the encryption key ID selected during disk creation. To obtain a key ID, log in to the DEW console, locate the key to be encrypted, and copy the key ID.
everest.io/enterprise-project-id	No	Optional. Enterprise project ID of DSS. If an enterprise project is specified, use the same enterprise project when creating a PVC. Otherwise, the PVC cannot be bound to a PV. To obtain an enterprise project ID, log in to the EPS console, click the name of the target enterprise project, and copy the enterprise project ID.
everest.io/disk-volume-tags	No	This field is optional. It is supported when the Everest version in the cluster is 2.1.39 or later. You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. These tags are available to all resources that support tags. You can use these tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. CCE automatically creates system tags **CCE-Cluster-ID=***{Cluster ID}*, **CCE-Cluster-Name=***{Cluster name}*, and **CCE-Namespace=***{Namespace name}*. These tags cannot be modified.

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	StorageClass name, which is **csi-disk-dss** for a DSS disk.

Step 3 Run the following command to create a workload to which the DSS volume is mounted:

```
kubectl apply -f statefulset-dss.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and DSS volume files.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep statefulset-dss
   ```

   Expected output:

   ```
   statefulset-dss-0       1/1     Running   0          45s
   statefulset-dss-1       1/1     Running   0          28s
   ```

2. Run the following command to check whether the DSS volume has been mounted to the **/data** path:

   ```
   kubectl exec statefulset-dss-0 -- df | grep data
   ```

   Expected output:

   ```
   /dev/sdd        10255636     36888  10202364   0% /data
   ```

3. Run the following command to check the files in the **/data** path:

   ```
   kubectl exec statefulset-dss-0 -- ls /data
   ```

   Expected output:

   ```
   lost+found
   ```


Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec statefulset-dss-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-dss-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-dss-auto-0**:

```
kubectl delete pod statefulset-dss-0
```

Expected output:

```
pod "statefulset-dss-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec statefulset-dss-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the DSS volume can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-56**.

Table 8-56 Related operations

Operation	Description	Procedure
Expanding the capacity of DSS storage	Quickly expand the capacity of an attached DSS disk on the CCE console.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **More** in the **Operation** column of the target PVC and select **Scale-out**. 2. Enter the capacity to be added and click **OK**.
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).

Operation	Description	Procedure
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.8 Local PVs

8.8.1 Overview

Introduction

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. A PV that uses a local persistent volume as the medium is considered local PV.

Compared with the HostPath volume, the local PV can be used in a persistent and portable manner. In addition, the PV of the local PV has the node affinity configuration. The pod mounted to the local PV is automatically scheduled based on the affinity configuration. You do not need to manually schedule the pod to a specific node.

Mount Modes

Local PVs can be mounted only in the following modes:

- **Using a Local PV Through a Dynamic PV**: dynamic creation mode, in which you specify a StorageClass when creating a PVC. Then, an OBS volume and PV will be created automatically.
- **Dynamically Mounting a Local PV to a StatefulSet**: available only for StatefulSets. In this mode, each pod is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. This mode applies to StatefulSets with multiple pods.

NOTE

Local PVs cannot be used through static PVs. That is, local PVs cannot be manually created and then mounted to workloads through PVCs.

Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.
- **Removing**, **deleting**, **resetting**, or **scaling** a node will cause the loss of the PVC/PV data of the local PV associated with the node. The lost data cannot be restored, and the affected PVC/PV cannot be used again. In these

scenarios, the pod that uses the local PV is evicted from the node. A new pod will be created and stay in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled. After the node is reset, the pod may be scheduled to the reset node. In this case, the pod remains in the creating state because the underlying logical volume corresponding to the PVC does not exist.

- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Local PVs are in non-shared mode and cannot be mounted to multiple workloads or tasks concurrently. Additionally, local PVs cannot be mounted to multiple pods of a workload concurrently.

8.8.2 Importing a PV to a Storage Pool

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. Before creating a local PV, import the data disk of the node to the storage pool.

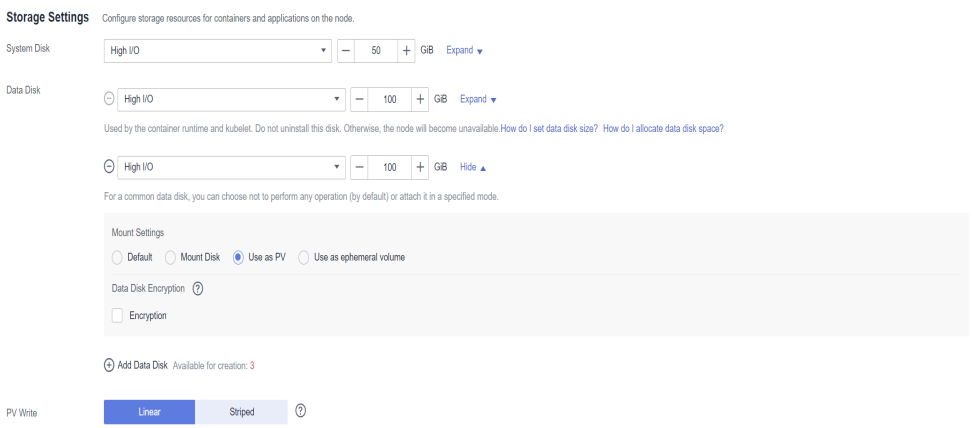
Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.
- The first data disk (used by container runtime and the kubelet component) on a node cannot be imported as a storage pool.
- Storage pools in striped mode do not support scale-out. After scale-out, fragmented space may be generated and the storage pool cannot be used.
- Storage pools cannot be scaled in or deleted.
- If disks in a storage pool on a node are deleted, the storage pool will malfunction.

Importing a Storage Pool

Imported during node creation

When creating a node, you can add a data disk to the node in **Storage Settings** and import the data disk to the storage pool as a PV. For details, see **Creating a Node**.



Imported manually

If no PV is imported during node creation, or the capacity of the current PV is insufficient, you can manually import a PV.

Step 1 Go to the ECS console and add a SCSI disk to the node. For details, see **Adding a Disk**.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console.

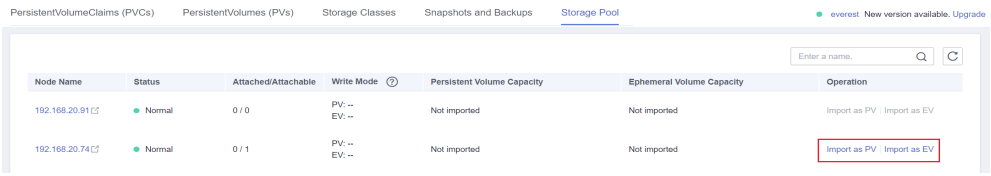
Step 3 Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.

Step 4 View the node to which the disk has been added and select **Import as PV**. You can select a write mode during the import.

NOTE

If the manually attached disk is not displayed in the storage pool, wait for 1 minute and refresh the list.

- **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
- **Striped**: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. Select this option only when there are multiple volumes.



----End

Expanding a Storage Pool

The capacity of a storage pool can be expanded in either of the following ways:

1. Add more large-capacity disks to the pool using the preceding manual import method.
2. Increase the size of the existing disks on the ECS console. Then, the storage pool capacity will be automatically expanded.

Either of these methods will work.

8.8.3 Using a Local PV Through a Dynamic PV

Prerequisites

- You have created a cluster and installed the CSI add-on (**Everest**) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have imported a data disk of a node to the local PV storage pool. For details, see **Importing a PV to a Storage Pool**.

Notes and Constraints

- Local PVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 2.1.23 or later. Version 2.1.23 or later is recommended.
- **Removing**, **deleting**, **resetting**, or **scaling** a node will cause the loss of the PVC/PV data of the local PV associated with the node. The lost data cannot be restored, and the affected PVC/PV cannot be used again. In these scenarios, the pod that uses the local PV is evicted from the node. A new pod will be created and stay in the pending state. This is because the PVC used by the pod has a node label, due to which the pod cannot be scheduled. After the node is reset, the pod may be scheduled to the reset node. In this case, the pod remains in the creating state because the underlying logical volume corresponding to the PVC does not exist.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Local PVs are in non-shared mode and cannot be mounted to multiple workloads or tasks concurrently. Additionally, local PVs cannot be mounted to multiple pods of a workload concurrently.

Automatically Creating a Local PV on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Dynamically create a PVC and PV.

1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure PVC parameters.

Parameter	Description
PVC Type	In this section, select **Local PV**.
PVC Name	Enter the PVC name, which must be unique in a namespace.
Creation Method	You can only select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass of local PVs is **csi-local-topology**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.

Parameter	Description
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
Capacity	Requested capacity of the storage volume in GiB or MiB. **NOTE** To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
Access Mode	Local PVs support only **ReadWriteOnce**, indicating that a PV can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Storage Pool	View the imported storage pool. For details about how to import a new data volume to the storage pool, see **Importing a PV to a Storage Pool**.

2. Click **Create** to create a PVC and a PV.

   You can choose **Storage** in the navigation pane and view the created PVC and PV on the **PVCs** and **PVs** tab pages, respectively.

   NOTE

   The volume binding mode of the local storage class (named **csi-local-topology**) is late binding (that is, the value of **volumeBindingMode** is **WaitForFirstConsumer**). In this mode, PV creation and binding are delayed. The corresponding PV is created and bound only when the PVC is used during workload creation.

Step 3 Create an application.

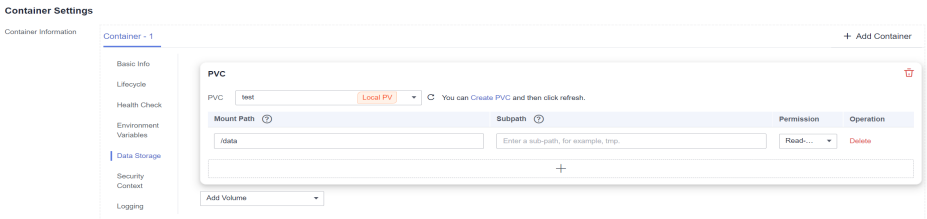
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 8-57**. For details about other parameters, see **Workloads**.

Table 8-57 Mounting a storage volume

Parameter	Description
PVC	Select an existing local PV. A local PV can be mounted to only one workload.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	– **Read-only**: You can only read the data in the mounted volumes. – **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the local PV.



3. After the configuration, click **Create Workload**.

   After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Automatically Creating a Local PV Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Use **StorageClass** to dynamically create a PVC and PV.

1. Create the **pvc-local.yaml** file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local
  namespace: default
  annotations:
    everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
spec:
  accessModes:
    - ReadWriteOnce             # The value must be ReadWriteOnce for local PVs.
  resources:
    requests:
      storage: 10Gi             # Local PV capacity
  storageClassName: csi-local-topology    # StorageClass is local PV.
```

Table 8-58 Key parameters

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-***{UID}*.
storage	Yes	Requested PVC capacity in Gi or Mi. **NOTE** To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.

Parameter	Mandatory	Description
storageClassName	Yes	StorageClass name, which is **csi-local-topology** for a local PV.

2. Run the following command to create a PVC:
```
kubectl apply -f pvc-local.yaml
```

Step 3 Create an application.

1. Create a file named **web-local.yaml**. In this example, the local PV is mounted to the **/data** path.
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-local
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-local
  serviceName: web-local   # Headless Service name
  template:
    metadata:
      labels:
        app: web-local
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: pvc-disk    # Volume name, which must be the same as the volume name in the volumes field.
          mountPath: /data  # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: pvc-disk    # Volume name, which can be customized
          persistentVolumeClaim:
            claimName: pvc-local    # Name of the created PVC
---
apiVersion: v1
kind: Service
metadata:
  name: web-local   # Headless Service name
  namespace: default
  labels:
    app: web-local
spec:
  selector:
    app: web-local
  clusterIP: None
  ports:
    - name: web-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

2. Run the following command to create a workload to which the local PV is mounted:
```
kubectl apply -f web-local.yaml
```


After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and local files.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-local
   ```
   Expected output:
   ```
   web-local-0              1/1     Running   0             38s
   ```
2. Run the following command to check whether the local PV has been mounted to the **/data** path:
   ```
   kubectl exec web-local-0 -- df | grep data
   ```
   Expected output:
   ```
   /dev/mapper/vg--everest--localvolume--persistent-pvc-local        10255636     36888  10202364   0% /data
   ```
3. Run the following command to check the files in the **/data** path:
   ```
   kubectl exec web-local-0 -- ls /data
   ```
   Expected output:
   ```
   lost+found
   ```

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-local-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-local-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-local-0**:

```
kubectl delete pod web-local-0
```

Expected output:

```
pod "web-local-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec web-local-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the local PV can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-59**.

Table 8-59 Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.8.4 Dynamically Mounting a Local PV to a StatefulSet

Application Scenarios

Dynamic mounting is available only for creating a **StatefulSet**. It is implemented through a volume claim template (**volumeClaimTemplates** field) and depends on dynamic creation of PVs through StorageClass. In this mode, each pod in a multi-pod StatefulSet is associated with a unique PVC and PV. After a pod is rescheduled, the original data can still be mounted to it based on the PVC name. In the common mounting mode for a Deployment, if ReadWriteMany is supported, multiple pods of the Deployment will be mounted to the same underlying storage.

NOTE

When updating a StatefulSet in Kubernetes, it is not allowed to add or delete the **volumeClaimTemplates** field. This field can only be configured during the creation of the StatefulSet.

Prerequisites

- You have created a cluster and installed the CSI add-on (**Everest**) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- You have imported a data disk of a node to the local PV storage pool. For details, see **Importing a PV to a Storage Pool**.

Dynamically Mounting a Local PV on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **StatefulSets** tab.

Step 3 Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **VolumeClaimTemplate**.

Step 4 Click **Create PVC**. In the dialog box displayed, configure the volume claim template parameters.

Click **Create**.

Parameter	Description
PVC Type	In this section, select **Local PV**.
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <*Custom PVC name*>-<*Serial number*>, for example, example-0.
Creation Method	You can only select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The default StorageClass of local PVs is **csi-local-topology**. You can customize a StorageClass and configure its reclaim policy and binding mode. For details, see **Creating a StorageClass Using the CCE Console**.
(Optional) Storage Volume Name Prefix	Available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-***{UID}*.
Capacity	Requested capacity of the storage volume in GiB or MiB. **NOTE** To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.

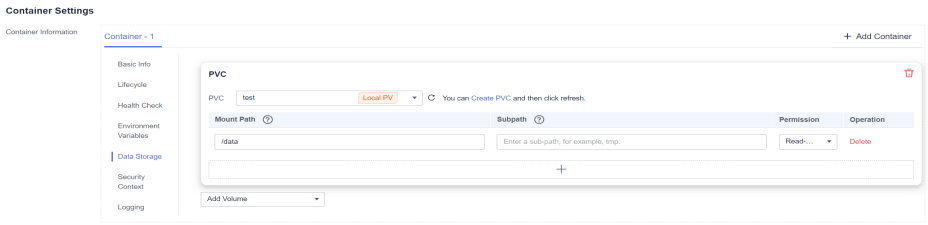
Parameter	Description
Access Mode	Local PVs support only **ReadWriteOnce**, indicating that a PV can be mounted to one node in read/write mode. For details, see **Volume Access Modes**.
Storage Pool	View the imported storage pool. For details about how to import a new data volume to the storage pool, see **Importing a PV to a Storage Pool**.

Step 5 Enter the path to which the volume is mounted.

Table 8-60 Mounting a storage volume

Parameter	Description
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

In this example, the disk is mounted to the **/data** path of the container. The container data generated in this path is stored in the local PV.



Step 6 Dynamically mount and use storage volumes. For details about other parameters, see **Creating a StatefulSet**. After the configuration, click **Create Workload**.

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence**.

----End

Dynamically Mounting a Local PV Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a file named **statefulset-local.yaml**. In this example, the local PV is mounted to the **/data** path.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: statefulset-local
  namespace: default
spec:
  selector:
    matchLabels:
      app: statefulset-local
  template:
    metadata:
      labels:
        app: statefulset-local
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: pvc-local          # The value must be the same as that in the volumeClaimTemplates field.
              mountPath: /data         # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
  serviceName: statefulset-local       # Headless Service name
  replicas: 2
  volumeClaimTemplates:
    - apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
        name: pvc-local
        namespace: default
        annotations:
          everest.io/csi.volume-name-prefix: test  # (Optional) Storage volume name prefix of the automatically created underlying storage
      spec:
        accessModes:
          - ReadWriteOnce              # The value must be ReadWriteOnce for local PVs.
        resources:
          requests:
            storage: 10Gi              # Storage volume capacity
        storageClassName: csi-local-topology      # StorageClass is local PV.
---
apiVersion: v1
kind: Service
```



```
metadata:
  name: statefulset-local   # Headless Service name
  namespace: default
  labels:
    app: statefulset-local
spec:
  selector:
    app: statefulset-local
  clusterIP: None
  ports:
    - name: statefulset-local
      targetPort: 80
      nodePort: 0
      port: 80
      protocol: TCP
  type: ClusterIP
```

Table 8-61 Key parameters

Parameter	Mandatory	Description
everest.io/csi.volume-name-prefix	No	(Optional) This parameter is available only when the cluster version is v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later, and Everest of v2.4.15 or later is installed in the cluster. This parameter specifies the name of the underlying storage that is automatically created. The actual underlying storage name is in the format of "Storage volume name prefix + PVC UID". If this parameter is left blank, the default prefix **pvc** will be used. Enter 1 to 26 characters that cannot start or end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed. For example, if the storage volume name prefix is set to **test**, the actual underlying storage name is **test-*{UID}***.
storage	Yes	Requested PVC capacity in Gi or Mi. **NOTE** To implement local PVs, Logical Volume Manager (LVM) is used with a local extent (LE) of 4 MiB. If the requested PVC capacity is not a multiple of 4 MiB, the LVM logical volume's actual capacity will be rounded up to the nearest multiple of 4 MiB. For example, if you request 401 MiB of PVC capacity, the LVM logical volume's actual capacity will be 404 MiB (101 LEs), which is the same as what is displayed on the page.
storageClassName	Yes	The StorageClass of local PVs is **csi-local-topology**.

Step 3 Run the following command to create a workload to which the local PV is mounted:

```
kubectl apply -f statefulset-local.yaml
```


After the workload is created, you can try **Verifying Data Persistence**.

----End

Verifying Data Persistence

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep statefulset-local
   ```
   Expected output:
   ```
   statefulset-local-0         1/1     Running   0          45s
   statefulset-local-1         1/1     Running   0          28s
   ```
2. Run the following command to check whether the local PV has been mounted to the **/data** path:
   ```
   kubectl exec statefulset-local-0 -- df | grep data
   ```
   Expected output:
   ```
   /dev/mapper/vg--everest--localvolume--persistent-pvc-local      10255636     36888  10202364   0% /data
   ```
3. Run the following command to check the files in the **/data** path:
   ```
   kubectl exec statefulset-local-0 -- ls /data
   ```
   Expected output:
   ```
   lost+found
   ```

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec statefulset-local-0 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec statefulset-local-0 -- ls /data
```

Expected output:

```
lost+found
static
```

Step 4 Run the following command to delete the pod named **web-local-auto-0**:

```
kubectl delete pod statefulset-local-0
```

Expected output:

```
pod "statefulset-local-0" deleted
```

Step 5 After the deletion, the StatefulSet controller automatically creates a replica with the same name. Run the following command to check whether the files in the **/data** path have been modified:

```
kubectl exec statefulset-local-0 -- ls /data
```

Expected output:

```
lost+found
static
```

The **static** file is retained, indicating that the data in the local PV can be stored persistently.

----End

Related Operations

You can also perform the operations listed in **Table 8-62**.

Table 8-62 Related operations

Operation	Description	Procedure
Viewing events	View event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View Events** in the **Operation** column of the target PVC or PV to view events generated within one hour (events are retained for one hour).
Viewing a YAML file	View, copy, or download the YAML file of a PVC or PV.	1. Choose **Storage** in the navigation pane. In the right pane, click the **PVCs** or **PVs** tab. 2. Click **View YAML** in the **Operation** column of the target PVC or PV to view or download the YAML.

8.9 Ephemeral Volumes

8.9.1 Overview

Introduction

Some applications require additional storage, but whether the data is still available after a restart is not important. For example, although cache services are limited by memory size, cache services can move infrequently used data to storage slower than memory. As a result, overall performance is not impacted significantly. Other applications require read-only data injected as files, such as configuration data or secrets.

Ephemeral volumes (EVs) in Kubernetes are designed for the above scenario. EVs are created and deleted together with pods following the pod lifecycle.

Common EVs in Kubernetes:

- **emptyDir**: empty at pod startup, with storage coming locally from the kubelet base directory (usually the root disk) or memory. emptyDir is allocated from the **EV of the node**. If data from other sources (such as log files or image tiering data) occupies the ephemeral storage, the storage capacity may be insufficient.
- **ConfigMap**: Kubernetes data of the ConfigMap type is mounted to pods as data volumes.
- **Secret**: Kubernetes data of the Secret type is mounted to pods as data volumes.

emptyDir Types

CCE provides the following emptyDir types:

- **Using a Temporary Path**: Kubernetes-native emptyDir type. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.
- **Using a Local EV**: Local data disks in a node form a **storage pool** (VolumeGroup) through LVM. LVs are created as the storage medium of emptyDir and mounted to pods. LVs deliver better performance than the default storage medium of emptyDir.

Notes and Constraints

- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Ensure that the **/var/lib/kubelet/pods/** directory is not mounted to the pod on the node. Otherwise, the pod, mounted with such volumes, may fail to be deleted.

8.9.2 Importing an EV to a Storage Pool

CCE allows you to use LVM to combine data volumes on nodes into a storage pool (VolumeGroup) and create LVs for containers to mount. Before creating a local EV, import the data disk of the node to the storage pool.

Notes and Constraints

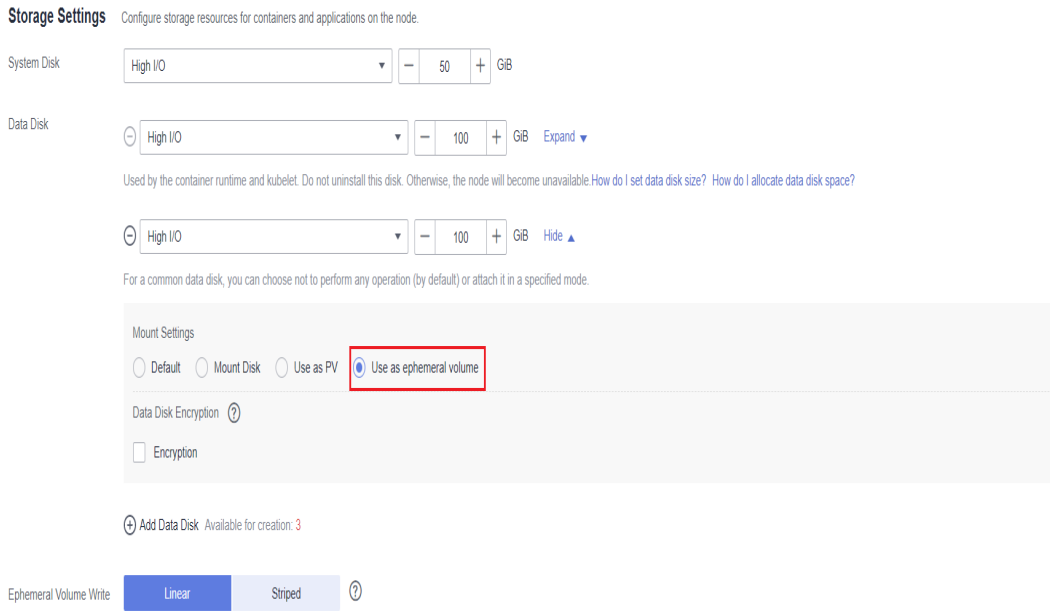
- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- The first data disk (used by container runtime and the kubelet component) on a node cannot be imported as a storage pool.
- Storage pools in striped mode do not support scale-out. After scale-out, fragmented space may be generated and the storage pool cannot be used.
- Storage pools cannot be scaled in or deleted.
- If disks in a storage pool on a node are deleted, the storage pool will malfunction.

Importing a Storage Pool

Imported during node creation

When creating a node, you can add a data disk to the node in **Storage Settings** and import the data disk to the storage pool as an EV. For details, see **Creating a Node**.

Figure 8-8 Importing as an EV



Imported manually

If no EV is imported during node creation, or the capacity of the current storage volume is insufficient, you can manually import an EV.

Step 1 Go to the ECS console and add a SCSI disk to the node. For details, see **Adding a Disk**.

Step 2 Log in to the CCE console and click the cluster name to access the cluster console.

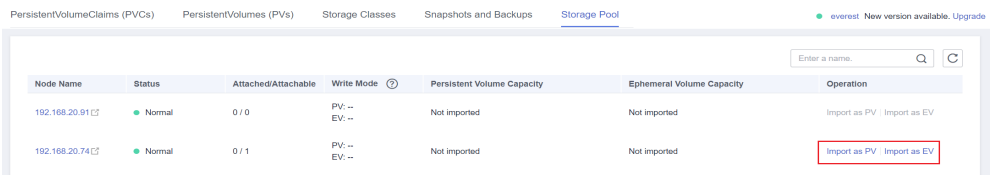
Step 3 Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.

Step 4 View the node to which the disk has been added and select **Import as EV**. You can select a write mode during the import.

NOTE

If the manually attached disk is not displayed in the storage pool, wait for 1 minute and refresh the list.

- **Linear**: A linear logical volume integrates one or more physical volumes. Data is written to the next physical volume when the previous one is used up.
- **Striped**: A striped logical volume stripes data into blocks of the same size and stores them in multiple physical volumes in sequence. This allows data to be concurrently read and written. Select this option only when there are multiple volumes.



----End

Expanding a Storage Pool

The capacity of a storage pool can be expanded in either of the following ways:

1. Add more large-capacity disks to the pool using the preceding manual import method.
2. Increase the size of the existing disks on the ECS console. Then, the storage pool capacity will be automatically expanded.

Either of these methods will work.

8.9.3 Using a Local EV

Local Ephemeral Volumes (EVs) are stored in EV **storage pools**. Local EVs deliver better performance than the default storage medium of native emptyDir and support scale-out.

Prerequisites

- You have created a cluster and installed the CSI add-on (**Everest**) in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **Connecting to a Cluster Using kubectl**.
- To use a local EV, import a data disk of a node to the local EV storage pool. For details, see **Importing an EV to a Storage Pool**.

Notes and Constraints

- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- Do not manually delete the corresponding storage pool or detach data disks from the node. Otherwise, exceptions such as data loss may occur.
- Ensure that the **/var/lib/kubelet/pods/** directory is not mounted to the pod on the node. Otherwise, the pod, mounted with such volumes, may fail to be deleted.

Using the Console to Mount a Local EV

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.

Step 3 Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **Local Ephemeral Volume (emptyDir)**.

Step 4 Mount and use storage volumes, as shown in **Table 8-63**. For details about other parameters, see **Workloads**.

Table 8-63 Mounting a local EV

Parameter	Description
Capacity	Capacity of the requested storage volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path will be used by default.
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 5 After the configuration, click **Create Workload**.

----End

Mounting a Local EV Through kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-emptydir.yaml** and edit it.

vi nginx-emptydir.yaml

Content of the YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
```



```
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir        # Volume name, which must be the same as the volume name in the volumes field.
              mountPath: /tmp           # Location where the emptyDir is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-emptydir            # Volume name, which can be customized
          emptyDir:
            medium: LocalVolume         # If the disk medium of emptyDir is set to LocalVolume, the local EV is used.
            sizeLimit: 1Gi              # Volume capacity
```

Step 3 Create a workload.

kubectl apply -f nginx-emptydir.yaml

----End

Handling Local EV Exceptions

If a user manually detaches a disk from ECS or manually runs the **vgremove** command, the EV storage pool may malfunction. To resolve this issue, set the node to be unschedulable by following the procedure described in **Configuring a Node Scheduling Policy in One-Click Mode** and then reset the node.

8.9.4 Using a Temporary Path

A temporary path is of the Kubernetes-native emptyDir type. Its lifecycle is the same as that of a pod. Memory can be specified as the storage medium. When the pod is deleted, the emptyDir volume is deleted and its data is lost.

Using a Temporary Path on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.

Step 3 Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **EmptyDir**.

Step 4 Mount and use storage volumes, as shown in **Table 8-64**. For details about other parameters, see **Workloads**.

Table 8-64 Mounting an EV

Parameter	Description
Storage Medium	**Memory**: • You can select this option to improve the running speed, but the storage capacity is subject to the memory size. This mode is applicable when data volume is small and efficient read and write is required. • If this function is disabled, data is stored in hard disks, which applies to a large amount of data with low requirements on reading and writing efficiency. **NOTE** • If **Memory** is selected, pay attention to the memory size. If the storage capacity exceeds the memory size, an OOM event occurs. • If **Memory** is selected, the size of an EV is the same as pod specifications. • If **Memory** is not selected, EVs will not occupy the system memory.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. Otherwise, containers will be malfunctional. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path is used by default.
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 5 After the configuration, click **Create Workload**.

----End

Using a Temporary Path Through kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-emptydir.yaml** and edit it.

vi nginx-emptydir.yaml

Content of the YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir     # Volume name, which must be the same as the volume name in the volumes field
              mountPath: /tmp        # Location where the emptyDir is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-emptydir         # Volume name, which can be customized
          emptyDir:
            medium: Memory           # EV disk medium: If this parameter is set to Memory, the memory is enabled. If this parameter is left blank, the native default storage medium is used.
            sizeLimit: 1Gi           # Volume capacity
```

Step 3 Create a workload.

kubectl apply -f nginx-emptydir.yaml

----End

8.10 hostPath

hostPath is used for mounting the file directory of the host where the container is located to the specified mount point of the container. If the container needs to access **/etc/hosts**, use hostPath to map **/etc/hosts**.

NOTICE

- Avoid using hostPath volumes as much as possible, as they are prone to security risks. If hostPath volumes must be used, they can only be applied to files or directories and mounted in read-only mode.
- After the pod to which a hostPath volume is mounted is deleted, the data in the hostPath volume is retained.

Mounting a hostPath Volume on the Console

You can mount a path on the host to a specified container path. A hostPath volume is usually used to **store workload logs permanently** or used by workloads that need to **access internal data structure of the Docker engine on the host**.

Step 1 Log in to the CCE console.

Step 2 When creating a workload, click **Data Storage** in **Container Settings**. Click **Add Volume** and choose **hostPath** from the drop-down list.

Step 3 Set parameters for adding a local volume, as listed in **Table 8-65**.

Table 8-65 Setting parameters for mounting a hostPath volume

Parameter	Description
Volume Type	Select **HostPath**.
HostPath	Path of the host to which the local volume is to be mounted, for example, **/etc/hosts**. **NOTE** **HostPath** cannot be set to the root directory **/**. Otherwise, the mounting fails. Mount paths can be as follows: • **/opt/*xxxx*** (excluding **/opt/cloud**) • **/mnt/*xxxx*** (excluding **/mnt/paas**) • **/tmp/*xxx*** • **/var/*xxx*** (excluding key directories such as **/var/lib**, **/var/script**, and **/var/paas**) • **/*xxxx*** (It cannot conflict with the system directory, such as **bin**, **lib**, **home**, **root**, **boot**, **dev**, **etc**, **lost+found**, **mnt**, **proc**, **sbin**, **srv**, **tmp**, **var**, **media**, **opt**, **selinux**, **sys**, and **usr**.) Do not set this parameter to **/home/paas**, **/var/paas**, **/var/lib**, **/var/script**, **/mnt/paas**, or **/opt/cloud**. Otherwise, the system or node installation will fail.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. Otherwise, containers will be malfunctional. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

Parameter	Description
Subpath	Enter the subpath of the storage volume and mount a path in the storage volume to the container. In this way, different folders of the same storage volume can be used in a single pod. **tmp**, for example, indicates that data in the mount path of the container is stored in the **tmp** folder of the storage volume. If this parameter is left blank, the root path is used by default.
Permission	• **Read-only**: You can only read the data in the mounted volumes. • **Read-write**: You can modify the data volumes mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 4 After the configuration, click **Create Workload**.

----End

Mounting a hostPath Volume Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a file named **nginx-hostpath.yaml** and edit it.

vi nginx-hostpath.yaml

The content of the YAML file is as follows. Mount the **/data** directory on the node to the **/data** directory in the container.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-hostpath
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-hostpath
  template:
    metadata:
      labels:
        app: nginx-hostpath
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-hostpath        # Volume name, which must be the same as the volume name in the volumes field.
              mountPath: /data          # Location where the storage volume is mounted
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-hostpath            # Volume name, which can be customized.
          hostPath:
            path: /data                 # Directory location on the host node.
```


Step 3 Create a workload.

kubectl apply -f nginx-hostpath.yaml

----End

8.11 StorageClass

Introduction

A StorageClass in Kubernetes is a resource that categorizes storage types in a cluster. It can also serve as a configuration template for creating PVs. When creating a PVC or PV, you must specify a StorageClass. This resource object defines the storage mode and configuration parameters for dynamic volume provisioning, such as volume type, access mode, and volume lifecycle policy.

When creating a PVC, you only need to specify the **StorageClassName**. Kubernetes will then automatically create the PVs and underlying storage based on the StorageClass. This greatly reduces the need for manual creation and maintenance of PVs.

In addition to the **default StorageClasses** provided by CCE, you can customize StorageClasses. For details, see **Application Scenarios of Custom Storage**.

Creating a StorageClass Using the CCE Console

NOTE

The Everest add-on must be in the running state in the target cluster.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Storage** in the navigation pane. In the right pane, click the **Storage Classes** tab. Click **Create Storage Class** in the upper right corner. In the dialog box displayed, configure parameters.

Parameter	Description
StorageClass Type	Select an underlying storage type.
Name	Enter the StorageClass name. The name of a StorageClass must be unique within a cluster.

Parameter	Description
Reclaim Policy	You can select **Delete** or **Retain** to specify the reclaim policy of the underlying storage when the PVC is deleted. For details, see **PV Reclaim Policy**. • **Delete**: When a PVC is deleted, its associated underlying storage resources will be deleted and the PV resources will be removed. Exercise caution if you select this option. • **Retain**: When a PVC is deleted, both of the PV and its associated underlying storage resources will be retained and the PV is marked as released. If you manually delete the PV afterwards, the underlying storage resources will not be deleted. To bind the PV to a new PVC, you need to remove the original binding information from the PV.
Binding Mode	Time when a PV is dynamically created, which can be created immediately or with a delay. • **Immediate**: After a PVC is created, the storage resources and PV will be created and associated with the PVC without delay. Local PVs do not support **Immediate**. • **WaitForFirstConsumer**: After a PVC is created, it will not be immediately bound to a PV. Instead, the storage resources and PV will be generated and bound to the PVC only after the pod that requires the PVC is scheduled. OBS, SFS, and SFS Turbo do not support **WaitForFirstConsumer**.

Step 3 Click **Create**. On the **Storage Classes** tab page, view the created StorageClass and its information.

----End

Creating a StorageClass Using a YAML File

As of now, CCE provides StorageClasses such as csi-disk, csi-nas, and csi-obs by default. When defining a PVC, you can use a **StorageClassName** to automatically create a PV of the corresponding type and automatically create underlying storage resources.

Run the following kubectl command to obtain the StorageClasses that CCE supports. Use the CSI add-on provided by CCE to create a StorageClass.

```
# kubectl get sc
NAME                PROVISIONER                  AGE
csi-disk            everest-csi-provisioner      17d      # EVS disk
csi-disk-topology   everest-csi-provisioner      17d      # EVS disks created with delay
csi-nas             everest-csi-provisioner      17d      # SFS 1.0
csi-sfs             everest-csi-provisioner      17d      # SFS 3.0
csi-obs             everest-csi-provisioner      17d      # OBS
csi-sfsturbo        everest-csi-provisioner      17d      # SFS Turbo
csi-local           everest-csi-provisioner      17d      # Local PV
csi-local-topology  everest-csi-provisioner      17d      # Local PV created with delay
```

Each StorageClass contains the default parameters used for dynamically creating a PV. The following is an example of StorageClass for EVS disks:


```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS
  everest.io/passthrough: 'true'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Table 8-66 Key parameters

Parameter	Description
provisioner	Specifies the storage resource provider, which is the Everest add-on for CCE. Set this parameter to **everest-csi-provisioner**.
parameters	Specifies the storage parameters, which vary with storage types. For details, see **Table 8-67**.
reclaimPolicy	Specifies the value of **persistentVolumeReclaimPolicy** for creating a PV. The value can be **Delete** or **Retain**. If **reclaimPolicy** is not specified when a StorageClass object is created, the value defaults to **Delete**. • **Delete**: When a PVC is deleted, its associated underlying storage resources will be deleted and the PV resources will be removed. Exercise caution if you select this option. • **Retain**: When a PVC is deleted, both of the PV and its associated underlying storage resources will be retained and the PV is marked as released. If you manually delete the PV afterwards, the underlying storage resources will not be deleted. To bind the PV to a new PVC, you need to remove the original binding information from the PV.
allowVolumeExpansion	Specifies whether the PV of this StorageClass supports dynamic capacity expansion. The default value is **false**. Dynamic capacity expansion is implemented by the underlying storage add-on. This is only a switch.
volumeBindingMode	Specifies the volume binding mode, that is, the time when a PV is dynamically created. The value can be **Immediate** or **WaitForFirstConsumer**. • **Immediate**: After a PVC is created, the storage resources and PV will be created and associated with the PVC without delay. • **WaitForFirstConsumer**: After a PVC is created, it will not be immediately bound to a PV. Instead, the storage resources and PV will be generated and bound to the PVC only after the pod that requires the PVC is scheduled.
mountOptions	This field must be supported by the underlying storage. If this field is not supported but is specified, the PV creation will fail.

Table 8-67 Parameters

Volume Type	Parameter	Mandatory	Description
EVS	csi.storage.k8s.io/csi-driver-name	Yes	Driver type. If an EVS disk is used, the parameter value is fixed at **disk.csi.everest.io**.
	csi.storage.k8s.io/fstype	Yes	If an EVS disk is used, the parameter value can be **ext4**.
	everest.io/disk-volume-type	Yes	EVS disk type. All letters are in uppercase. • **SAS**: high I/O • **SSD**: ultra-high I/O • **GPSSD**: general-purpose SSD • **ESSD**: extreme SSD • **GPSSD2**: general-purpose SSD v2, which is supported when the Everest version is 2.4.4 or later and the **everest.io/disk-iops** and **everest.io/disk-throughput** annotations are configured
	everest.io/passthrough	Yes	The parameter value is fixed at **true**, which indicates that the EVS device type is **SCSI**. Other parameter values are not allowed.
	everest.io/disk-iops	No	Preconfigured IOPS, which is supported only by general-purpose SSD v2 and extreme SSD v2 EVS disks. • The IOPS of general-purpose SSD v2 EVS disks ranges from 3000 to 128000, and the maximum value is 500 times of the capacity (GiB). If the IOPS of general-purpose SSD v2 disks is greater than 3000, extra IOPS will be billed. For details, see **Price Calculator**. • The IOPS of extreme SSD v2 disks ranges from 100 to 256000, and the maximum value is 1000 times of the capacity (GiB). The IOPS of extreme SSD v2 disks will be billed separately. For details, see **Price Calculator**.

Volume Type	Parameter	Mandatory	Description
	everest.io/disk-throughput	No	Preconfigured throughput, which is supported only by general-purpose SSD v2 EVS disks. The value ranges from 125 MiB/s to 1000 MiB/s. The maximum value is a quarter of IOPS. If the throughput is greater than 125 MiB/s, extra throughput will be billed. For details, see **Price Calculator**.
SFS	csi.storage.k8s.io/csi-driver-name	Yes	Driver type. If SFS is used, the parameter value is fixed at **nas.csi.everest.io**.
	csi.storage.k8s.io/fstype	Yes	If SFS is used, the value can be **nfs**.
	everest.io/share-access-level	Yes	The parameter value is fixed at **rw**, indicating that the SFS data is readable and writable.
	everest.io/share-access-to	Yes	VPC ID of the cluster.
	everest.io/share-is-public	No	The parameter value is fixed at **false**, indicating that the file is shared to private. When you use SFS 3.0, there is no need to configure this parameter.
	everest.io/sfs-version	No	This parameter is only required for SFS 3.0 and its value is fixed at **sfs3.0**.
SFS Turbo	csi.storage.k8s.io/csi-driver-name	Yes	Driver type. If SFS Turbo is used, the parameter value is fixed at **sfsturbo.csi.everest.io**.
	csi.storage.k8s.io/fstype	Yes	If SFS Turbo is used, the value can be **nfs**.
	everest.io/share-access-to	Yes	VPC ID of the cluster.
	everest.io/share-expand-type	No	Extension type. The default value is **bandwidth**, indicating an enhanced file system. This parameter does not take effect.
	everest.io/share-source	Yes	The parameter value is fixed at **sfs-turbo**.

Volume Type	Parameter	Mandatory	Description
	everest.io/share-volume-type	No	SFS Turbo StorageClass. The default value is **STANDARD**, indicating standard and standard enhanced editions. This parameter does not take effect.
OBS	csi.storage.k8s.io/csi-driver-name	Yes	Driver type. If OBS is used, the parameter value is fixed at **obs.csi.everest.io**.
	csi.storage.k8s.io/fstype	Yes	Instance type, which can be **obsfs** or **s3fs**. • **obsfs**: a parallel file system • **s3fs**: object bucket
	everest.io/obs-volume-type	Yes	OBS StorageClass. • If **fsType** is set to **s3fs**, **STANDARD** (standard buckets) and WARM (infrequent access buckets) are supported. • This parameter is invalid when **fsType** is set to **obsfs**.

Application Scenarios of Custom Storage

When using storage resources in CCE, the most common method is to specify **StorageClassName** to define the type of storage resources to be created when creating a PVC. The following configuration shows how to use a PVC to apply for a SAS (high I/O) EVS disk (block storage).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-example
  namespace: default
  annotations:
    everest.io/disk-volume-type: SAS
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

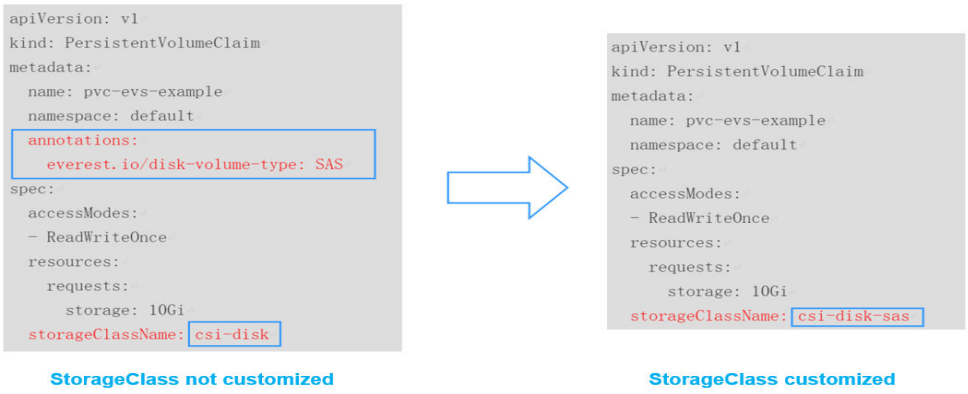
To specify the EVS disk type on CCE, use the **everest.io/disk-volume-type** field. SAS indicates the EVS disk type.

The preceding is a basic method of using StorageClass. In real-world scenarios, you can use StorageClass to perform other operations.

Application Scenario	Solution	Procedure
When **annotations** is used to specify storage configuration, the configuration is complex. For example, the **everest.io/disk-volume-type** field is used to specify the EVS disk type.	Define PVC annotations in the **parameters** field of StorageClass. When compiling a YAML file, you only need to specify **StorageClassName**. For example, you can define SAS EVS disk and SSD EVS disk as a StorageClass, respectively. If a StorageClass named **csi-disk-sas** is defined, it is used to create SAS storage.	**Scenario 1: Specifying the Disk Type in a StorageClass**
When a user migrates services from an on-premises Kubernetes cluster or other Kubernetes services to CCE, the StorageClass used in the original application YAML file is different from that used in CCE. As a result, a large number of YAML files or Helm chart packages need to be modified when the storage is used, which is complex and error-prone.	Create a StorageClass with the same name as that in the original application YAML file in the CCE cluster. After the migration, you do not need to modify the **StorageClassName** in the application YAML file. For example, the EVS disk StorageClass used before the migration is **disk-standard**. After migrating services to a CCE cluster, you can copy the YAML file of the **csi-disk** StorageClass in the CCE cluster, change its name to **disk-standard**, and create another StorageClass.	
StorageClassName must be specified in the YAML file to use the storage. If not, the storage cannot be created.	If you set the default StorageClass in the cluster, you can create storage without specifying the **StorageClassName** in the YAML file.	**Scenario 2: Specifying the Default StorageClass**
When creating storage resources, you must specify an enterprise project. Configuring annotations in each PVC can be a complex process.	Add an enterprise project to a StorageClass. In this case, you do not need to specify an enterprise project when creating a PVC. This is because storage resources will automatically be created in the enterprise project specified in the StorageClass.	**Scenario 3: Specifying an Enterprise Project for a StorageClass**

Scenario 1: Specifying the Disk Type in a StorageClass

This section uses the custom StorageClass of EVS disks as an example to describe how to define SAS EVS disk and SSD EVS disk as a StorageClass, respectively. For example, if you define a StorageClass named **csi-disk-sas**, which is used to create SAS storage, the differences are shown in the following figure. When editing a PVC's YAML file, you only need to specify **StorageClassName**.



- You can customize a high I/O StorageClass in a YAML file. For example, the name **csi-disk-sas** indicates that the disk type is SAS (high I/O).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-sas                          # Name of the high I/O StorageClass, which can be customized
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
   everest.io/disk-volume-type: SAS           # High I/O EVS disk type, which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true                    # true indicates that capacity expansion is allowed.
```

- For an ultra-high I/O StorageClass, you can set the class name to **csi-disk-ssd** to create SSD EVS disk (ultra-high I/O).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-ssd                          # Name of the ultra-high I/O StorageClass, which can be customized
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SSD            # Ultra-high I/O EVS disk type, which cannot be customized.
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
```

reclaimPolicy: indicates the reclaim policies of the underlying cloud storage. The value can be **Delete** or **Retain**.

- **Delete**: When a PVC is deleted, both the PV and the EVS disk are deleted.
- **Retain**: When a PVC is deleted, both the PV and underlying storage resources will be retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.

If high data security is required, select **Retain** to prevent data from being deleted by mistake.

After the definition is complete, run the **kubectl create** commands to create storage resources.

```
# kubectl create -f sas.yaml
storageclass.storage.k8s.io/csi-disk-sas created
# kubectl create -f ssd.yaml
storageclass.storage.k8s.io/csi-disk-ssd created
```

Query **StorageClass** again. The command output is as follows:

```
# kubectl get sc
NAME                PROVISIONER                  AGE
csi-disk            everest-csi-provisioner      17d
csi-disk-sas        everest-csi-provisioner      2m28s
csi-disk-ssd        everest-csi-provisioner      16s
csi-disk-topology   everest-csi-provisioner      17d
csi-nas             everest-csi-provisioner      17d
csi-obs             everest-csi-provisioner      17d
csi-sfsturbo        everest-csi-provisioner      17d
```

Scenario 2: Specifying the Default StorageClass

You can specify a StorageClass as the default class. In this way, any PVCs created without specifying **StorageClassName** will automatically use the default StorageClass.

For example, to specify **csi-disk-ssd** as the default StorageClass, edit your YAML file as follows:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk-ssd
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"   # Specify the default StorageClass in a cluster. There can only be one default StorageClass per cluster.
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SSD
  everest.io/passthrough: "true"
provisioner: everest-csi-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
```

Run the **kubectl create** command to create a StorageClass and then check it.

```
# kubectl create -f ssd.yaml
storageclass.storage.k8s.io/csi-disk-ssd created
# kubectl get sc
NAME                    PROVISIONER                  AGE
csi-disk                everest-csi-provisioner      17d
csi-disk-sas            everest-csi-provisioner      114m
csi-disk-ssd (default)  everest-csi-provisioner      9s
csi-disk-topology       everest-csi-provisioner      17d
csi-nas                 everest-csi-provisioner      17d
csi-obs                 everest-csi-provisioner      17d
csi-sfsturbo            everest-csi-provisioner      17d
```

Scenario 3: Specifying an Enterprise Project for a StorageClass

When creating EVS or OBS PVCs using a StorageClass in CCE, you can specify an enterprise project to assign the created storage resources (EVS disks and OBS) to.

This enterprise project can either be the default one or the same one as the cluster belongs to.

If you do not specify any enterprise project, the enterprise project in StorageClass will be used by default. The storage resources created using the CCE's csi-disk or csi-obs StorageClass belong to the **default** enterprise project.

If you want the storage resources created using a StorageClass to be in the same enterprise project as the cluster, you can customize a StorageClass and specify the enterprise project ID in it.

NOTE

To use this function, the Everest add-on must be upgraded to v1.2.33 or later.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk-epid      # Customize a StorageClass name.
provisioner: everest-csi-provisioner
parameters:
  csi.storage.k8s.io/csi-driver-name: disk.csi.everest.io
  csi.storage.k8s.io/fstype: ext4
  everest.io/disk-volume-type: SAS
  everest.io/enterprise-project-id: 86bfc701-9d9e-4871-a318-6385aa368183  # Specify the enterprise project
ID.
  everest.io/passthrough: 'true'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```


9 Auto Scaling

9.1 Overview

Auto scaling is a service that automatically and economically adjusts service resources based on your service requirements and configured policies.

Context

More and more applications run on Kubernetes. It becomes increasingly important to quickly scale out applications on Kubernetes to cope with service peaks and to scale in applications during off-peak hours to save resources and reduce costs.

In a Kubernetes cluster, auto scaling involves pods and nodes. A pod is an application instance. Each pod contains one or more containers and runs on a node (VM or bare-metal server). If a cluster does not have sufficient nodes to run new pods, add nodes to the cluster to ensure service running.

In CCE, auto scaling is used for online services, large-scale computing and training, deep learning GPU or shared GPU training and inference, periodic load changes, and many other scenarios.

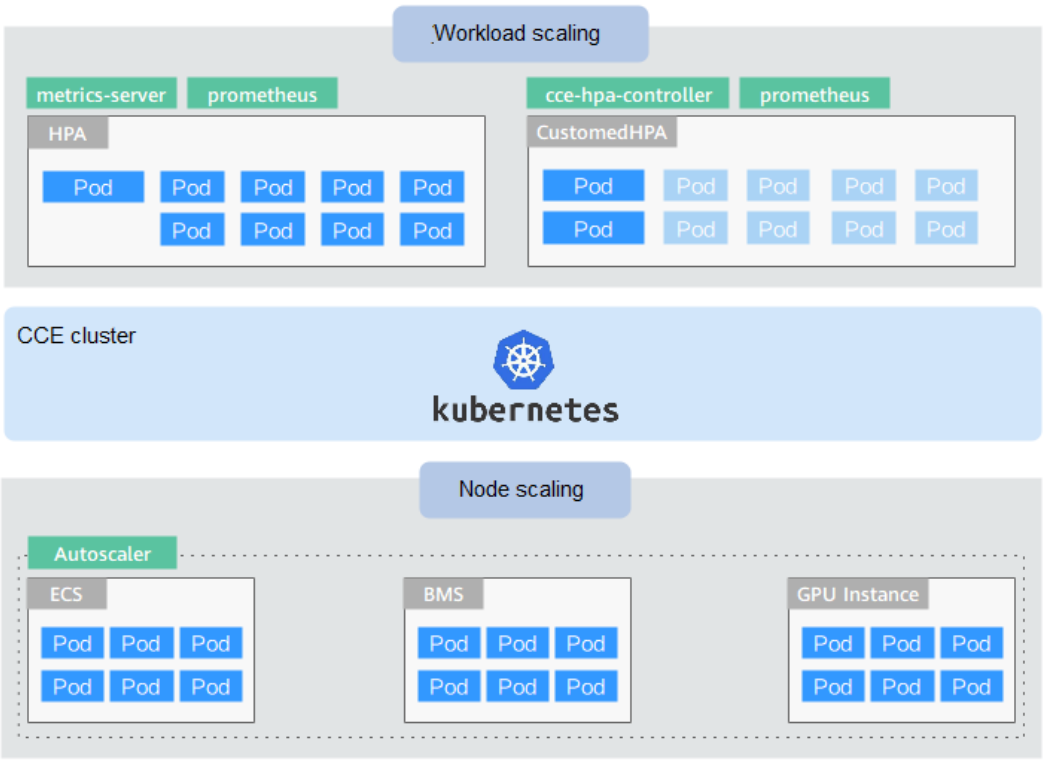
Auto Scaling in CCE

CCE supports auto scaling for workloads and nodes.

- **Workload scaling:** Auto scaling at the scheduling layer to change the scheduling capacity of workloads. For example, you can use the HPA, a scaling component at the scheduling layer, to adjust the number of replicas of an application. Adjusting the number of replicas changes the scheduling capacity occupied by the current workload, thereby enabling scaling at the scheduling layer.
- **Node scaling:** Auto scaling at the resource layer. When the planned cluster nodes cannot allow workload scheduling, ECS resources are provided to support scheduling.

Workload scaling and node scaling can work separately or together. For details, see **Using HPA and CA for Auto Scaling of Workloads and Nodes**.

Components



Workload Scaling Types

Table 9-1 Workload scaling types

Type	Component	Component Description	Reference
HPA	HorizontalPodAutoscaler (built-in Kubernetes component)	HorizontalPodAutoscaler is a built-in component of Kubernetes for Horizontal Pod Autoscaling (HPA). CCE incorporates the application-level cooldown time window and scaling threshold functions into Kubernetes HPA.	**Creating an HPA Policy**
CustomedHPA	**CCE Advanced HPA**	An enhanced auto scaling feature, used for auto scaling of Deployments based on metrics (CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year).	**Creating a CustomedHPA Policy**

Type	Component	Component Description	Reference
CronHPA	CCE Advanced HPA	CronHPA can scale in or out a cluster at a fixed time. It can work with HPA policies to periodically adjust the HPA scaling scope, implementing workload scaling in complex scenarios.	Creating a Scheduled CronHPA Policy

Node Scaling Types

Table 9-2 Node scaling types

Component Name	Component Description	Application Scenario	Reference
CCE Cluster Autoscaler	An open source Kubernetes component for horizontal scaling of nodes, which is optimized by CCE in scheduling, auto scaling, and costs.	Online services, deep learning, and large-scale computing with limited resource budgets	Creating a Node Scaling Policy
CCE Cloud Bursting Engine for CCI	Used to extend Kubernetes APIs to serverless container platforms (such as CCI), which means you no longer have to worry about node resources.	Online traffic surge, CI/CD, big data, and more	Elastic Scaling of CCE Pods to CCI

9.2 Scaling a Workload

9.2.1 Workload Scaling Rules

CCE supports multiple workload scaling modes. Comparisons between the scaling policies are listed in the following table.

Table 9-3 Comparisons between auto scaling policies

Item	HPA	CronHPA	CustomedHPA	VPA	AHPA
Introduction	**Horizontal Pod Autoscaling**	Enhanced based on HPA, CronHPA is mainly used if the resource usage of applications changes periodically.	Enhanced CCE auto scaling that is triggered based on metrics or at a scheduled time.	Vertical Pod Autoscaler in Kubernetes.	Advanced Horizontal Pod Autoscaler, which performs scaling beforehand based on historical data.
Rule	Scales Deployment **pods** based on **metrics** (CPU usage and memory usage).	Scales Deployment **pods periodically** (daily, weekly, monthly, or yearly at a specific time).	Scales Deployment **pods** based on **metrics** (CPU usage and memory usage) or **periodically** (daily, weekly, monthly, or yearly at a specific time).	Scales **resources** requested by a workload based on the **historical usage** of container resources (CPU and memory).	Predicts the resources needed based on the **historical usage** of container resources (CPU and memory) and automatically scales workload **pods** beforehand.

Item	HPA	CronHPA	CustomedHPA	VPA	AHPA
Enhancement	Adds the application-level cooldown time window and scaling threshold functions based on the Kubernetes HPA.	Compatible with HPA objects, allowing you to use both CronHPA and HPA. • If both CronHPA and HPA are used, CronHPA runs based on HPA and periodically adjusts the number of pods for HPA. • If CronHPA is separately used: CronHPA periodically adjusts the number of pods for workloads.	**Metric-based**: • Scaling can be performed based on the percentage of the current number of pods. • The minimum scaling step can be set. Scaling can be performed step by step. • Different scaling operations can be performed based on the actual metric values. **Periodic**: You can select a specific time point every day, every week, every month, or every year or a period as the trigger time.	Calculates the optimal value by analyzing historical CPU and memory usage and adjusts the requested pod resources accordingly.	Identifies periods of pod scaling and predicts future fluctuations by analyzing historical service metrics. This proactive approach can resolve the issue of delayed scaling in native HPA.

Item	HPA	CronHPA	CustomedHPA	VPA	AHPA
Usage	Creating an HPA Policy	Creating a Scheduled CronHPA Policy	Creating a CustomedHPA Policy	Creating a VPA Policy	Creating an AHPA Policy

How HPA Works

HPA is a controller that controls horizontal pod scaling. HPA periodically checks the pod metrics, calculates the number of pods required to meet the target values configured for HPA resources, and then adjusts the value of the **replicas** field in the target resource object (such as a Deployment).

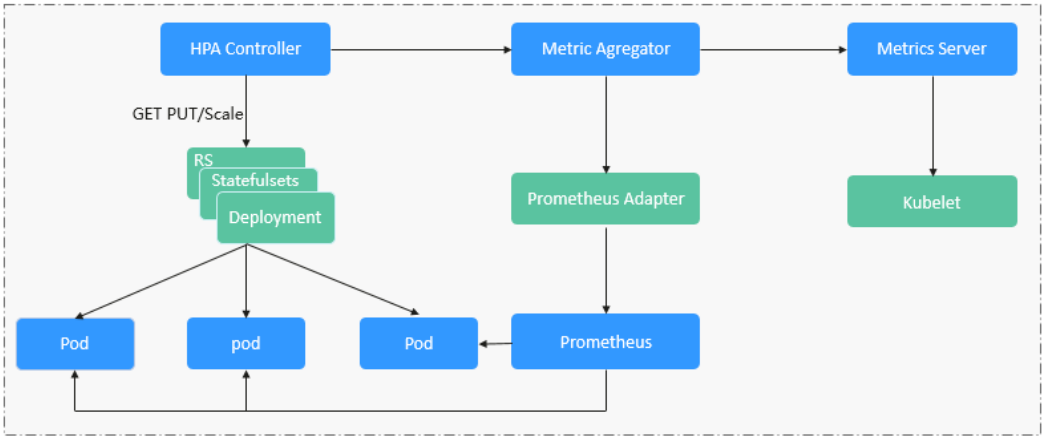
A prerequisite for auto scaling is that your container running data can be collected, such as number of cluster nodes/pods, and CPU and memory usage of containers. Kubernetes does not have built-in monitoring capabilities, but you can use extensions like **Prometheus** and **Metrics Server** to monitor and collect data.

- **Prometheus** is an open-source monitoring and alarming framework that can collect multiple types of metrics. Prometheus has been a standard monitoring solution of Kubernetes.
- **Metrics Server** is a cluster-wide aggregator of resource utilization data. Metrics Server collects metrics from the Summary API exposed by kubelet. These metrics are set for core Kubernetes resources, such as pods, nodes, containers, and Services. Metrics Server provides a set of standard APIs for external systems to collect these metrics.

HPA can work with Metrics Server for auto scaling based on the CPU and memory usage. It can also work with Prometheus for auto scaling based on custom monitoring metrics.

Figure 9-1 shows how HPA works.

Figure 9-1 HPA working process



Two core modules of HPA:

- Data Source Monitoring

  The community provided only CPU- and memory-based HPA at the early stage. With the population of Kubernetes and Prometheus, developers need more custom metrics or monitoring information at the access layer for their own applications, for example, the QPS of the load balancer and the number of online users of the website. In response, the community defines a set of standard metric APIs to provide services externally through these aggregated APIs.

  - **metrics.k8s.io** provides monitoring metrics related to the CPU and memory of pods and nodes.
  - **custom.metrics.k8s.io** provides custom monitoring metrics related to Kubernetes objects.
  - **external.metrics.k8s.io** provides metrics that come from external systems and are irrelevant to any Kubernetes resource metrics.

- Scaling Decision-Making Algorithms

  The HPA controller calculates the scaling ratio based on the current metric values and desired metric values using the following formula:

  Desired number of pods = Rounded up value of [Number of current pods x (Current metric value/Target value)]

  For example, if the current metric value is 200m and the target value is 100m, the desired number of pods will be doubled according to the formula. In practice, pods may be constantly added or reduced. To ensure stability, the HPA controller is optimized from the following aspects:

  - Cooldown interval: In v1.11 and earlier versions, Kubernetes introduced the startup parameters **horizontal-pod-autoscaler-downscale-stabilization-window** and **horizontal-pod-autoScaler-upscale-stabilization-window** to indicate the cooldown intervals after a scale-in and scale-out, respectively, in which no scaling operation will not be performed. In versions later than v1.14, the scheduling queue is introduced to store all decision-making suggestions detected within a period of time. Then, the system makes decisions based on all valid decision-making suggestions to minimize changes of the desired number of pods to ensure stability.
  - Tolerance: It can be considered as a buffer zone. If the pod number changes can be tolerated, the number of pods remains unchanged.

    ratio = Current metric value/Target value

    When |ratio – 1.0| ≤ tolerance, scaling will not be performed.

    When |ratio – 1.0| > tolerance, the desired value is calculated using the formula mentioned above.

    The default value is 0.1 in the current community version.

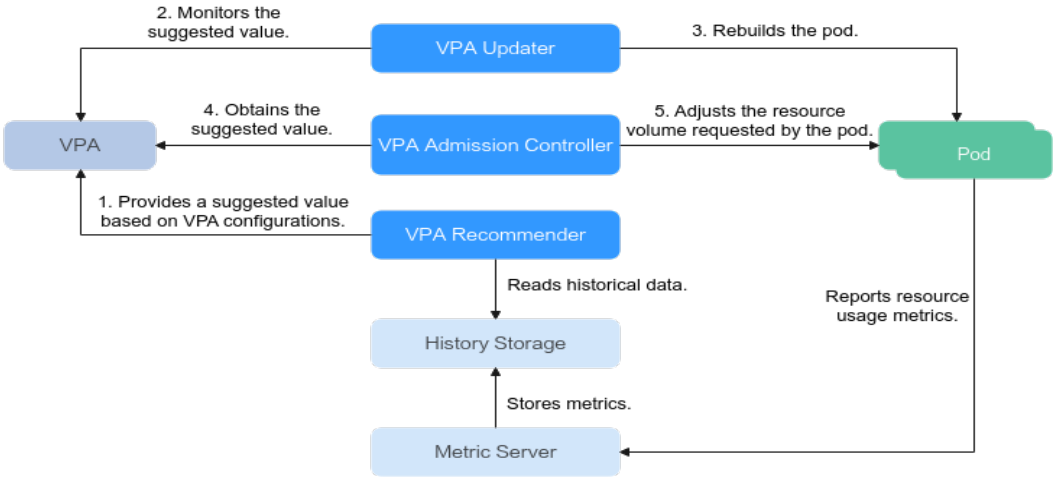
The HPA performs scaling based on metric thresholds. Common metrics include the CPU and memory usage. You can also set custom metrics, such as the QPS and number of connections, to trigger scaling. However, metric-based scaling brings in latency of minutes generated during data collection, determination, and scaling phases. Such latency may cause high CPU usage and slow response. To solve this problem, CCE allows you to configure scheduled policies to scale resources regularly for applications with periodic changes.

How VPA Works

VPA runs using the following components:

- VPA Recommender: uses historical data to suggest how to modify the resources allocated to a pod.
- VPA Updater: checks whether the suggested value is the same as the current value. If they are different, it will rebuild the pod.
- VPA Admission Controller: adjusts the requested resource volume to match the suggested value during pod rebuilding.

Figure 9-2 VPA working process



The following describes how VPA works:

1. VPA Recommender suggests modifications to the resources allocated to a pod based on the historical pod resource usage.
2. VPA Updater checks whether the suggested value is the same as the current value. If they are different, it will rebuild the pod.
3. During pod rebuilding, VPA Admission Controller changes the value requested by the pod to the suggested value.

9.2.2 Creating an HPA Policy

Horizontal Pod Autoscaling (HPA) in Kubernetes implements horizontal scaling of pods. In a CCE HPA policy, you can configure different cooldown time windows and scaling thresholds for different applications based on the Kubernetes HPA.

Prerequisites

To use HPA, install an add-on that provides metrics APIs. Select one of the following add-ons based on your cluster version and service requirements.

- **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
- **Cloud Native Cluster Monitoring**: available only in clusters of v1.17 or later.

  - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**.
  - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see **Creating an HPA Policy Using Custom Metrics**.
- **Prometheus (EOM)**: Prometheus needs to be registered as a metrics API. For details, see **Providing Resource Metrics Through the Metrics API**. This add-on supports only clusters of v1.21 or earlier.

Notes and Constraints

- HPA policies can be created only for clusters of v1.13 or later.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

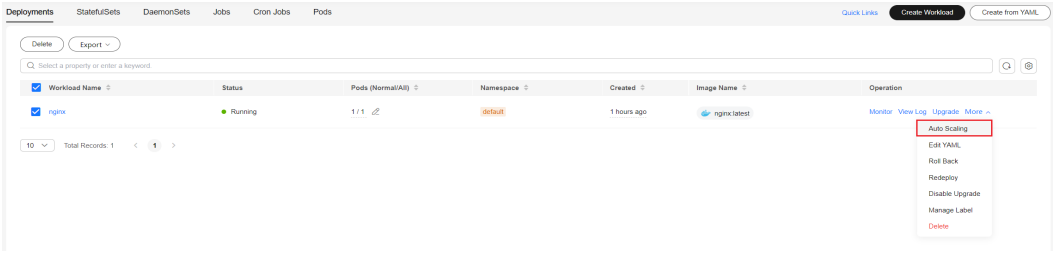
  For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **Auto Scaling** in the **Operation** column.

Figure 9-3 Scaling a workload



Step 3 Set **Policy Type** to **HPA+CronHPA**, enable the created HPA policy, and configure parameters.

This section describes only HPA policies. To enable CronHPA, see **Creating a Scheduled CronHPA Policy**.

Figure 9-4 Enabling the HPA policy

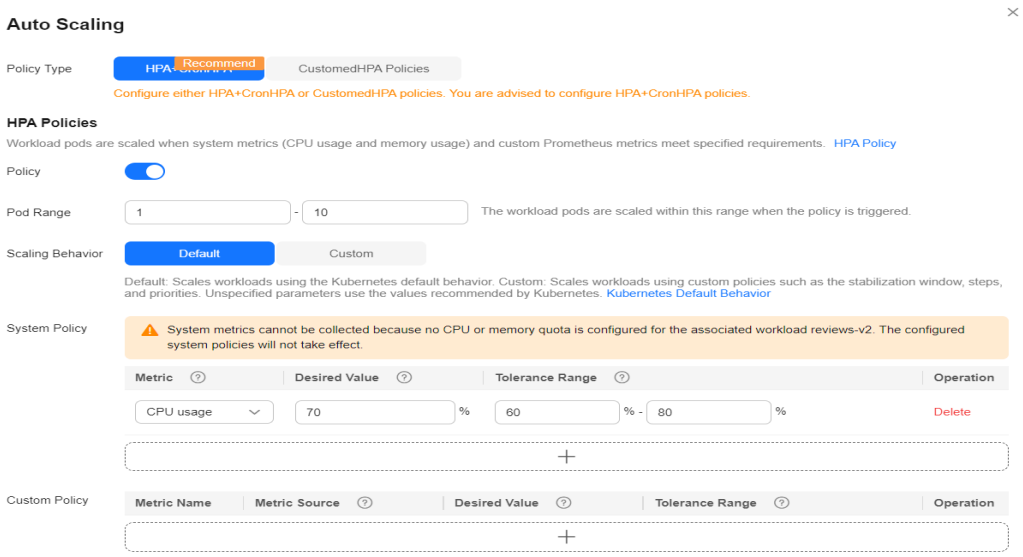


Table 9-4 HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range. **NOTICE** In CCE Turbo clusters, if you use a dedicated load balancer for your workload, the number of pods cannot exceed the backend server group quota of the load balancer, which is 500 by default. If you exceed this limit, you will not be able to add any more pods to the load balancer backend.
Cooldown Period	Interval between a scale-in and a scale-out. The unit is minute. **The interval cannot be shorter than 1 minute.** **This parameter is supported only in clusters of v1.15 to v1.23.** This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.

Parameter	Description
Scaling Behavior	**This parameter is supported only in clusters of v1.25 or later.** • **Default**: scales workloads using the Kubernetes default behavior. For details, see **Default Behavior**. • **Custom**: scales workloads using custom policies such as stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. – **Disable scale-out/scale-in**: Select whether to disable scale-out or scale-in. – **Stabilization Window**: a period during which CCE continuously checks whether the metrics used for scaling keep fluctuating. CCE triggers scaling if the desired state is not maintained for the entire window. This window restricts the unwanted flapping of pod count due to metric changes. – **Step**: specifies the scaling step. You can set the number or percentage of pods to be scaled in or out within a specified period. If there are multiple policies, you can select the policy that maximizes or minimizes the number of pods.
System Policy	• **Metric**: You can select **CPU usage** or **Memory usage**. **NOTE** Usage = Average resource usage of all pods in a workload/ Requested resources • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. Number of target pods (rounded up) = (Current metric value/Desired value) x Number of current pods **NOTE** When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes. • **Tolerance Range**: Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range. If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling is triggered. **This parameter is supported only in clusters of v1.15 or later.**

Parameter	Description
Custom Policy (supported only in clusters of v1.15 or later)	**NOTE** Before creating a custom policy, install an add-on that supports custom metric collection (for example, Prometheus) in the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**. • **Metric Name**: name of the custom metric. You can select a name as prompted. • **Metric Source**: Select an object type from the drop-down list. You can select **Pod**. • **Desired Value**: the average metric value of all pods. Number of pods to be scaled (rounded up) = (Current metric value/ Desired value) x Number of current pods **NOTE** When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes. • **Tolerance Range**: Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range.

Step 4 Click **Create**.

----End

9.2.3 Creating an HPA Policy with Custom Metrics

Kubernetes' default HPA policy only allows for auto scaling based on CPU and memory usage. However, in more complex service scenarios, this may not be sufficient to meet routine O&M requirements. To resolve this issue, you can configure HPA policies with custom metrics, allowing for flexible workload scaling.

This section uses an example to describe how to deploy an Nginx application, which uses the **container_cpu_usage_core_per_second** metric exposed by Prometheus to identify the number of CPU cores used by a container on a per-second basis. For more information about Prometheus metrics, see **METRIC TYPES**.

Step 1: Install Cloud Native Cluster Monitoring

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.

Step 2 Locate the **Cloud Native Cluster Monitoring** add-on and click **Install**.

Prioritize the configurations listed below and adjust any other configurations as needed. For details, see **Cloud Native Cluster Monitoring**.

- **Data Storage Configuration**: Local data storage is mandatory, and other configurations are optional.
- **Custom Metric Collection**: Enable this option in this example. If this option is not enabled, custom metrics cannot be collected.

Step 3 Click **Install**.

----End

Step 2: Create a Workload

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane of the cluster console, choose **Workloads**. Then, click **Create Workload** in the upper right corner. Create an Nginx workload. For details, see **Creating a Deployment**.

----End

Step 3: Modify the Configuration File

Step 1 In the navigation pane of the cluster console, choose **ConfigMaps and Secrets** and switch to the **monitoring** namespace.

Step 2 Update **user-adapter-config**. You can modify the **rules** field in **user-adapter-config** to convert the metrics exposed by Prometheus to metrics that can be associated with HPA.

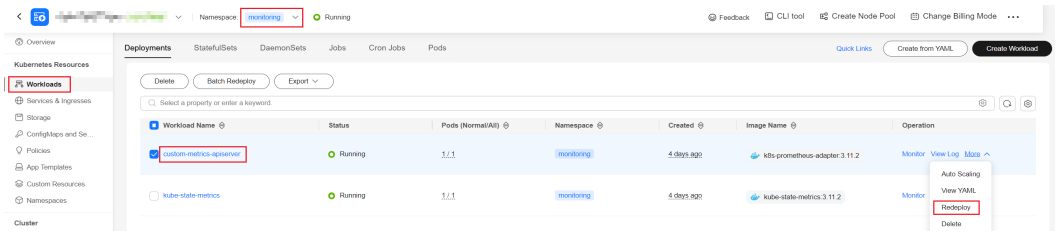
Add the following example rule:

```
rules:
  - seriesQuery: 'container_cpu_usage_seconds_total{namespace!="",pod!=""}'
    seriesFilters: []
    resources:
      overrides:
        namespace:
          resource: namespace
        pod:
          resource: pod
    name:
      matches: "^(.*)_seconds_total"
      as: "${1}_core_per_second"
    metricsQuery: 'sum(rate(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<<.GroupBy>>)'
```

In this example, the existing **container_cpu_usage_seconds_total** metrics are aggregated into the **container_cpu_usage_core_per_second** metric, which is then used for HPA policies. For details, see **Metrics Discovery and Presentation Configuration**.

- **seriesQuery**: PromQL request data, which specifies the metrics to be obtained by users. You can configure this parameter as needed.
- **metricsQuery**: aggregates the PromQL query data in **seriesQuery**.
- **resources**: data labels in PromQL, which are used to match resources. The resources here refer to api-resource such as pods, namespaces, and nodes in a cluster. You can run **kubectl api-resources -o wide** to check the resources. The key corresponds to **LabelName** in the Prometheus data. Ensure that the Prometheus metric data contains **LabelName**.
- **name**: indicates that Prometheus metric names are converted to readable metric names based on regular expression matching. In this example, **container_cpu_usage_seconds_total** is converted to **container_cpu_usage_core_per_second**.

Step 3 Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.



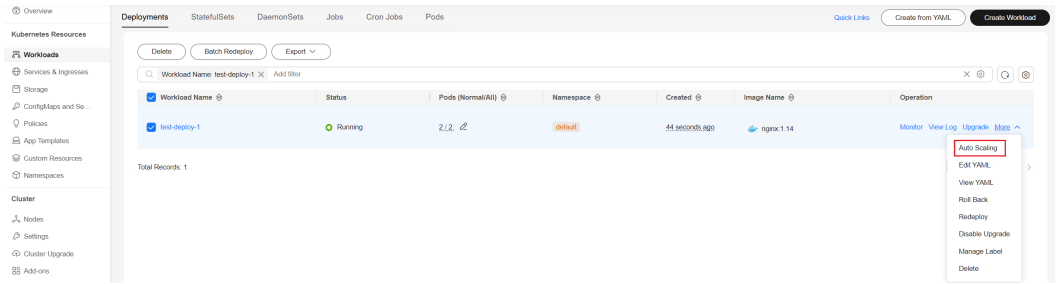
Step 4 Run the following command to check whether the metric has been added:

```
kubectl get --raw  "/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pods/*/
container_cpu_usage_core_per_second"
```

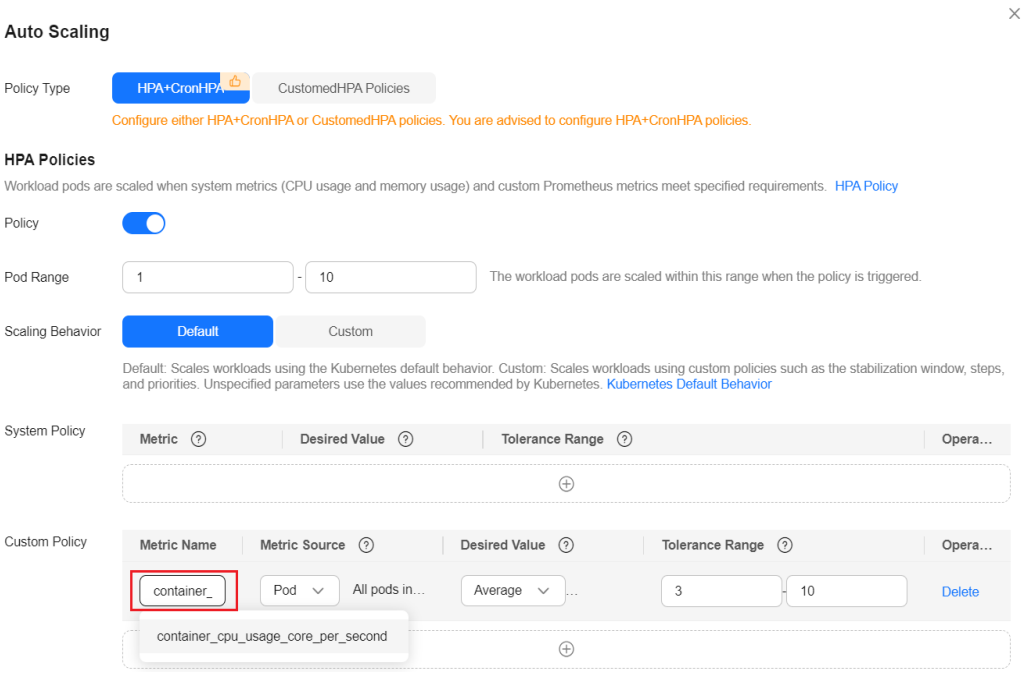
----End

Step 4: Verify HPA Scaling

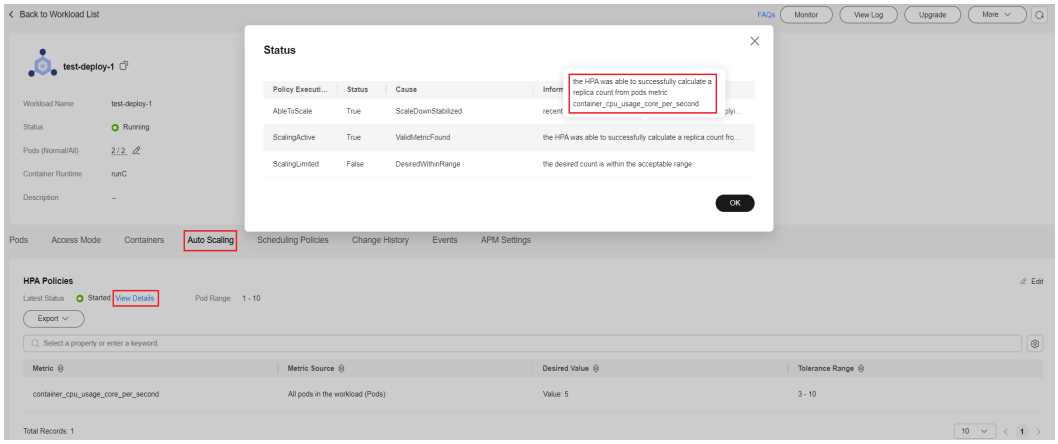
Step 1 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **Auto Scaling** in the **Operation** column.



Step 2 Set **Policy Type** to **HPA+CronHPA** and enable an HPA policy. You can use the custom metrics configured in **rules** to create the HPA policy.



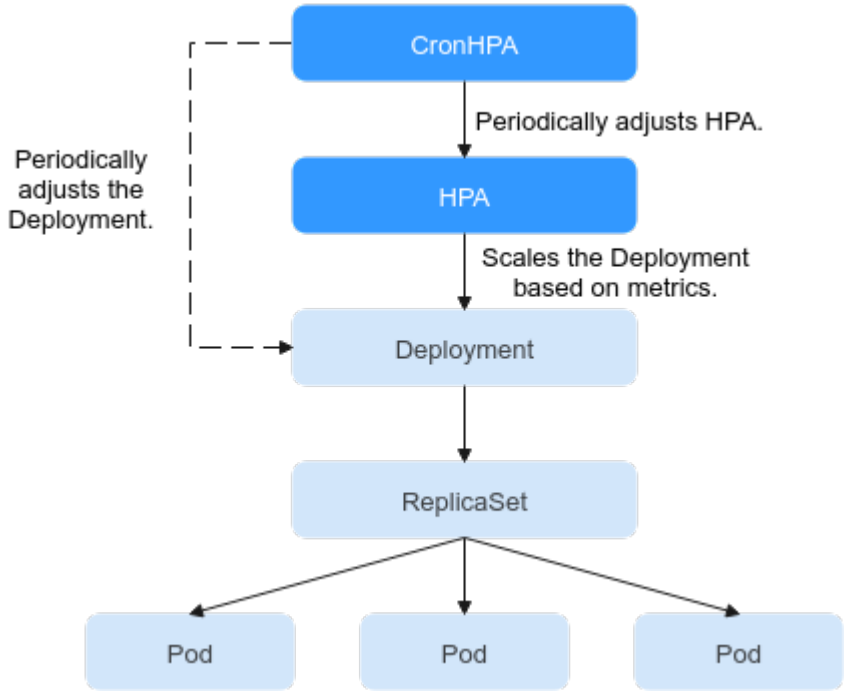
Step 3 Click the workload name and switch to the **Auto Scaling** tab page. You can find that the HPA policy has been triggered.



----End

9.2.4 Creating a Scheduled CronHPA Policy

There are predictable and unpredictable traffic peaks for some services. For such services, CCE CronHPA allows you to scale resources in fixed periods. It can work with HPA policies to periodically adjust the HPA scaling scope, implementing workload scaling.



CronHPA can periodically adjust the maximum and minimum numbers of pods in the HPA policy or directly adjust the number of pods of a Deployment.

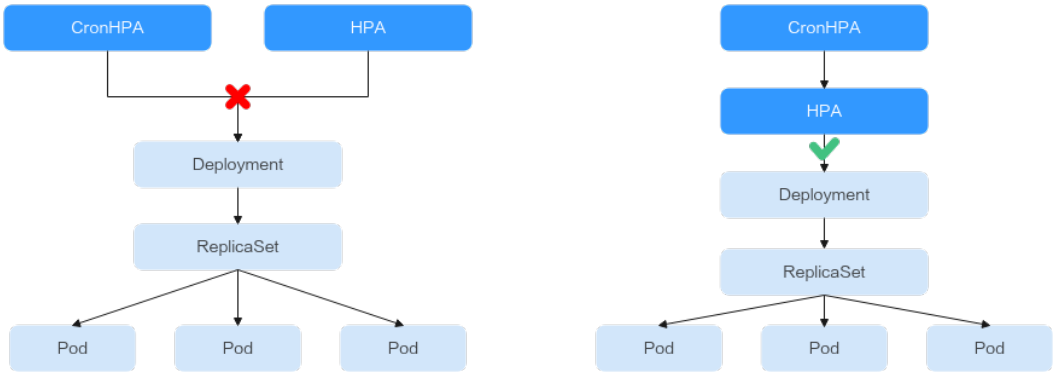
Prerequisites

The add-on **CCE Advanced HPA** of v1.2.13 or later has been installed.

Using CronHPA to Adjust the HPA Scaling Scope

CronHPA can periodically scale out/in pods in HPA policies to satisfy complex services.

HPA and CronHPA associate scaling objects using the **scaleTargetRef** field. If a Deployment is the scaling object for both CronHPA and HPA, the two scaling policies are independent of each other. The operation performed later overwrites the operation performed earlier. As a result, the scaling effect does not meet the expectation.



When CronHPA and HPA are used together, CronHPA rules take effect based on the HPA policy. CronHPA uses HPA to perform operations on the Deployment. Understanding the following parameters can better understand the working rules of the CronHPA.

- **targetReplicas**: Number of pods set for CronHPA. When CronHPA takes effect, this parameter adjusts the maximum or minimum number of pods in HPA policies to adjust the number of Deployment pods.
- **minReplicas**: Minimum number of Deployment pods.
- **maxReplicas**: Maximum number of Deployment pods.
- **replicas**: Number of pods in a Deployment before the CronHPA policy takes effect.

When the CronHPA rule takes effect, the maximum or minimum number of pods are adjusted by comparing the number of **targetReplicas** with the actual number of pods and combining the minimum or maximum number of pods in the HPA policy.

Figure 9-5 CronHPA scaling scenarios

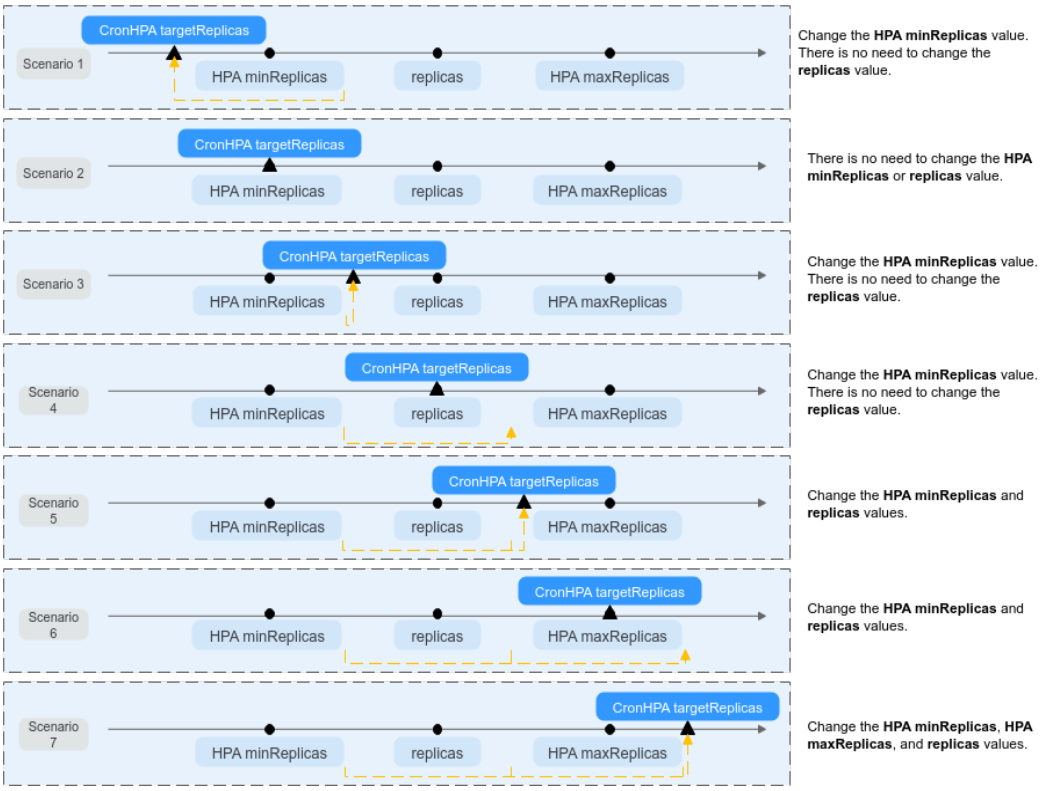


Figure 9-5 shows possible scaling scenarios. The following examples detail how CronHPA modifies the number of pods in HPAs.

Table 9-5 CronHPA scaling parameters

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
1	**targetReplicas** < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Deployment: 5	When the value of **targetReplicas** is smaller than that of **minReplicas**: • Change the value of **minReplicas**. • The value of **replicas** requires no change.
2	**targetReplicas** = minReplicas ≤ replicas ≤ maxReplicas	5	6	5/10	HPA: 5/10 Deployment: 6	When the value of **targetReplicas** is equal to that of **minReplicas**: • The value of **minReplicas** requires no change. • The value of **replicas** requires no change.

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
3	minReplicas < **targetReplicas** < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Deployment: 5	When the value of **targetReplicas** is greater than that of **minReplicas** and smaller than that of **replicates**: • Change the value of **minReplicas**. • The value of **replicas** requires no change.
4	minReplicas < **targetReplicas** = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Deployment: 5	When the value of **targetReplicas** is greater than that of **minReplicas** and equal to that of **replicates**: • Change the value of **minReplicas**. • The value of **replicas** requires no change.

Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
5	minReplicas ≤ replicas < **targetReplicas** < maxReplicas	6	5	1/10	HPA: 6/10 Deployment: 6	When the value of **targetReplicas** is greater than that of **replicates** and less than that of **maxReplicas**: • Change the value of **minReplicas**. • Change the value of **replicas**.
6	minReplicas ≤ replicas < **targetReplicas** = maxReplicas	10	5	1/10	HPA: 10/10 Deployment: 10	When the value of **targetReplicas** is greater than that of **replicates** and equal to that of **maxReplicas**: • Change the value of **minReplicas**. • Change the value of **replicas**.

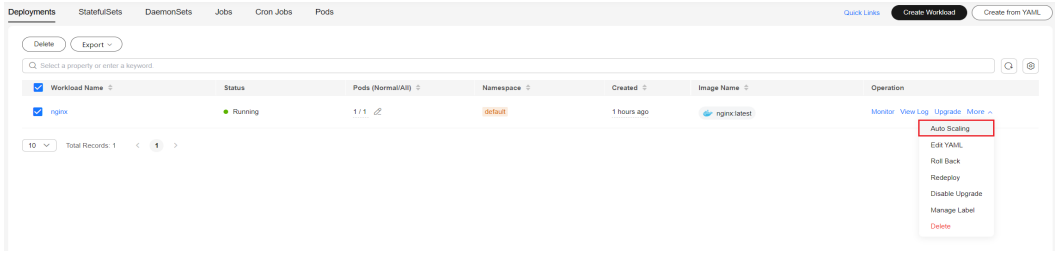
Scenario	Scenario Description	Scaling Condition			Result	Operation Description
		Target CronHPA Pods (targetReplicas)	Deployment Pods (replicas)	Upper and Lower Limits of HPA Pods (minReplicas / maxReplicas)		
7	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Deployment: 11	When the value of **targetReplicas** is greater than that of **maxReplicas**: • Change the value of **minReplicas**. • Change the value of **maxReplicas**. • Change the value of **replicas**.

Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **Auto Scaling** in the **Operation** column.

Figure 9-6 Scaling a workload



Step 3 Set **Policy Type** to **HPA+CronHPA** and enable HPA and CronHPA policies.

CronHPA periodically adjusts the maximum and minimum numbers of pods using the HPA policy.

Step 4 Configure the HPA policy. For details, see **Creating an HPA Policy**.

Figure 9-7 Enabling the HPA policy

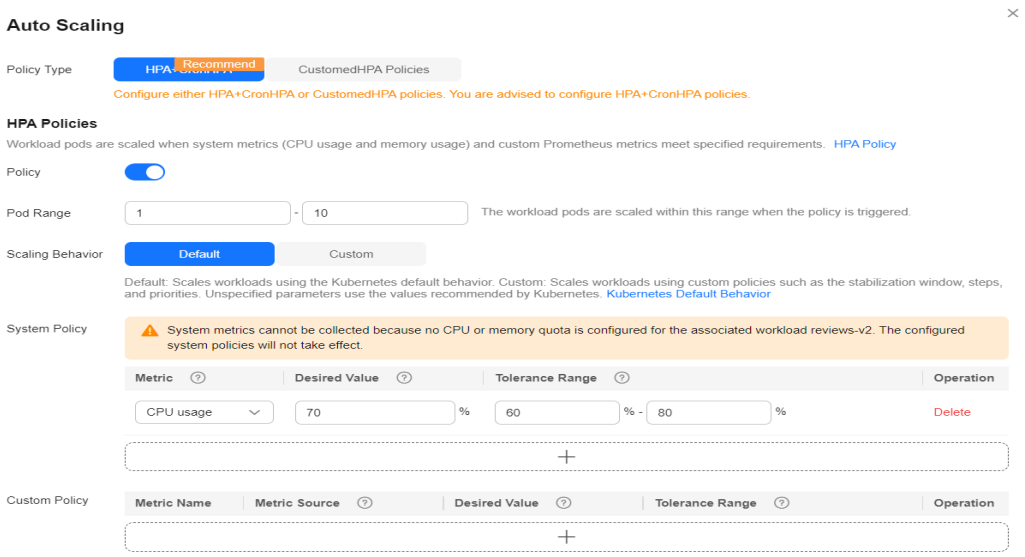


Table 9-6 HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range. **NOTICE** In CCE Turbo clusters, if you use a dedicated load balancer for your workload, the number of pods cannot exceed the backend server group quota of the load balancer, which is 500 by default. If you exceed this limit, you will not be able to add any more pods to the load balancer backend.
Cooldown Period	Interval between a scale-in and a scale-out. The unit is minute. **The interval cannot be shorter than 1 minute.** **This parameter is supported only in clusters of v1.15 to v1.23.** This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.

Parameter	Description
Scaling Behavior	**This parameter is supported only in clusters of v1.25 or later.** • **Default**: scales workloads using the Kubernetes default behavior. For details, see **Default Behavior**. • **Custom**: scales workloads using custom policies such as stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. – **Disable scale-out/scale-in**: Select whether to disable scale-out or scale-in. – **Stabilization Window**: a period during which CCE continuously checks whether the metrics used for scaling keep fluctuating. CCE triggers scaling if the desired state is not maintained for the entire window. This window restricts the unwanted flapping of pod count due to metric changes. – **Step**: specifies the scaling step. You can set the number or percentage of pods to be scaled in or out within a specified period. If there are multiple policies, you can select the policy that maximizes or minimizes the number of pods.
System Policy	• **Metric**: You can select **CPU usage** or **Memory usage**. **NOTE** Usage = Average resource usage of all pods in a workload/ Requested resources • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. Number of target pods (rounded up) = (Current metric value/Desired value) x Number of current pods **NOTE** When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes. • **Tolerance Range**: Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range. If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling is triggered. **This parameter is supported only in clusters of v1.15 or later.**

Parameter	Description
Custom Policy (supported only in clusters of v1.15 or later)	**NOTE** Before creating a custom policy, install an add-on that supports custom metric collection (for example, Prometheus) in the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**. • **Metric Name**: name of the custom metric. You can select a name as prompted. • **Metric Source**: Select an object type from the drop-down list. You can select **Pod**. • **Desired Value**: the average metric value of all pods. Number of pods to be scaled (rounded up) = (Current metric value/ Desired value) x Number of current pods **NOTE** When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes. • **Tolerance Range**: Scaling is not triggered when the metric value is within the tolerance range. The desired value must be within the tolerance range.

Step 5 Click + in the CronHPA policy rule. In the dialog box displayed, configure scaling policy parameters.

Figure 9-8 Enabling the CronHPA policy

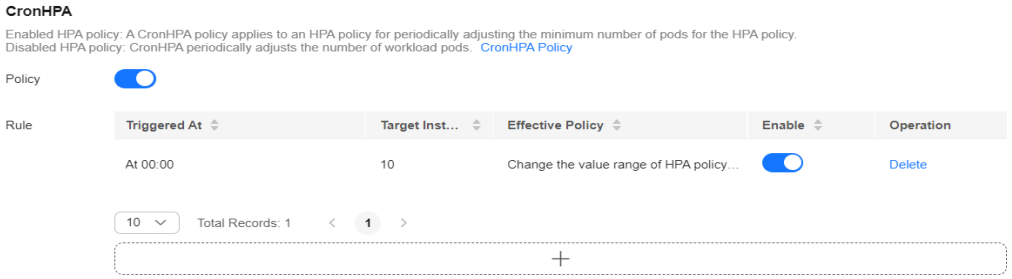


Table 9-7 CronHPA policy parameters

Parameter	Description
Target Instances	When the policy is triggered, CCE will adjust the number of HPA policy pods based on service requirements. For details, see **Table 9-5**.
Trigger Time	You can select a specific time every day, every week, every month, or every year. **NOTE** This time indicates the local time of where the node is deployed.
Enable	Enable or disable the policy rule.

Step 6 After configuring the preceding parameters, click **OK**. Then, the added policy rule is displayed in the rule list. Repeat the preceding steps to add multiple policy rules, but the triggering time of the policies must be different.

Step 7 Click **Create**.

----End

Using kubectl

When the CronHPA is compatible with the HPA policy, the **scaleTargetRef** field in CronHPA must be set to the HPA policy, and the **scaleTargetRef** field in the HPA policy must be set to Deployment. In this way, CronHPA adjusts the maximum and minimum numbers of pods in the HPA policy at a fixed time and the scheduled scaling is compatible with the auto scaling.

Step 1 Create an HPA policy for the Deployment.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-test
  namespace: default
spec:
   maxReplicas: 10              #  Maximum number of pods
  minReplicas: 5              #  Minimum number of pods
  scaleTargetRef:             #  Associate a Deployment.
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  targetCPUUtilizationPercentage: 50
```

Step 2 Create a CronHPA policy and associate it with the HPA policy created in **Step 1**.

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
   scaleTargetRef:                # Associate an HPA policy.
      apiVersion: autoscaling/v1
      kind: HorizontalPodAutoscaler
      name: hpa-test
   rules:
   - ruleName: "scale-down"
     schedule: "15 * * * *"        # Running time and period of a job. For details, see Cron, for example, 0 * * *
* or @hourly.
     targetReplicas: 1             # Number of target pods
     disable: false
   - ruleName: "scale-up"
     schedule: "13 * * * *"
     targetReplicas: 11
     disable: false
```

Table 9-8 Key fields of CronHPA

Field	Description
apiVersion	API version. The value is fixed at **autoscaling.cce.io/v2alpha1**.

Field	Description
kind	API type. The value is fixed at **CronHorizontalPodAutoscaler**.
metadata.name	Name of a CronHPA policy.
metadata.namespace	Namespace to which the CronHPA policy belongs.
spec.scaleTargetRef	Specifies the scaling object of CronHPA. The following fields can be configured: • **apiVersion**: API version of the CronHPA scaling object. • **kind**: API type of the CronHPA scaling object. • **name**: Name of the CronHPA scaling object. CronHPA supports HPA policies or Deployments. For details, see **Using CronHPA to Adjust the HPA Scaling Scope** or **Using CronHPA to Directly Adjust the Number of Deployment Pods**.
spec.rules	CronHPA policy rule. Multiple rules can be added. The following fields can be configured for each rule: • **ruleName**: CronHPA rule name, which must be unique. • **schedule**: Running time and period of a job. For details, see **Cron**, for example, 0 * * * * or @hourly. **NOTE** This time indicates the local time of where the node is deployed. • **targetReplicas**: indicates the number of pods to be scaled in or out. • **disable**: The value can be **true** or **false**. **false** indicates that the rule takes effect, and **true** indicates that the rule does not take effect.

----End

Using CronHPA to Directly Adjust the Number of Deployment Pods

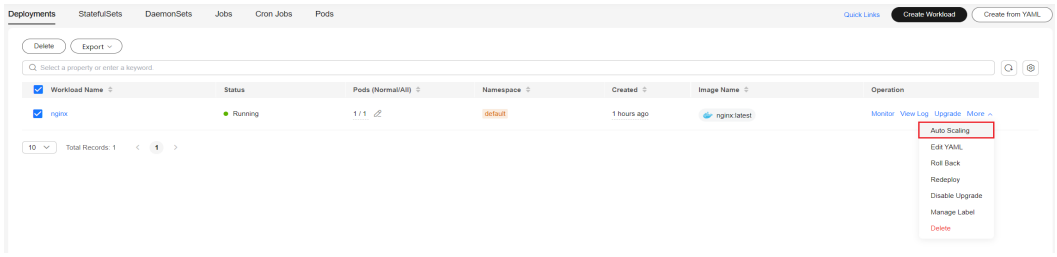
CronHPA adjusts associated Deployments separately to periodically adjust the number of Deployment pods. The method is as follows:

Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **Auto Scaling** in the **Operation** column.

Figure 9-9 Scaling a workload



Step 3 Set **Policy Type** to **HPA+CronHPA**, disable HPA, and enable CronHPA.

CronHPA periodically adjusts the number of workload pods.

Step 4 Click + in the CronHPA policy rule. In the dialog box displayed, configure scaling policy parameters.

Figure 9-10 Using CronHPA to adjust the number of workload pods

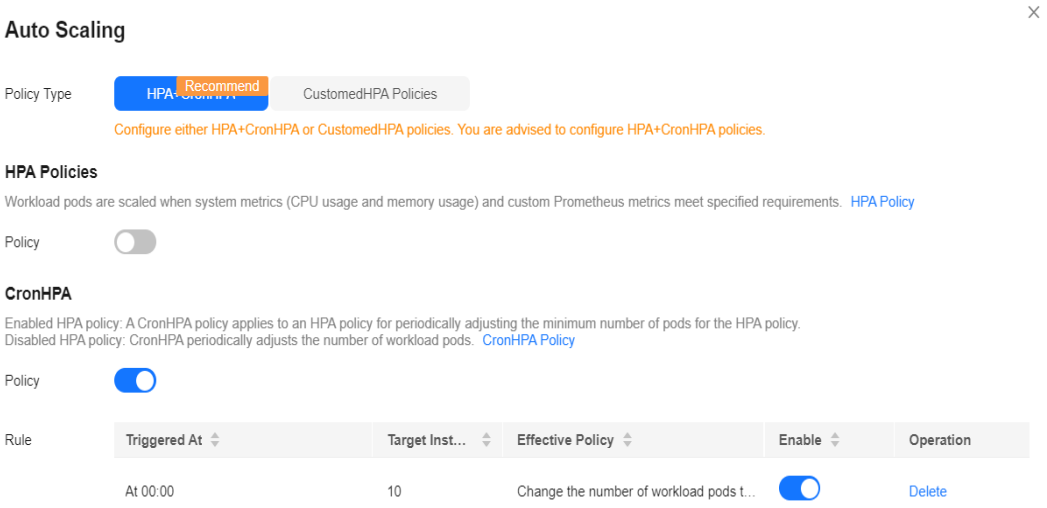


Table 9-9 CronHPA policy parameters

Parameter	Description
Target Instances	When a policy is triggered, the number of workload pods will be adjusted to the value of this parameter.
Trigger Time	You can select a specific time every day, every week, every month, or every year. **NOTE** This time indicates the local time of where the node is deployed.
Enable	Enable or disable the policy rule.

Step 5 After configuring the preceding parameters, click **OK**. Then, the added policy rule is displayed in the rule list. Repeat the preceding steps to add multiple policy rules, but the triggering time of the policies must be different.

Step 6 Click **Create**.

----End

Using kubectl

```
apiVersion: autoscaling.cce.io/v2alpha1
kind: CronHorizontalPodAutoscaler
metadata:
  name: ccetest
  namespace: default
spec:
  scaleTargetRef:            # Associate a Deployment.
     apiVersion: apps/v1
     kind: Deployment
     name: nginx
  rules:
  - ruleName: "scale-down"
    schedule: "08 * * * *"    # Running time and period of a job. For details, see Cron, for example, 0 * * * * or
@hourly.
    targetReplicas: 1
    disable: false
  - ruleName: "scale-up"
    schedule: "05 * * * *"
    targetReplicas: 3
    disable: false
```

9.2.5 Creating a CustomedHPA Policy

A CustomedHPA policy scales Deployments based on metrics (such as CPU usage and memory usage) or at a periodic interval (a specific time point every day, every week, every month, or every year). This type of policy is a CCE-enhanced auto scaling capability.

Supported functions:

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.
- Different scaling operations can be performed based on the actual metric values.

Prerequisites

The **CCE Advanced HPA** add-on must be installed. If the add-on version is earlier than 1.2.11, **Prometheus** must be installed. If the add-on version is 1.2.11 or later, one of the following add-ons that can provide metrics APIs must be installed based on your cluster version and service requirements:

- **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
- **Cloud Native Cluster Monitoring**: available only in clusters of v1.17 or later.
  - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**.
  - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see **Creating an HPA Policy Using Custom Metrics**.

- **Prometheus (EOM)**: Prometheus needs to be registered as a metrics API. For details, see **Providing Resource Metrics Through the Metrics API**. This add-on supports only clusters of v1.21 or earlier.

Notes and Constraints

- CustomedHPA policies apply only to clusters of v1.15 or later.
- For clusters earlier than v1.19.10, if an HPA policy is used to scale out a workload with EVS volumes mounted, the existing pods cannot be read or written when a new pod is scheduled to another node.

  For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.
- The specifications of the CCE Advanced HPA add-on are determined based on the total number of containers in the cluster and the number of scaling policies. Configure 500m CPU cores and 1000 MiB of memory for every 5000 containers, and 100m CPU cores and 500 MiB of memory for every 1000 scaling policies.
- After a CustomedHPA policy is created, the type of its associated workload cannot be changed.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **Auto Scaling** in the **Operation** column.

Step 3 Set **Policy Type** to **CustomedHPA** and configure policy parameters.

Table 9-10 CustomedHPA policy parameters

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range. **NOTICE** In CCE Turbo clusters, if you use a dedicated load balancer for your workload, the number of pods cannot exceed the backend server group quota of the load balancer, which is 500 by default. If you exceed this limit, you will not be able to add any more pods to the load balancer backend.
Cooldown Period	Enter an interval, in minutes. This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably. **NOTE** If you are using CCE Cluster Autoscaler version 1.3.10 or later, the cooldown period will only apply to metric policies. Periodic policies will not be affected by the cooldown period.

Parameter	Description
Rule	Click +. In the dialog box displayed, set the following parameters: • **Type**: You can select **Metric-based** (Table 9-11) or **Periodic** (Table 9-12). Then, configure trigger conditions and actions. • **Enable**: Enable or disable the policy rule. After configuring the preceding parameters, click **OK**. Then, the added policy rule is displayed in the rule list.

Table 9-11 Metric-based rules

Parameter	Description
Trigger	Select **CPU usage** or **Memory usage**, choose **>** or **<**, and enter a percentage. **NOTE** Usage = Average resource usage of all pods in a workload/Requested resources

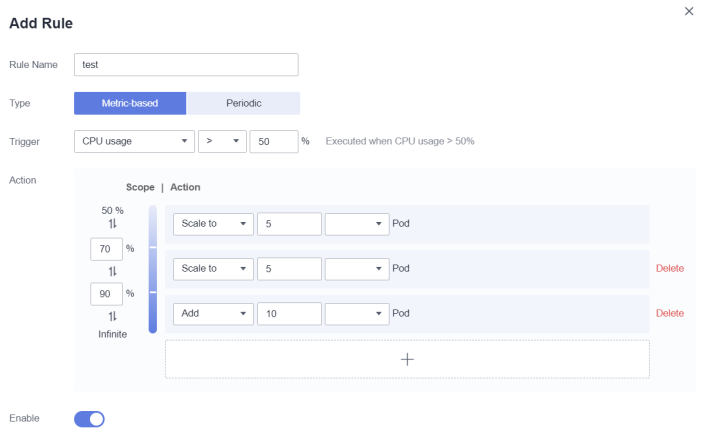
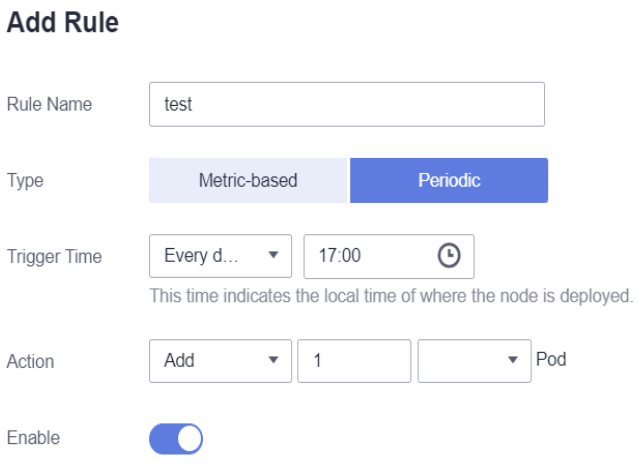
Parameter	Description
Action	Set an action to be performed when the trigger condition is met. Multiple actions can be added. • **Scale To**: Adjust the number of pods to the specified value. Both a number and a percentage will do. This action can be used to scale in or out pods. If the current number of pods is less than the target value or the target percentage is greater than 100%, the number of pods will be scaled out to the target value. If the current number of pods is greater than the target value or the target percentage is less than 100%, the number of pods will be scaled in to the target value. • **Add**: Configure this parameter when **Trigger** is set to **>**. Add the number of pods. You can specify a number or a percentage. This action can only be used to scale out pods. • **Reduce**: Configure this parameter when **Trigger** is set to **<**. Reduce the number of pods. You can specify a number or a percentage. This action can only be used to scale in pods. **NOTE** You can enter a number or a percentage for the preceding actions. When entering a percentage, you are required to specify the minimum number of available pods. Final number of pods = Number of current pods x Percentage. The result is rounded up. If the result is smaller than the minimum number of available pods, the preset value is used. Otherwise, the calculation result is used. As shown below, when the CPU usage exceeds 50%, the number of pods is scaled out to 5. When the CPU usage exceeds 70%, the number of pods is scaled out to 8. When the CPU usage exceeds 90%, the number of pods is scaled out to 18 (adding 10 more pods). These rules also work for scale-in operations. **Figure 9-11** Setting a trigger condition 

Table 9-12 Periodic-based rules

Parameter	Description
Trigger Time	You can select a specific time every day, every week, every month, or every year.
Action	Set an action to be performed at the **Triggered Time**. As shown below, one pod will be added at 17:00 every day. • **Scale To**: Adjust the number of pods to the specified value. Both a number and a percentage will do. This action can be used to scale in or out pods. If the current number of pods is less than the target value or the target percentage is greater than 100%, the number of pods will be scaled out to the target value. If the current number of pods is greater than the target value or the target percentage is less than 100%, the number of pods will be scaled in to the target value. • **Add**: Add the number of pods. You can specify a number or a percentage. This action can only be used to scale out pods. • **Reduce**: Reduce the number of pods. You can specify a number or a percentage. This action can only be used to scale in pods. **NOTE** You can enter a number or a percentage for the preceding actions. When entering a percentage, you are required to specify the minimum number of available pods. Final number of pods = Number of current pods x Percentage. The result is rounded up. If the result is smaller than the minimum number of available pods, the preset value is used. Otherwise, the calculation result is used. **Figure 9-12** Periodic triggering (Daily) 

Step 4 Click **Create**.

----End

Using kubectl

A CustomedHPA policy is a CustomResourceDefinition (CRD) and can be defined as follows in YAML:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: CustomedHorizontalPodAutoscaler
metadata:
  name: customhpa-example
  namespace: default
spec:
  coolDownTime: 3m            # Cooldown period
  maxReplicas: 10             # Maximum number of pods
  minReplicas: 1              # Minimum number of pods
  rules:
    - actions:                # Policy rules
        - metricRange: 0,0.1          # Metric range, from 0 to 10%
          operationType: ScaleDown    # Scaling type. ScaleDown indicates downsizing.
          operationUnit: Task         # Operation unit. Task indicates the number of tasks.
          operationValue: 1           # Resource quantity in each scaling
        - metricRange: 0.1,0.3        # Metric range, from 10% to 30%
          operationType: ScaleDown
          operationUnit: Task
          operationValue: 2
      disable: false
      metricTrigger:
        hitThreshold: 1
        metricName: CPURatioToRequest   # Metric name. CPURatioToRequest indicates the CPU usage.
        metricOperation: <              # Metric expression operator
        metricValue: 0.3                # Value on the right of the metric expression
        periodSeconds: 60               #
        statistic: instantaneous        #
      ruleName: low
      ruleType: Metric
    - actions:
        - metricRange: 0.7,0.9
          operationType: ScaleUp
          operationUnit: Task
          operationValue: 1
        - metricRange: 0.9,+Infinity
          operationType: ScaleUp
          operationUnit: Task
          operationValue: 2
      disable: false
      metricTrigger:
        hitThreshold: 1
        metricName: CPURatioToRequest
        metricOperation: '>'
        metricValue: 0.7
        periodSeconds: 60
        statistic: instantaneous
      ruleName: high
      ruleType: Metric
  scaleTargetRef:               # Associated workload
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
```

9.2.6 Creating a VPA Policy

Kubernetes Vertical Pod Autoscaler (VPA) scales pods vertically. It does this by analyzing the historical usage of container resources and automatically adjusting the CPU and memory resources requested by pods. VPA can adjust container resource requests within a specific range to match the service load. It increases the resource requests when the service load increases and reduces them when the service load decreases to conserve computing resources. Additionally, VPA can

provide recommendations on CPU and memory requests to optimize container resource utilization while ensuring that the containers have enough resources to function properly.

Overview

VPA collects and analyzes resource metrics for each container, adjusts the requested resources based on actual usage, and maintains the ratio of resource limit to request before and after the adjustment. VPA can increase or decrease CPU and memory resources as needed.

The rules are as follows:

- VPA generates the CPU and memory resource recommendations using the data collected by the Metrics API.
- VPA, in theory, recommends a minimum of 250 MiB of memory for each pod and 250 MiB divided by the number of containers in the pod for each container. It also recommends a minimum of 25m vCPUs for each pod and 25m divided by the number of containers in the pod for each container.

  When setting up a VPA, you can establish the minimum and maximum number of elastic resources in containers by configuring the **containerPolicies** field.
- If a container has both resource request and limit configured, VPA will provide resource recommendations. It will adjust the requested resources of the container to match the recommendations and generate recommended resource limit based on the ratio of the original resource request to the limit set during the container's initial creation.

  Assume that the requested vCPUs of a container are 100m and the limit is 200m (with a ratio of 1:2). If VPA recommends a requested vCPU of 80m, the container's vCPU limit will be 160m.
- VPA ensures its recommendations align with other resource limits. If the VPA recommendations conflict with a resource limit, they will not be adjusted to fit the limit. This means that the resource configuration suggested by VPA may go beyond other resource limits.

  Assume that the requested memory of a namespace cannot exceed 2 GiB. If VPA recommends a high memory configuration for a pod in that namespace, the total memory requested by the namespace may exceed 2 GiB after the pod's resource configuration is updated. This means the pod will not be scheduled.

Prerequisites

- The cluster version must be v1.25 or later.
- An add-on that provides Metrics API has been installed in the cluster. You can select one of the following add-ons based on your service requirements:
  - **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage.
  - **Cloud Native Cluster Monitoring**: provides basic resource usage metrics using Prometheus. You need to register Prometheus as a service of Metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**.

- The Vertical Pod Autoscaler (**Vertical Pod Autoscaler**) add-on has been installed in the cluster.

Precautions

NOTICE

The VPA feature is being tested. Exercise caution when using this feature.

- When VPA adjusts a pod's resource configuration dynamically, the pod will be recreated and could be scheduled to a different node. However, VPA cannot ensure that the scheduling will be successful.
- VPA can dynamically adjust the resource configuration of pods managed by replication controllers (such as Deployments and StatefulSets), but not for pods that are not managed by replication controllers.
- VPA and HPA that monitors CPU and memory metrics cannot operate simultaneously.
- VPA admission webhooks update the pod resource configuration. If there are other admission webhooks present in the cluster, make sure they do not conflict with the VPA admission webhooks.
- VPA handles the majority of OOM events, but it may not handle all of them.
- VPA performance has not yet been tested in large clusters.
- VPA may recommend more resources than what is currently available, such as exceeding the maximum limit or resource quotas of nodes. This can result in recreated pods being stuck in a pending state and unable to be scheduled.
- Configuring multiple VPAs for the same workload can lead to inconsistent behavior.

Procedure

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Deploy a sample workload. If a workload already runs in the cluster, skip this step.

```
kubectl create -f hamster.yaml
```

Example configuration of **hamster.yaml**:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hamster
spec:
  selector:
    matchLabels:
      app: hamster
  replicas: 2
  template:
    metadata:
      labels:
        app: hamster
    spec:
      containers:
        - name: hamster
          image: registry.k8s.io/ubuntu-slim:0.1
```



```
        resources:
          requests:
            cpu: 100m
            memory: 50Mi
        command: ["/bin/sh"]
        args:
          - "-c"
          - "while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done"
```

Step 3 Create a VPA.

```
kubectl create -f hamster-vpa.yaml
```

Example configuration of **hamster-vpa.yaml**:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: hamster-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind: Deployment
    name: hamster
  updatePolicy:
    updateMode: "Off"
  resourcePolicy:
    containerPolicies:
      - containerName: '*'
        minAllowed:
          cpu: 100m
          memory: 50Mi
        maxAllowed:
          cpu: 1
          memory: 500Mi
        controlledResources: ["cpu", "memory"]
```

Table 9-13 Key fields of a VPA

Field	Mandatory	Description
spec.targetRef	Yes	The target workloads of the VPA Workload types such as Deployment, StatefulSet, and DaemonSet are supported.
spec.updatePolicy.updateMode	No	The update policy of the VPA recommended resources. The default value is **Auto**. Options: • **Off**: The VPA only generates recommended resources and does not change the requested resources of pods. • **Recreate**: The VPA generates recommended resources and automatically changes the requested resources of pods. • **Initial**: The VPA generates recommended resources, changes the requested resources upon pod creation, but leaves the requested resources of existing pods unchanged. • **Auto**: The VPA behaves similarly to setting this parameter to **Recreate**.

Field	Mandatory	Description
spec.resourcePolicy.containerPolicies	No	Specified VPA policies for container resources, including the minimum and maximum number of resources allowed for containers For details, see **Table 9-14**.

Table 9-14 Key fields in containerPolicy

Field	Mandatory	Description
containerName	Yes	Container name
minAllowed	No	The minimum number of resources allowed for a container. The number of VPA recommended resources cannot be lower than the value of this parameter. Supported resources: • CPU • Memory
maxAllowed	No	The maximum number of resources allowed for a container. The number of VPA recommended resources cannot be higher than the value of this parameter. Supported resources: • CPU • Memory
controlledResources	No	The types of container resources controlled by the VPA The default value is **["cpu", "memory"]**. Supported resources: • CPU • Memory
mode	No	Whether the VPA policy of a container works. The default value is **Auto**. Options: • **Auto**: The VPA policy for the container is enabled. • **Off**: The VPA policy for the container is disabled.

Step 4 Wait for the VPA to generate the resource recommendations and run the following command to check the VPA recommendations:

```
kubectl get vpa hamster-vpa -oyaml
```

Command output:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: hamster-vpa
  namespace: default
spec:
  resourcePolicy:
    containerPolicies:
    - containerName: '*'
      controlledResources:
      - cpu
      - memory
      maxAllowed:
        cpu: 1
        memory: 500Mi
      minAllowed:
        cpu: 100m
        memory: 50Mi
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hamster
  updatePolicy:
    updateMode: "Off"
status:
  conditions:
  - lastTransitionTime: "2024-06-27T07:37:01Z"
    status: "True"
    type: RecommendationProvided
  recommendation:
    containerRecommendations:
    - containerName: hamster
      lowerBound:
        cpu: 475m
        memory: 262144k
      target:
        cpu: 587m
        memory: 262144k
      uncappedTarget:
        cpu: 587m
        memory: 262144k
      upperBound:
        cpu: 673m
        memory: 262144k
```

The **status.recommendation** field specifies the resource recommendations generated by the VPA.

If **updateMode** is set to **Auto**, the VPA will change the requested resources of the pod based on the recommended resources. This change will **trigger the rebuilding of the pod**.

Table 9-15 Key fields in containerRecommendation

Field	Description
containerName	Name of the container for which the VPA policy takes effect

Field	Description
target	Recommended resources generated by the VPA. The resources are calculated based on the minimum and maximum number of resources configured in the **containerPolicy** field. The VPA uses these values to dynamically adjust the number of pod resources that can be requested.
lowerBound	The minimum number of recommended resources
upperBound	The maximum number of recommended resources
uncappedTarget	Actual recommended resources generated by the VPA. The resources are not calculated based on the minimum and maximum number of resources configured in the **containerPolicy** field.

----End

9.2.7 Creating an AHPA Policy

The native Horizontal Pod Autoscaler (HPA) in Kubernetes can cause delays in scaling due to its passive triggering mechanism. To address this issue, the Advanced Horizontal Pod Autoscaler (AHPA) proactively identifies periods of pod scaling and predicts future fluctuations by analyzing historical service metrics.

Features

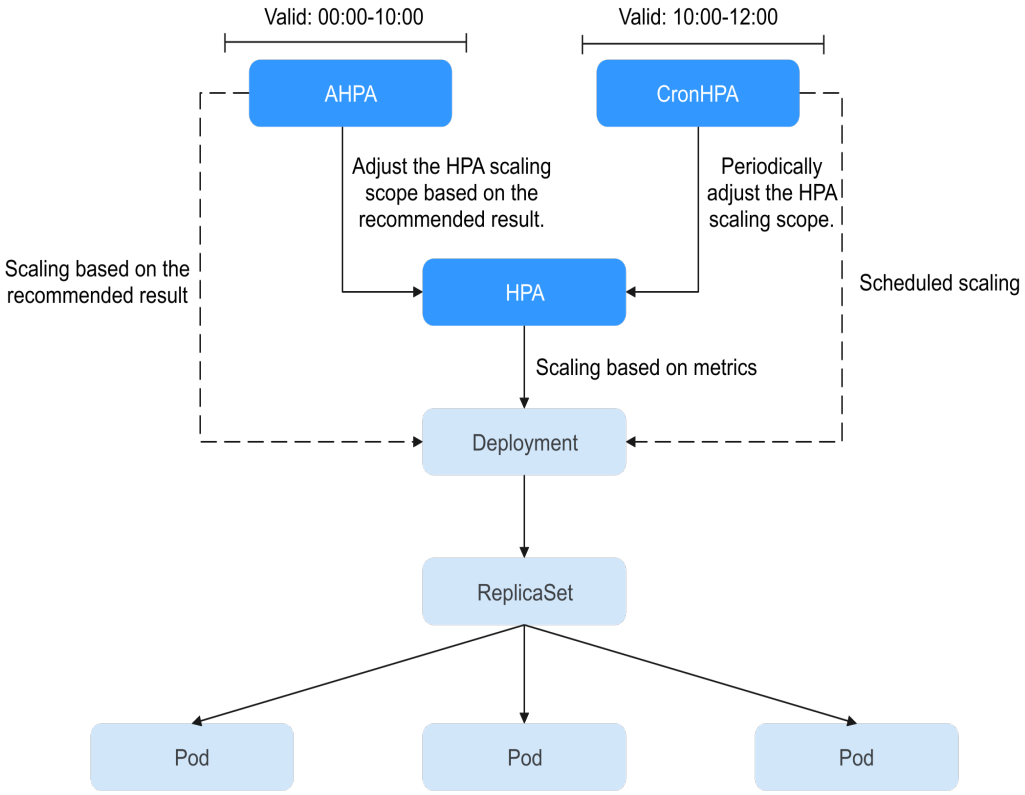
AHPA monitors historical workload metrics and performs weekly modeling, making it particularly effective for workloads with clear periodic patterns.

After AHPA is enabled, it collects monitoring data for a specific workload over a period of one to eight weeks. It then analyzes and models the data using statistical principles. Then, AHPA uses historical monitoring data and future service trends to recommend the optimal number of pods for a workload every minute. This allows pods to be prepared in advance to handle anticipated increases in service volume, ensuring adequate resource availability.

AHPA can work with HPA and CronHPA to scale pods in complex scenarios.

AHPA allows you to change the maximum and minimum number of pods in an HPA policy based on the recommended result, or directly adjust the number of pods in a Deployment.

AHPA and CronHPA share the same approach for adjusting the maximum and minimum pod numbers specified in an HPA policy. For details, see **Using CronHPA to Adjust the HPA Scaling Scope**.



Prerequisites

- The CCE Advanced HPA add-on of v1.5.2 or later has been installed in the cluster. For details, see **CCE Advanced HPA**.
- The Cloud Native Cluster Monitoring add-on has been installed in the cluster, and the **monitoring data is reported to AOM**. For details, see **Cloud Native Cluster Monitoring**.

Notes and Constraints

- AHPA policies apply only to clusters of v1.23 or later.
- For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volumes mounted, a new pod cannot be started because EVS disks cannot be attached.
- The specifications of the CCE Advanced HPA add-on are determined based on the total number of containers in the cluster and the number of scaling policies. Configure 500m CPU cores and 1000 MiB of memory for every 5000 containers, and 100m CPU cores and 500 MiB of memory for every 1000 scaling policies.
- AHPA needs extra memory as it analyzes and processes historical workload data. It is advised to allocate 100m CPU cores and 300 MiB of memory for every 100 AHPA policies.
- After an AHPA policy is created, the type of its associated workload cannot be changed.
- Either an AHPA policy or a CustomedHPA policy can be enabled.

Using AHPA

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Deploy a sample workload. If a workload already runs in the cluster, skip this step. It is advised to use a workload that has been monitored for over seven days, as AHPA requires a minimum of seven days of monitoring data.

```
kubectl create -f hamster.yaml
```

Example configuration of **hamster.yaml**:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hamster
spec:
  selector:
    matchLabels:
      app: hamster
  replicas: 2
  template:
    metadata:
      labels:
        app: hamster
    spec:
      containers:
        - name: hamster
          image: registry.k8s.io/ubuntu-slim:0.1
          resources:
            requests:
              cpu: 100m
              memory: 50Mi
          command: ["/bin/sh"]
          args:
            - "-c"
            - "while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done"
```

Step 3 Create an AHPA task.

```
kubectl create -f hamster-ahpa.yaml
```

Example configuration of **hamster-vpa.yaml**:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: AdvancedHorizontalPodAutoscaler
metadata:
  name: hamster-ahpa
  namespace: default
spec:
  scaleTargetRef:  # Associated workload, which can only be Deployment/HPA
    apiVersion: apps/v1
    kind: Deployment
    name: hamster
  minReplicas: 2  # Minimum number of pods
  maxReplicas: 10  # Maximum number of pods
  metrics:  # Metrics, whose format is the same as that of the community HPA
  - type: Resource  # Metric source type, which can only be Resource
    resource:
      name: cpu  # Metric source name, which can only be CPU or memory
      target:
        type: Utilization  # Metric source type, which can only be Utilization
        averageUtilization: 50
  predictConfig:
    predictWindowSeconds: 1800
    stabilizationWindowSeconds: 1800
    quantile: "0.97"
  effectiveTime:
  - '* * 11-22 ? * MON-FRI'  # Valid from 11:00 to 22:00, from Monday to Friday
```


Table 9-16 Key AHPA parameters

Parameter	Mandatory	Description
scaleTargetRef	Yes	Target Deployment/HPA.
metrics	Yes	Metrics for scaling, which can only be CPU or memory. Only one metric can be configured. Either CPU or memory can be configured.
maxReplicas	Yes	Maximum number of pods, which ranges from 0 to 2147483647. **NOTICE** In CCE Turbo clusters, if you use a dedicated load balancer for your workload, the number of pods cannot exceed the backend server group quota of the load balancer, which is 500 by default. If you exceed this limit, you will not be able to add any more pods to the load balancer backend.
minReplicas	Yes	Minimum number of pods, which ranges from 0 to 2147483647.
predictConfig.predictWindowSeconds	Yes	Recommendation time window, which starts from the current time. The historical monitoring data within this time window will be used to calculate the recommended number of pods. The value can range from 1 to 3600.
predictConfig.stabilizationWindowSeconds	No	Scale-in cooling duration. The value ranges from 0 to 3600.
predictConfig.quantile	Yes	Prediction quantile, which indicates the probability that a service metric's actual value will be lower than the preset target value. A higher value indicates a more conservative estimate. The value ranges from 0 to 1. Two decimal places are supported. The default value is 0.99. The recommended value ranges from 0.90 to 0.99.
effectiveTime	No	If multiple cron expressions are specified, AHPA will take effect on the combined set of these expressions. The default setting is always valid.

Step 4 After AOM has collected at least seven days of monitoring data for the target workload, AHPA can create a model and suggest the appropriate number of pods. Wait for the recommended number of pods to be provided and run the following command to check AHPA resource details:

```
kubectl get ahpa hamster-ahpa -oyaml
```

Command output:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: AdvancedHorizontalPodAutoscaler
```



```
metadata:
  creationTimestamp: "2024-10-07T13:11:58Z"
  generation: 2
  name: hamster-ahpa
  namespace: default
  resourceVersion: "15529454"
  uid: e5ffbb01-50b0-4485-8cf5-bc2be884b1ee
spec:
  effectiveTime:
  - '* * 11-22 ? * MON-FRI'
  maxReplicas: 10
  metrics:
  - resource:
      name: cpu
      target:
        averageUtilization: 50
        type: Utilization
    type: Resource
  minReplicas: 2
  predictConfig:
    predictWindowSeconds: 1800
    quantile: "0.97"
    stabilizationWindowSeconds: 1800
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hamster
status:
  conditions:
  - lastTransitionTime: "2024-10-07T13:24:19Z"
    message: the AHPA's model is ready
    reason: ModelIsReady
    status: "True"
    type: ModelAvailable
  - lastTransitionTime: "2024-10-07T13:24:19Z"
    message: the AHPA was able to successfully calculate a replica count
    reason: SucceededRunPrediction
    status: "True"
    type: ScalingActive
  - lastTransitionTime: "2024-10-07T13:24:19Z"
    message: ths ahpa checkpoint is fresh
    reason: CheckpointIsFresh
    status: "True"
    type: CheckpointAvailable
  - lastTransitionTime: "2024-10-07T13:24:19Z"
    message: recommended size matches current size
    reason: ReadyForNewScale
    status: "True"
    type: AbleToScale
  - lastTransitionTime: "2024-10-07T13:24:19Z"
    message: the desired replica count is more than the maximum replica count
    reason: TooManyReplicas
    status: "True"
    type: ScalingLimited
  currentReplicas: 10
  desiredReplicas: 10
  lastScaleTime: "2024-10-07T13:24:19Z"
```

Step 5 If you no longer need an AHPA policy, run the following command to delete it:

```
kubectl delete ahpa hamster-ahpa
```

During the validity period of AHPA, you can use custom ahpacheckpoint resources to keep the recommended settings for the next 6 hours. If you do not need this configuration, manually delete it.

```
kubectl delete ahpacheckpoint hamster-ahpa
```

----End

9.2.8 Managing Workload Scaling Policies

Scenario

After a workload scaling policy is created, you can update and delete the policy, as well as edit the YAML file.

Procedure

You can view the rules, latest status, and events of a workload scaling policy and handle exceptions based on the error information displayed.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Policies**. On the **Scaling Policies** tab page, click the **HPA Policies**/**CronHPA Policies**/**CustomedHPA Policies** tab based on the scaling policy type.

Step 3 Check the latest status, rules, and associated workloads of a scaling policy.

NOTE

You can also check a created scaling policy on the workload details page.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Workloads**. Click the workload name to view its details.
3. On the workload details page, switch to the **Auto Scaling** tab page to obtain the scaling policies. You can also obtain the scaling policies you configured on the **Policies** page.

Step 4 Manage scaling policies.

Scaling Policy Type	Operation
HPA	• **View Events**: Check HPA policy events. If an error occurred, locate and rectify the fault based on the error message displayed on the page. • **Edit YAML**: In the dialog box displayed, edit, copy, or download the YAML file. • **Edit**: On the **Edit HPA Policy** page, configure policy parameters listed in **Table 9-4**. • **Clone**: Duplicate an existing auto scaling policy and modify the parameter settings as required. • **Delete**: In the dialog box displayed, click **Yes**.
CronHPA	• **View YAML**: In the dialog box displayed, copy or download the YAML file but you are not allowed to modify it. • **Delete**: In the dialog box displayed, click **Yes**.

Scaling Policy Type	Operation
CustomedHPA	• **Edit**: On the **Edit HPA Policy** page, configure policy parameters listed in **Table 9-10**. • **Clone**: Duplicate an existing auto scaling policy and modify the parameter settings as required. • **View YAML**: In the dialog box displayed, copy or download the YAML file but you are not allowed to modify it. • **Delete**: In the dialog box displayed, click **Yes**.

----End

9.3 Scaling a Node

9.3.1 Node Scaling Rules

HPA is designed for pod-level scaling and can dynamically adjust the number of replicas based on workload metrics. However, if cluster resources are insufficient and new replicas cannot run, you can only scale out the cluster.

CCE Cluster Autoscaler is a node scaling component provided by Kubernetes. It automatically scales in or out nodes in a cluster based on the pod scheduling status and resource usage. It supports multiple scaling modes, such as multi-AZ, multi-pod-specifications, metric triggering, and periodic triggering, to meet the requirements of different node scaling scenarios.

Prerequisites

Before using the node scaling function, you must install the **CCE Cluster Autoscaler** add-on of v1.13.8 or later in the cluster.

How Cluster Autoscaler Works

Cluster Autoscaler goes through two processes.

- Scale-out: Autoscaler checks all unscheduled pods every 10 seconds and selects a node pool that meets the requirements for scale-out based on the policy you set.

  NOTE

  When Autoscaler checks unscheduled pods for scale outs, it uses the scheduling algorithm consistent with the Kubernetes community version for simulated scheduling calculation. If non-built-in kube-schedulers or other non-Kubernetes community scheduling policies are used for application scheduling, when Autoscaler is used to expand the capacity for such applications, the capacity may fail to be expanded or may be expanded more than expected due to inconsistent scheduling algorithms.

- Scale-in: Autoscaler scans all nodes every 10 seconds. If the number of pod requests on a node is less than the custom scale-in threshold (in percentage),

Autoscaler will check whether pods on the current node can be migrated to other nodes.

When a cluster node is idle for a period of time (10 minutes by default), cluster scale-in is triggered, and the node is automatically deleted. However, a node cannot be deleted from a cluster if the following pods exist:

- Pods that do not meet specific requirements set in Pod Disruption Budgets (**PodDisruptionBudget**)
- Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
- Pods that have the **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'** annotation
- Pods (except those created by DaemonSets in the kube-system namespace) that exist in the kube-system namespace on the node
- Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

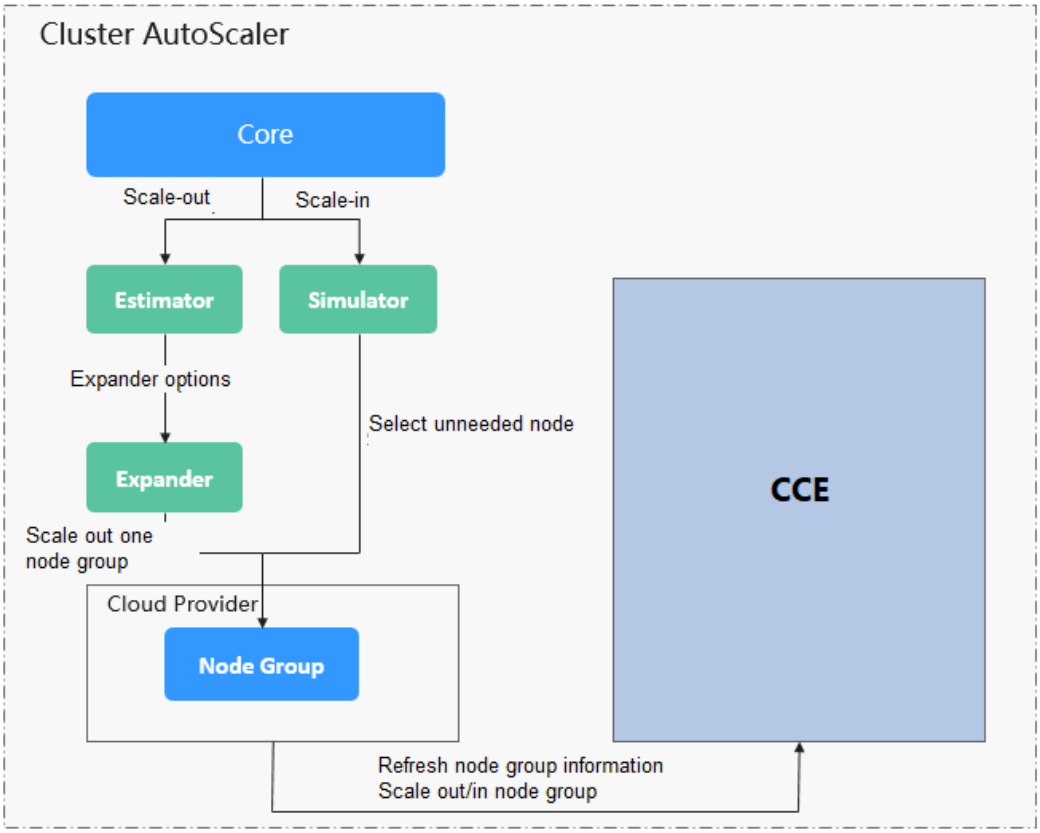
NOTE

When a node meets the scale-in conditions, Autoscaler adds the **DeletionCandidateOfClusterAutoscaler** taint to the node in advance to prevent pods from being scheduled to the node. After the Autoscaler add-on is uninstalled, if the taint still exists on the node, manually delete it.

Cluster Autoscaler Architecture

Figure 9-13 shows the Cluster Autoscaler architecture and its core modules.

Figure 9-13 Cluster Autoscaler architecture



Description

- **Estimator**: Evaluates the number of nodes to be added to each node pool to host unschedulable pods.
- **Simulator**: Finds the nodes that meet the scale-in conditions in the scale-in scenario.
- **Expander**: Selects an optimal node from the node pool picked out by the Estimator based on the user-defined policy in the scale-out scenario. **Table 9-17** lists the Expander policies.

Table 9-17 Expander policies supported by CCE

Policy	Description	Application Scenario	Example
Random	Randomly selects a schedulable node pool to perform the scale-out.	This policy is typically used as a basic backup for other complex policies. Only use this policy if the other policies cannot be used.	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2. 3. Autoscaler randomly selects node pool 1 or node pool 2 for scale-out.

Policy	Description	Application Scenario	Example
most-pods	A combined policy. It takes precedence over the random policy. Preferentially selects the node pool that can schedule the most pods after scale-out. If multiple node pools meet the condition, the random policy is used for further decision-making.	This policy is based on the maximum number of pods that can be scheduled.	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2. 3. Autoscaler evaluates that node pool 1 can schedule 20 new pods and node pool 2 can schedule only 10 new pods after scale-out. Therefore, Autoscaler selects node pool 1 for scale-out.

Policy	Description	Application Scenario	Example
least-waste	A combined policy. It takes precedence over the random policy. Autoscaler evaluates the overall CPU or memory allocation rate of the node pools and selects the node pool with the minimum CPU or memory waste. If multiple node pools meet the condition, the random policy is used for further decision-making.	This policy uses the minimum waste score of CPU or memory resources as the selection criteria. The formula for calculating the minimum waste score (wastedScore) is as follows: • wastedCPU = (Total number of CPUs of the nodes to be scaled out – Total number of CPUs of the pods to be scheduled)/Total number of CPUs of the nodes to be scaled out • wastedMemory = (Total memory size of nodes to be scaled out – Total memory size of pods to be scheduled)/Total memory size of nodes to be scaled out • wastedScore = wastedCPU + wastedMemory	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2. 3. Autoscaler evaluates that the minimum waste score of node pool 1 after scale-out is smaller than that of node pool 2. Therefore, Autoscaler selects node pool 1 for scale-out.

Policy	Description	Application Scenario	Example
priority	A combined policy. The priorities for the policies are as follows: priority > least-waste > topology-balance > random. It is an enhanced least-waste policy configured based on the node pool or scaling group priority. If multiple node pools meet the condition, the least-waste policy is used for further decision-making. Based on the least-waste policy, the topology-balance policy balances the number of nodes in different AZs. The AZ with fewer nodes is preferentially selected for a scale-	This policy allows you to configure the priorities of node pools or scaling groups through the console or API, while the least-waste policy can effectively minimize resource waste in various scenarios. The topology-balance policy ensures that the number of nodes in each AZ is balanced. The priority policy is used as the **default preferred policy** thanks to its good universality.	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2. 3. Autoscaler evaluates that node pool 1 has a higher priority than node pool 2. Therefore, Autoscaler selects node pool 1 for scale-out.

Policy	Description	Application Scenario	Example
	out to ensure that the number of nodes in each AZ is balanced. This policy is available in the add-on of version 1.23.122, 1.25.117, 1.27.85, 1.28.52, 1.29.14, or later.		

Policy	Description	Application Scenario	Example
priority-ratio	A combined policy. The priorities for the policies are as follows: priority > priority-ratio > least-waste > random. If there are multiple node pools with the same priority, evaluate the CPU to memory ratios for the nodes in the cluster. Then compare that ratio, for what was allocated to what had been requested. Finally, you should preferentially select the node pools where these two ratios are the closest.	This policy is used for rescheduling global resources for pods or nodes (instead of only adding nodes) to reduce the overall resource fragmentation rate of the cluster. Use this policy only in rescheduling scenarios.	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. Autoscaler simulates the scheduling phase and evaluates that some pending pods can be scheduled to the added nodes in both node pools 1 and 2. 3. Autoscaler determines a preferentially selected node pool and evaluates that the CPU/memory ratio of pods is 1:4. The node flavor in node pool 1 is 2 vCPUs and 8 GiB of memory (the CPU/memory ratio is 1:4), and the node flavor in node pool 2 is 2 vCPUs and 4 GiB of memory (the CPU/memory ratio is 1:2). Therefore, node pool 1 is preferred for this scale-out.

Policy	Description	Application Scenario	Example
topology-balance	A combined policy. The priorities for the policies are as follows: topology-balance > least-waste > random. Load is balanced based on the number of nodes in different AZs. The AZ with a small number of nodes is preferentially selected for a scale-out to ensure a balanced number of nodes in each AZ.	When the topology-balance policy is used, the node pool priority does not take effect. When a node scale-out is triggered, multiple nodes may be added at a time. This policy cannot evenly distribute these nodes to each AZ. It only ensures that the number of nodes in each AZ is balanced after multiple scale-outs.	Assume that auto scaling is enabled for node pools 1 and 2 in the cluster and the scale-out upper limit is not reached. The policy for scaling out the number of pods for a workload is as follows: 1. Pending pods trigger the Autoscaler to determine the scale-out process. 2. During the simulated scheduling, the Autoscaler checks the number of nodes in each AZ of the node pool with auto scaling enabled. 3. The Autoscaler determines a preferentially selected node pool and selects the AZ with the minimum number of nodes for the scale-out. If there are two AZs with the same number of nodes, the AZs are sorted using the next-priority policy (least-waste).

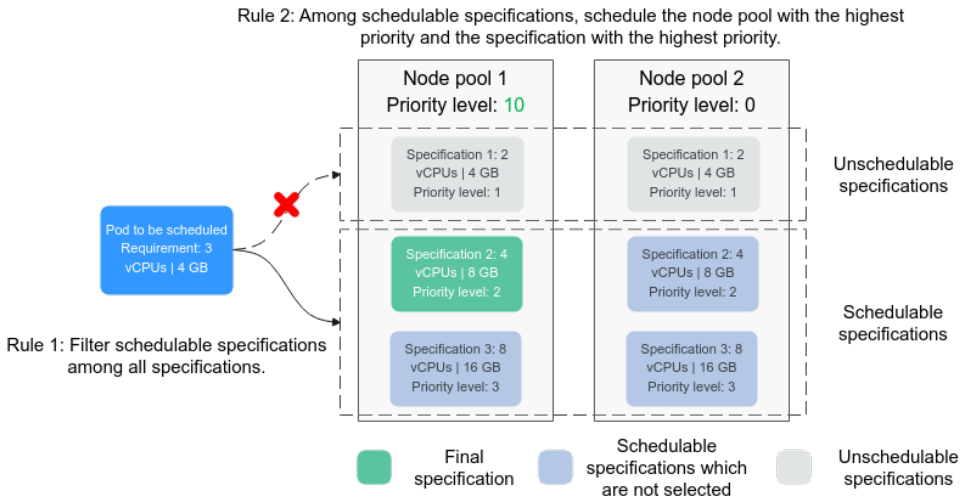
9.3.2 Priorities for Scaling Node Pools

Prerequisites

To use node flavor priorities, the Autoscaler version must be 1.19.35, 1.21.28, 1.23.30, 1.25.20, or later. To balance load among AZs, the version must be 1.23.122, 1.25.117, 1.27.85, 1.28.52, or later.

Elastic Capacity Expansion Policies

Node pools are scaled according to their priorities and flavor priorities.



1. Predictive flavor filtering:
   - The predictive algorithm chooses proper flavors that meet the scheduling needs of pending pods from all node pools.
   - When pods are scheduled, several factors are taken into account. These include checking if the node resources meet the requested resources of the pods, and verifying if the nodeSelector, nodeAffinity, and taints meet the conditions for pod scheduling.
   - If certain node pool flavors experience scale-out failures, for example, due to insufficient resources, they will enter a 5-minute cooldown period. During this time, the scale-out algorithm will automatically exclude them from being considered.
2. Node pool sorting by priority

   Node pools are assigned priorities and sorted accordingly. The node pool with the highest priority is preferentially selected.
3. Flavor selection by priority

   When multiple node pools have the same highest priority, the flavor with the highest priority is selected according to the following rules:
   - The flavor with the highest priority in each node pool is selected.
   - If multiple flavors have the same priority, choose the one that requires the least volume of resources to meet the pod scheduling requirements.
   - If multiple flavors require the minimum volume of resources, choose one based on a balanced distribution among AZs.
4. Troubleshooting if resources are insufficient or a creation failed

   If the preferred flavor is unavailable due to insufficient resources or quota in the AZ, CCE will try to use the next priority flavor in the node pool, and the original instance will enter a 5-minute cooldown period.

   If none of the flavors in a node pool can be used to create instances, CCE will try to use the next priority node pool to create instances.

Manual Capacity Expansion Policies

When manually scaling out a node pool, you can select a specified flavor. If the resources of the selected flavor are insufficient or the quota is insufficient, the scale-out will fail.

Configuring Priorities

For details about how to configure the priorities of node pool flavors, see **Configuring an Auto Scaling Policy for a Cluster**.

9.3.3 Creating a Node Scaling Policy

CCE provides auto scaling through the **CCE Cluster Autoscaler** add-on. Nodes with different flavors can be automatically added across AZs on demand.

If both a node scaling policy and the configuration in the Autoscaler add-on take effect, for example, there are pods that cannot be scheduled and a metric value exceeds the threshold, a scale-out will be performed first for the unschedulable pods.

- If the scale-out succeeds for the unschedulable pods, the system skips the metric-based rule logic and enters the next loop.
- If the scale-out fails for the unschedulable pods, the metric-based rule is executed.

Prerequisites

Before using the node scaling function, you must install the **CCE Cluster Autoscaler** add-on of v1.13.8 or later in the cluster.

To use node flavor priorities, the Autoscaler version must be 1.19.35, 1.21.28, 1.23.30, 1.25.20, or later. To balance load among AZs, the version must be 1.23.122, 1.25.117, 1.27.85, 1.28.52, or later.

Notes and Constraints

- If there are no nodes in a node pool, Autoscaler cannot obtain the CPU or memory data of the node, and the node scaling rule triggered using these metrics will not take effect.
- If the driver of a GPU or NPU node is not installed, Autoscaler will determine that the node is not fully available and the node scaling rules triggered using the CPU or memory metrics will not take effect.
- When CCE Cluster Autoscaler is used, some taints or annotations may affect auto scaling. Therefore, do not use the following taints or annotations in clusters:
  - **ignore-taint.cluster-autoscaler.kubernetes.io**: The taint works on nodes. Kubernetes-native Autoscaler supports protection against abnormal scale-outs and periodically evaluates the proportion of available nodes in the cluster. When the proportion of non-ready nodes exceeds 45%, protection will be triggered. In this case, all nodes with the **ignore-taint.cluster-autoscaler.kubernetes.io** taint in the cluster are filtered out from the Autoscaler template and recorded as non-ready nodes, which affect cluster scaling.

  - **cluster-autoscaler.kubernetes.io/enable-ds-eviction**: The annotation works on pods, which determines whether DaemonSet pods can be evicted by Autoscaler. For details, see **Well-Known Labels, Annotations and Taints**.

Configuring Node Pool Scaling Policies

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**. On the **Node Pools** tab, locate the row containing the target node pool and click **Auto Scaling**.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see **CCE Cluster Autoscaler**.
- If Autoscaler has been installed, directly configure scaling policies.

Step 3 Configure auto scaling policies.

AS Configuration

- **Customized Rule**: Click **Add Rule**. In the dialog box displayed, configure parameters. You can add multiple node scaling policies, a maximum of one CPU usage-based rule, and one memory usage-based rule. The total number of rules cannot exceed 10.

  The following table lists custom rules.

Table 9-18 Custom rules

Rule Type	Configuration
Metric-based	– **Trigger**: Select **CPU allocation rate** or **Memory allocation rate** and enter a value. The percentage must be greater than the value specified in the node resource requirements for a node scale-in when you configure a scaling policy (**Configuring an Auto Scaling Policy for a Cluster**). **NOTE** ▪ Resource allocation (%) = Resources requested by pods in the node pool/Resources allocatable to pods in the node pool ▪ **If multiple rules meet the conditions, the rules are executed in either of the following modes:** If rules based on the **CPU allocation rate** and **memory allocation rate** are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed. If a rule is configured based on the **CPU allocation rate** and **a periodic rule** and both the rules meet the scale-out conditions, the periodic rule executed early changes the node pool to the scaling state. As a result, the metric-based rule cannot be executed. After the periodic rule is executed and the node pool status becomes normal, the metric-based rule will not be executed. If the metric-based rule is executed early, the periodic rule will be executed after the metric-based rule is executed. ▪ If a rule is configured based on the **CPU allocation rate** and **memory allocation rate**, the policy detection period varies with the processing logic of each loop of the Autoscaler add-on. A scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cooldown period and node pool status. ▪ If the number of nodes reaches the upper limit of the cluster scale, **the upper limit of the nodes supported in a node pool**, or **the upper limit of the nodes of a specific flavor**, a metric-based scale-out will not be triggered. ▪ If the number of nodes, CPUs, or memory resources reaches **the upper limit for a node scale-out**, a metric-based scale-out will not be triggered. – **Action**: Configure an action to be performed when the triggering condition is met. ▪ **Custom**: Add a specified number of nodes to a node pool. ▪ **Auto calculation**: When the trigger condition is met, nodes are automatically added and the allocation rate is restored to a value lower than the threshold. The formula is as follows: Number of nodes to be added = [Resource request of pods in the node pool/(Available resources of a single node x Target allocation rate)] – Number of current nodes + 1

Rule Type	Configuration
Periodic	– **Trigger Time**: You can select a specific time every day, every week, every month, or every year. – **Action**: specifies an action to be carried out when the trigger time is reached. A specified number of nodes will be added to the node pool.

- **Nodes**: The number of nodes in a node pool will always be within the range during auto scaling.
- **Cooldown Period**: a period during which the nodes added in the current node pool cannot be scaled in.

AS Object

- **Specifications**: Configure whether to enable auto scaling for node flavors in a node pool.

  NOTE

  If multiple flavors are configured for a node pool, you can specify the upper limit for the number of nodes and the priority for each flavor separately.

Step 4 View cluster-level auto scaling configurations, which take effect for all node pools in the cluster. On this page, you can only view cluster-level auto scaling policies. To modify these policies, go to the **Settings** page. For details, see **Configuring an Auto Scaling Policy for a Cluster**.

Step 5 Click **OK**.

----End

Configuring an Auto Scaling Policy for a Cluster

NOTE

An auto scaling policy applies to all node pools in a cluster. After the policy is modified, the Autoscaler add-on will be restarted.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Settings** and click the **Auto Scaling** tab.

- If Autoscaler has not been installed, configure add-on parameters based on service requirements, click **Install**, and wait until the add-on is installed. For details about add-on configurations, see **CCE Cluster Autoscaler**.
- If Autoscaler has been installed, directly configure scaling policies.

Step 3 Configure for an elastic scale-out.

- **Auto Scale-out when the load cannot be scheduled**: When workload pods in a cluster cannot be scheduled (pods remain in pending state), CCE automatically adds nodes to the slave node pool. If a pod has been scheduled to a node, the node will not be involved in an automatic scale-out. Such auto scaling typically works with an HPA policy. For details, see **Using HPA and CA for Auto Scaling of Workloads and Nodes**.

If this function is not enabled, **custom scaling rules** are the only option for performing a scale-out.

- **Upper limit of resources to be expanded**: the upper limit for the cluster's resources, such as the number of nodes, CPU cores, and memory. Once this limit is reached, no new nodes will be automatically added.
- **Scale-Out Priority**: You can drag and drop the node pools in a list to adjust their scale-out priorities.

Step 4 Configure for an elastic scale-in. Elastic scale-in is disabled by default. After it is enabled, the following configurations are supported:

Node Scale-In Conditions: Nodes in a cluster are automatically scaled in when the scale-in conditions are met.

- **Node Resource Condition**: When the requested cluster node resources (both CPU and memory) are lower than a certain percentage (50% by default) for a period of time (10 minutes by default), a cluster scale-in is triggered.
- **Node Status Condition**: If a node is unavailable for a specified period of time, the node will be automatically reclaimed. The default value is 20 minutes.
- **Scale-in Exception Scenarios**: When a node meets the following exception scenarios, CCE will not scale in the node even if the node resources or status meets scale-in conditions:

  a. Resources on other nodes in the cluster are insufficient.

  b. Scale-in protection is enabled on the node. To enable or disable node scale-in protection, choose **Nodes** in the navigation pane and then click the **Nodes** tab. Locate the target node, choose **More**, and then enable or disable node scale-in protection in the **Operation** column.

  c. There is a pod with the non-scale label on the node.

  d. Policies such as reliability have been configured on some containers on the node.

  e. There are non-DaemonSet containers in the **kube-system** namespace on the node.

  f. (Optional) A container managed by a third-party pod controller is running on a node. Third-party pod controllers are for custom workloads except Kubernetes-native workloads such as Deployments and StatefulSets. Such controllers can be created using **CustomResourceDefinitions**.

Node Scale-in Policy

- **Number of Concurrent Scale-In Requests**: maximum number of idle nodes that can be concurrently deleted. Default value: 10.

  Only idle nodes can be concurrently scaled in. Nodes that are not idle can only be scaled in one by one.

  NOTE

  During a node scale-in, if the pods on the node do not need to be evicted (such as DaemonSet pods), the node is idle. Otherwise, the node is not idle.

- **Node Recheck Timeout**: interval for rechecking a node that cannot not be removed during a scale-in. Default value: 5 minutes.
- **Cooldown Time**

- **Scale-in Cooldown Time After Node Deletion**: cooldown period for starting scale-in evaluation again after an auto scale-in is triggered in a cluster. Default value: 10 minutes.
- **Scale-in Cooldown Time After Scale-out**: cooldown period for starting scale-in evaluation again after an auto scale-out is triggered in a cluster. Default value: 10 minutes.

  NOTE

  If both auto scale-out and scale-in exist in a cluster, set **Scale-in Cooldown Time After Scale-out** to 0 minutes. This prevents the node scale-in from being blocked due to continuous scale-out of some node pools or retries upon a scale-out failure, which results in unexpected waste of node resources.
- **Scale-in Cooldown Time After Failure**: cooldown period for starting scale-in evaluation again after an auto scale-in is failed in a cluster. Default value: 3 minutes. For details, see **Cooldown Period**.

Step 5 Click **Confirm configuration**.

----End

Cooldown Period

The impact and relationship between the two cooldown periods configured for a node pool are as follows:

Cooldown Period During a Scale-out

This interval indicates the period during which nodes added to the current node pool after a scale-out cannot be deleted. This setting takes effect in the entire node pool.

Cooldown Period During a Scale-in

The interval after a scale-out indicates the period during which the entire cluster cannot be scaled in after the Autoscaler add-on triggers a scale-out (due to the unschedulable pods, metrics, or scaling policies). This interval applies to the entire cluster.

The interval after a node is deleted indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers a scale-in. This setting applies to the entire cluster.

The interval after a failed scale-in indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers a scale-in. This setting applies to the entire cluster.

Period for Autoscaler to Retry a Scale-out

If a node pool failed to scale out, for example, due to insufficient resources or quota, or an error occurred during node installation, Autoscaler can retry the scale-out in the node pool or switch to another node pool. The retry period varies depending on failure causes:

- When resources in a node pool are sold out or the user quota is insufficient, Autoscaler cools down the node pool for 5 minutes, 10 minutes, or 20 minutes. The maximum cooldown duration is 30 minutes. Then, Autoscaler

switches to another node pool in the next 10 seconds for a scale-out until the expected nodes are added or all node pools have cooled down.

- If an error occurred during node installation in a node pool, the node pool enters a 5-minute cooldown period. After the period expires, Autoscaler can trigger a node pool scale-out again. If the faulty node is automatically reclaimed, Autoscaler re-evaluates the cluster status within 1 minute and triggers a node pool scale-out as needed.
- During a node pool scale-out, if a node remains in the installing state for a long time, Autoscaler tolerates the node for a maximum of 15 minutes. After the tolerance period expires, Autoscaler re-evaluates the cluster status and triggers a node pool scale-out as needed.

Example YAML

The following is a YAML example of a node scaling policy:

```
apiVersion: autoscaling.cce.io/v1alpha1
kind: HorizontalNodeAutoscaler
metadata:
  name: xxxx
  namespace: kube-system
spec:
  disable: false
  rules:
  - action:
      type: ScaleUp
      unit: Node
      value: 1
    cronTrigger:
      schedule: 47 20 * * *
    disable: false
    ruleName: cronrule
    type: Cron
  - action:
      type: ScaleUp
      unit: Node
      value: 2
    disable: false
    metricTrigger:
      metricName: Cpu
      metricOperation: '>'
      metricValue: "40"
      unit: Percent
    ruleName: metricrule
    type: Metric
  targetNodepoolIds:
  - 7d48eca7-3419-11ea-bc29-0255ac1001a8
```

Table 9-19 Key parameters

Parameter	Type	Description
spec.disable	Bool	Whether to enable the scaling policy. This parameter takes effect for all rules in the policy.
spec.rules	Array	All rules in a scaling policy.
spec.rules[x].ruleName	String	Rule name.

Parameter	Type	Description
spec.rules[x].type	String	Rule type. **Cron** and **Metric** are supported.
spec.rules[x].disable	Bool	Rule switch. Currently, only **false** is supported.
spec.rules[x].action.type	String	Rule action type. Currently, only **ScaleUp** is supported.
spec.rules[x].action.unit	String	Rule action unit. Currently, only **Node** is supported.
spec.rules[x].action.value	Integer	Rule action value.
spec.rules[x].cronTrigger	N/A	Optional. This parameter is valid only in periodic rules.
spec.rules[x].cronTrigger.schedule	String	Cron expression of a periodic rule.
spec.rules[x].metricTrigger	N/A	Optional. This parameter is valid only in metric-based rules.
spec.rules[x].metricTrigger.metricName	String	Metric of a metric-based rule. Currently, **Cpu** and **Memory** are supported.
spec.rules[x].metricTrigger.metricOperation	String	Comparison operator of a metric-based rule. Only **>** is supported.
spec.rules[x].metricTrigger.metricValue	String	Threshold of the metric rule. The value can be an integer ranging from 1 to 100 and must be a character. If the value is set to **-1**, the threshold is automatically calculated.
spec.rules[x].metricTrigger.Unit	String	Unit of the metric-based rule threshold. Only **%** is supported.
spec.targetNodepoolIds	Array	All node pools associated with the scaling policy.
spec.targetNodepoolIds[x]	String	UID of the node pool associated with the scaling policy.

9.3.4 Managing Node Scaling Policies

Scenario

After a node scaling policy is created, you can delete, edit, disable, enable, or clone the policy.

Viewing a Node Scaling Policy

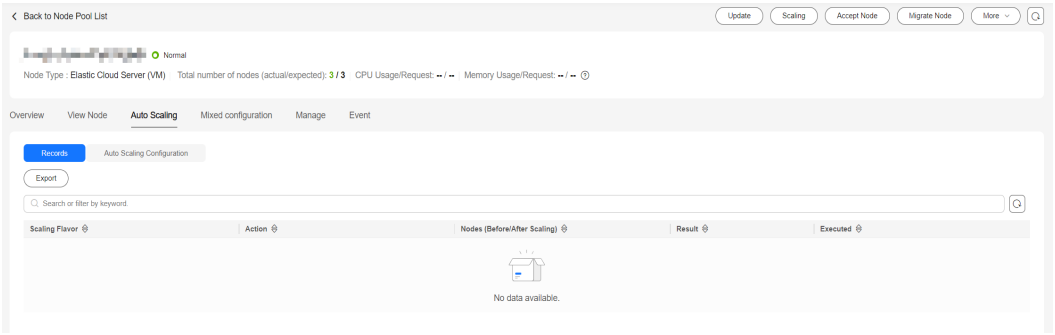
You can view the associated node pool, rules, and scaling history of a node scaling policy and rectify faults according to the error information displayed.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes**.On the page displayed, click the **Node Pools** tab and then the name of the node pool for which an auto scaling policy has been created to view the node pool details.

Step 3 On the node pool details page, click the **Auto Scaling** tab to view the auto scaling configuration and scaling records.

Figure 9-14 Checking auto scaling policies



NOTE

You can obtain created auto scaling policies on the **Policies** page.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab.
3. Check the configuration of the auto scaling policies. Choose **More** > **Scaling History** for the target policy to check the scaling records of the policy.

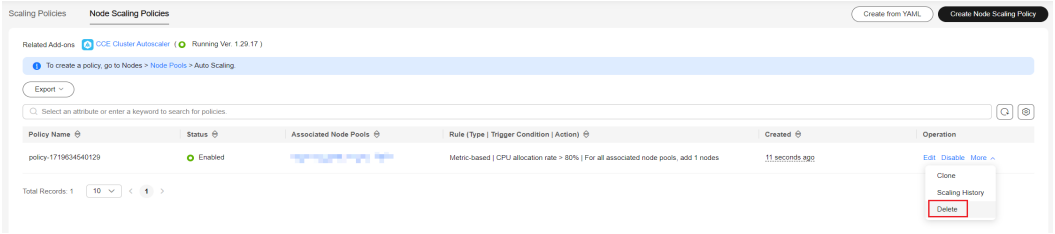
----End

Deleting a Node Scaling Policy

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and choose **More** > **Delete** in the **Operation** column.

Figure 9-15 Deleting a node scaling policy



Step 3 In the **Delete Node Scaling Policy** dialog box displayed, confirm whether to delete the policy.

Step 4 Click **Yes** to delete the policy.

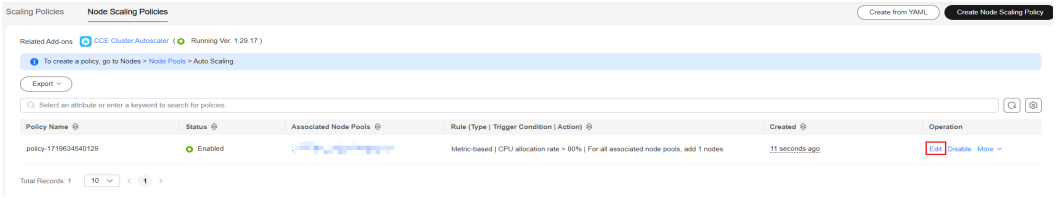
----End

Editing a Node Scaling Policy

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and click **Edit** in the **Operation** column.

Figure 9-16 Editing a node scaling policy



Step 3 On the **Edit Node Scaling Policy** page displayed, configure policy parameters listed in **Table 9-19**.

Step 4 After the configuration is complete, click **OK**.

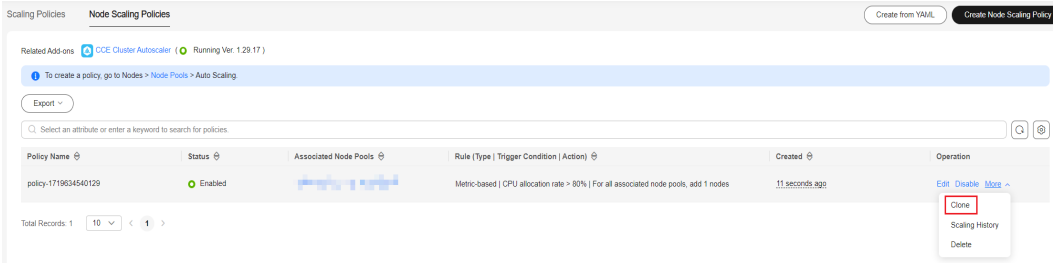
----End

Cloning a Node Scaling Policy

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy and choose **More** > **Clone** in the **Operation** column.

Figure 9-17 Cloning a node scaling policy



Step 3 On the **Create Node Scaling Policy** page displayed, certain parameters have been cloned. Add or modify other policy parameters based on service requirements.

Step 4 Click **OK**.

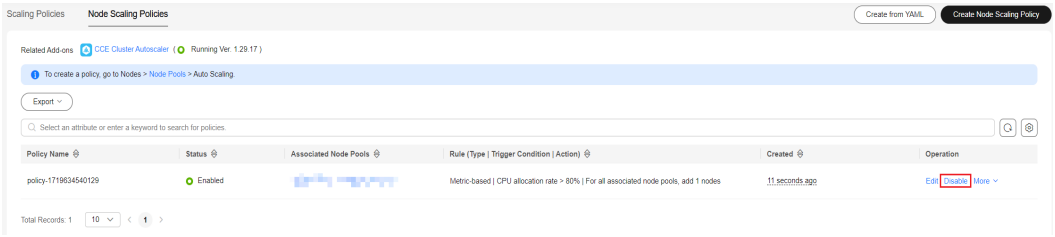
----End

Enabling or Disabling a Node Scaling Policy

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Policies**. On the page displayed, click the **Node Scaling Policies** tab, locate the row containing the target policy click **Disable** in the **Operation** column. If the policy is in the disabled state, click **Enable** in the **Operation** column.

Figure 9-18 Disabling a node scaling policy



Step 3 In the dialog box displayed, confirm whether to disable or enable the node policy.

----End

9.4 Using HPA and CA for Auto Scaling of Workloads and Nodes

Application Scenarios

The best way to handle surging traffic is to automatically adjust the number of machines based on the traffic volume or resource usage, which is called scaling.

When deploying applications in pods, you can configure requested resources and resource limits for the pods to prevent unlimited usage of resources during peak hours. However, after the upper limit is reached, an application error may occur. Pod scaling can effectively resolve this issue. If the resource usage on the node increases to a certain extent, newly added pods cannot be scheduled to this node. In this case, CCE will add nodes accordingly.

Solution

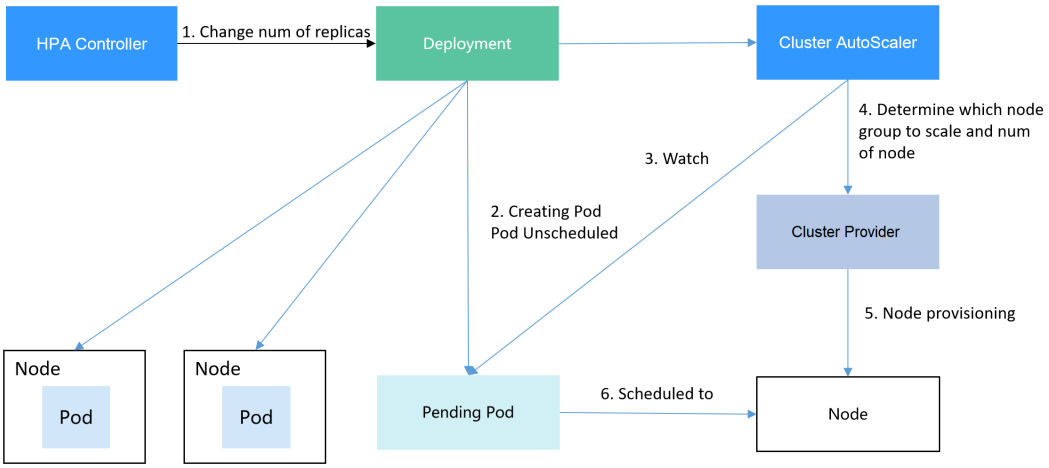
Two major auto scaling policies are HPA (Horizontal Pod Autoscaling) and CA (Cluster AutoScaling). HPA is for workload auto scaling and CA is for node auto scaling.

HPA and CA work with each other. HPA requires sufficient cluster resources for successful scaling. When the cluster resources are insufficient, CA is needed to add nodes. If HPA reduces workloads, the cluster will have a large number of idle resources. In this case, CA needs to release nodes to avoid resource waste.

As shown in **Figure 9-19**, HPA performs scale-out based on the monitoring metrics. When cluster resources are insufficient, newly created pods are in Pending state. CA then checks these pending pods and selects the most appropriate node pool based on the configured scaling policy to scale out the node pool. For details

about how HPA and CA work, see **Workload Scaling Mechanisms** and **Node Scaling Mechanisms**.

Figure 9-19 HPA and CA working flows

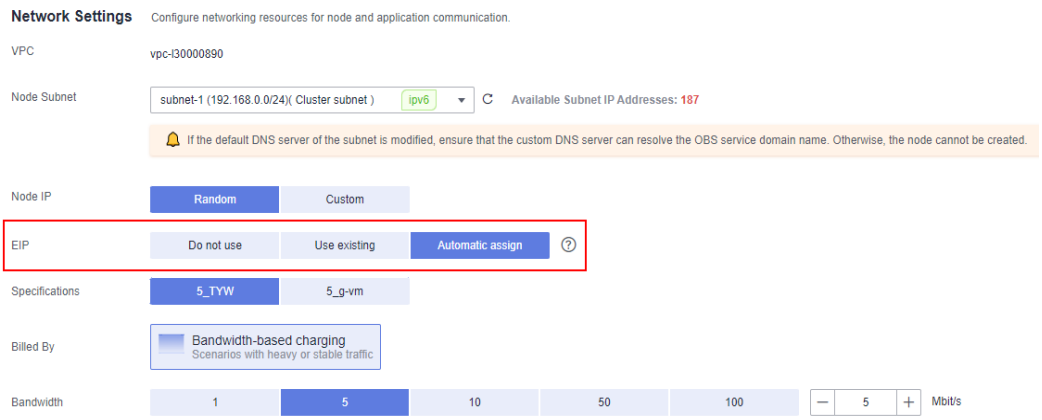


Using HPA and CA enables automatic scaling for most scenarios while also providing monitoring capabilities.

This section uses an example to describe the auto scaling process using HPA and CA policies together.

Preparations

Step 1 Create a cluster with one node. The node should have 2 cores of vCPUs and 4 GiB of memory, or a higher specification, as well as an EIP to allow external access. If no EIP is bound to the node during node creation, you can manually bind one on the ECS console after creating the node.



Step 2 Install add-ons for the cluster.

- autoscaler: node scaling add-on
- metrics-server: an aggregator of resource usage data in a Kubernetes cluster. It can collect measurement data of major Kubernetes resources, such as pods, nodes, containers, and Services.

Step 3 Log in to the cluster node and run a computing-intensive application. When a user sends a request, the result needs to be calculated before being returned to the user.

1. Create a PHP file named **index.php** to calculate the square root of the request for 1,000,000 times before returning **OK!**.

   ```
   vi index.php
   ```

   The file content is as follows:

   ```
   <?php
    $x = 0.0001;
    for ($i = 0; $i <= 1000000; $i++) {
      $x += sqrt($x);
    }
    echo "OK!";
   ?>
   ```

2. Compile a **Dockerfile** file to build an image.

   ```
   vi Dockerfile
   ```

   The content is as follows:

   ```
   FROM php:5-apache
   COPY index.php /var/www/html/index.php
   RUN chmod a+rx index.php
   ```

3. Run the following command to build an image named **hpa-example** with the tag **latest**.

   ```
   docker build -t hpa-example:latest .
   ```

4. (Optional) Log in to the SWR console, choose **Organizations** in the navigation pane, and click **Create Organization** in the upper right corner.

   Skip this step if you already have an organization.

5. In the navigation pane, choose **My Images** and then click **Upload Through Client**. On the page displayed, click **Generate a temporary login command** and click  to copy the command.

6. Run the login command copied in the previous step on the cluster node. If the login is successful, the message "Login Succeeded" is displayed.

7. Tag the hpa-example image.

   docker tag *{Image name 1:Tag 1}|{Image repository address}|{Organization name}|{Image name 2:Tag 2}*

   - *{Image name 1:Tag 1}*: name and tag of the local image to be uploaded.
   - *{Image repository address}*: the domain name at the end of the login command in **login command**. It can be obtained on the SWR console.
   - *{Organization name}*: name of the **created organization**.
   - *{Image name 2:Tag 2}*: desired image name and tag to be displayed on the SWR console.

   The following is an example:

   docker tag hpa-example:latest swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest

8. Push the image to the image repository.

   docker push *{Image repository address}|{Organization name}|{Image name 2:Tag 2}*

   The following is an example:

   docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest

The following information will be returned upon a successful push:

```
6d6b9812c8ae: Pushed
...
fe4c16cbf7a4: Pushed
latest: digest: sha256:eb7e3bbd*** size: **
```

To view the pushed image, go to the SWR console and refresh the **My Images** page.

----End

Creating a Node Pool and a Node Scaling Policy

Step 1 Log in to the CCE console, access the created cluster, click **Nodes** on the left, click the **Node Pools** tab, and click **Create Node Pool** in the upper right corner.

Step 2 Configure the node pool.

- **Node Type**: Select a node type.
- **Specifications**: 2 vCPUs | 4 GiB

Retain the defaults for other parameters. For details, see **Creating a Node Pool**.

Step 3 Locate the row containing the newly created node pool and click **Auto Scaling** in the upper right corner. For details, see **Creating a Node Scaling Policy**.

If the CCE Cluster Autoscaler add-on is not installed in the cluster, install it first. For details, see **CCE Cluster Autoscaler**.

- **Customize scale-out rules.**: Click **Add Rule**. In the dialog box displayed, configure parameters. If the CPU allocation rate is greater than 70%, a node is added to each associated node pool. A node scaling policy needs to be associated with a node pool. Multiple node pools can be associated. When you need to scale nodes, node with proper specifications will be added or reduced from the node pool based on the minimum waste principle.
- **Nodes**: Modify the node quantity range. The number of nodes in a node pool will always be within the range during auto scaling.
- **Cooldown Period**: a period during which the nodes added in the current node pool cannot be scaled in

Step 4 Click **OK**.

----End

Creating a Workload

Use the hpa-example image to create a Deployment with one replica. The image path is related to the organization uploaded to the SWR repository and needs to be replaced with the actual value.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hpa-example
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hpa-example
  template:
```



```
  metadata:
    labels:
      app: hpa-example
  spec:
    containers:
    - name: container-1
      image: 'hpa-example:latest' # Replace it with the address of the image you uploaded to SWR.
      resources:
        limits:                 # The value of limits must be the same as that of requests to prevent flapping during scaling.
          cpu: 500m
          memory: 200Mi
        requests:
          cpu: 500m
          memory: 200Mi
    imagePullSecrets:
    - name: default-secret
```

Then, create a NodePort Service for the workload so that the workload can be accessed from external networks.

NOTE

To allow external access to NodePort Services, allocate an EIP for the node in the cluster. After the allocation, synchronize node data. For details, see **Synchronizing Data with Cloud Servers**. If the node has already bound with an EIP, you do not need to create one.

Alternatively, you can create a Service with an ELB load balancer for external access. For details, see **Using kubectl to Create a Service (Automatically Creating a Load Balancer)**.

```
kind: Service
apiVersion: v1
metadata:
  name: hpa-example
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 31144
  selector:
    app: hpa-example
  type: NodePort
```

Creating an HPA Policy

Create an HPA policy. As shown below, the policy is associated with the hpa-example workload, and the target CPU usage is 50%.

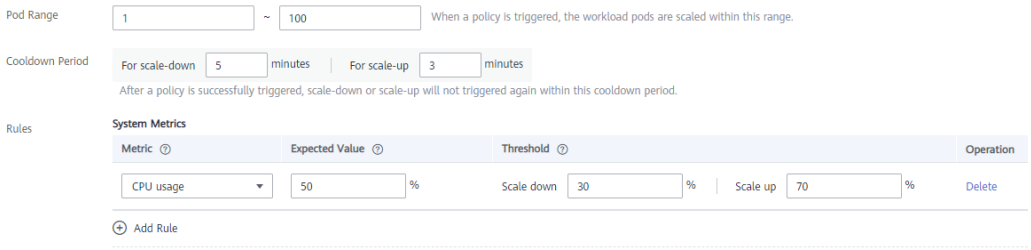
There are two other annotations. One annotation defines the CPU thresholds, indicating that scaling is not performed when the CPU usage is between 30% and 70% to prevent impact caused by slight fluctuation. The other is the scaling time window, indicating that after the policy is successfully executed, a scaling operation will not be triggered again in this cooling interval to prevent impact caused by short-term fluctuation.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-policy
  annotations:
    extendedhpa.metrics: '[{"type":"Resource","name":"cpu","targetType":"Utilization","targetRange":{"low":"30","high":"70"}}]'
    extendedhpa.option: '{"downscaleWindow":"5m","upscaleWindow":"3m"}'
spec:
```



```
scaleTargetRef:
  kind: Deployment
  name: hpa-example
  apiVersion: apps/v1
minReplicas: 1
maxReplicas: 100
metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Configure the parameters as follows if you are using the console.



Observing the Auto Scaling Process

Step 1 Check the cluster node status. In the following example, there are two nodes.

```
# kubectl get node
NAME          STATUS   ROLES    AGE     VERSION
192.168.0.183   Ready    <none>   2m20s   v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.26    Ready    <none>   55m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
```

Check the HPA policy. The CPU usage of the target workload is 0%.

```
# kubectl get hpa hpa-policy
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
hpa-policy   Deployment/hpa-example   0%/50%    1         100       1          4m
```

Step 2 Run the following command to access the workload. In the following command, {ip:port} indicates the access address of the workload, which can be queried on the workload details page.

while true;do wget -q -O- http://*{ip:port}*; done

NOTE

If no EIP is displayed, the cluster node has not been assigned any EIP. Allocate one, bind it to the node, and synchronize node data. For details, see **Synchronizing Data with Cloud Servers**.

Observe the scaling process of the workload.

```
# kubectl get hpa hpa-policy --watch
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-policy   Deployment/hpa-example   0%/50%     1         100       1          4m
hpa-policy   Deployment/hpa-example   190%/50%   1         100       1          4m23s
hpa-policy   Deployment/hpa-example   190%/50%   1         100       4          4m31s
hpa-policy   Deployment/hpa-example   200%/50%   1         100       4          5m16s
hpa-policy   Deployment/hpa-example   200%/50%   1         100       4          6m16s
hpa-policy   Deployment/hpa-example   85%/50%    1         100       4          7m16s
hpa-policy   Deployment/hpa-example   81%/50%    1         100       4          8m16s
```



```
hpa-policy   Deployment/hpa-example   81%/50%   1   100   7   8m31s
hpa-policy   Deployment/hpa-example   57%/50%   1   100   7   9m16s
hpa-policy   Deployment/hpa-example   51%/50%   1   100   7   10m
hpa-policy   Deployment/hpa-example   58%/50%   1   100   7   11m
```

You can see that the CPU usage of the workload is 190% at 4m23s, which exceeds the target value. In this case, scaling is triggered to expand the workload to four replicas/pods. In the subsequent several minutes, the CPU usage does not decrease until 7m16s. This is because the new pods may not be successfully created. The possible cause is that resources are insufficient and the pods are in the pending state. During this period, nodes are being scaled out.

At 7m16s, the CPU usage decreases, indicating that the pods are successfully created and start to bear traffic. The CPU usage decreases to 81% at 8m, still greater than the target value (50%) and the high threshold (70%). Therefore, 7 pods are added at 9m16s, and the CPU usage decreases to 51%, which is within the range of 30% to 70%. From then on, the number of pods remains 7.

In the following output, you can see the workload scaling process and the time when the HPA policy takes effect.

```
# kubectl describe deploy hpa-example
...
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  25m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 1
  Normal  ScalingReplicaSet  20m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 4
  Normal  ScalingReplicaSet  16m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 7
# kubectl describe hpa hpa-policy
...
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  20m    horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization
(percentage of request) above target
  Normal  SuccessfulRescale  16m    horizontal-pod-autoscaler  New size: 7; reason: cpu resource utilization
(percentage of request) above target
```

Check the number of nodes. The following output shows that two nodes are added.

```
# kubectl get node
NAME           STATUS   ROLES    AGE     VERSION
192.168.0.120  Ready    <none>   3m5s    v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.136  Ready    <none>   6m58s   v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.183  Ready    <none>   18m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
192.168.0.26   Ready    <none>   71m     v1.17.9-r0-CCE21.1.1.3.B001-17.36.8
```

You can also view the scaling history on the console. For example, the CA policy is executed once when the CPU allocation rate in the cluster is greater than 70%, and the number of nodes in the node pool is increased from 2 to 3. The new node is automatically added by autoscaler based on the pending state of pods in the initial phase of HPA.

The node scaling process is as follows:

1. After the number of pods changes to 4, the pods are in Pending state due to insufficient resources. As a result, the default scale-out policy of the autoscaler add-on is triggered, and the number of nodes is increased by one.

2. The second node scale-out is triggered because the CPU allocation rate in the cluster is greater than 70%. As a result, the number of nodes is increased by one, which is recorded in the scaling history on the console. Scaling based on the allocation rate ensures that the cluster has sufficient resources.

Step 3 Stop accessing the workload and check the number of pods.

```
# kubectl get hpa hpa-policy --watch
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-policy   Deployment/hpa-example   50%/50%    1         100       7          12m
hpa-policy   Deployment/hpa-example   21%/50%    1         100       7          13m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       7          14m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       7          18m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          18m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          19m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          23m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       3          23m
hpa-policy   Deployment/hpa-example   0%/50%     1         100       1          23m
```

You can see that the CPU usage is 21% at 13m. The number of pods is reduced to 3 at 18m, and then reduced to 1 at 23m.

In the following output, you can see the workload scaling process and the time when the HPA policy takes effect.

```
# kubectl describe deploy hpa-example
...
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  25m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 1
  Normal  ScalingReplicaSet  20m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 4
  Normal  ScalingReplicaSet  16m    deployment-controller  Scaled up replica set hpa-example-79dd795485
to 7
  Normal  ScalingReplicaSet  6m28s  deployment-controller  Scaled down replica set hpa-
example-79dd795485 to 3
  Normal  ScalingReplicaSet  72s    deployment-controller  Scaled down replica set hpa-
example-79dd795485 to 1
# kubectl describe hpa hpa-policy
...
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  20m    horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization
(percentage of request) above target
  Normal  SuccessfulRescale  16m    horizontal-pod-autoscaler  New size: 7; reason: cpu resource utilization
(percentage of request) above target
  Normal  SuccessfulRescale  6m45s  horizontal-pod-autoscaler  New size: 3; reason: All metrics below target
  Normal  SuccessfulRescale  90s    horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
```

You can also view the HPA policy execution history on the console. Wait until the one node is reduced.

The reason why the other two nodes in the node pool are not reduced is that they both have pods in the kube-system namespace (and these pods are not created by DaemonSets). For details, see **Node Scaling Mechanisms**.

----End

Summary

By using HPA and CA, auto scaling can be effortlessly implemented in various scenarios. Additionally, the scaling process of nodes and pods can be conveniently tracked.

9.5 Elastic Scaling of CCE Pods to CCI

The bursting add-on functions as a virtual kubelet to connect Kubernetes clusters to APIs of other platforms. This add-on is mainly used to extend Kubernetes APIs to serverless container services such as Huawei Cloud CCI.

With this add-on, you can schedule Deployments, StatefulSets, jobs, and CronJobs running in CCE clusters to **CCI** during peak hours. In this way, you can reduce consumption caused by cluster scaling.

Installing the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Install**.
5. Configure the add-on parameters.

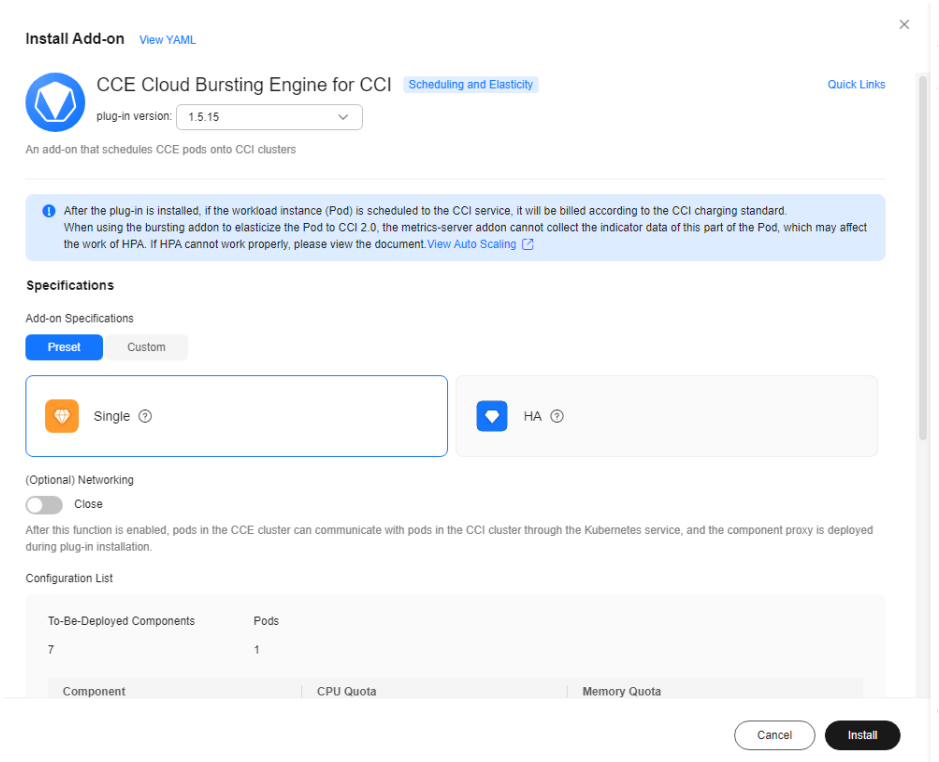


Table 9-20 Add-on parameters

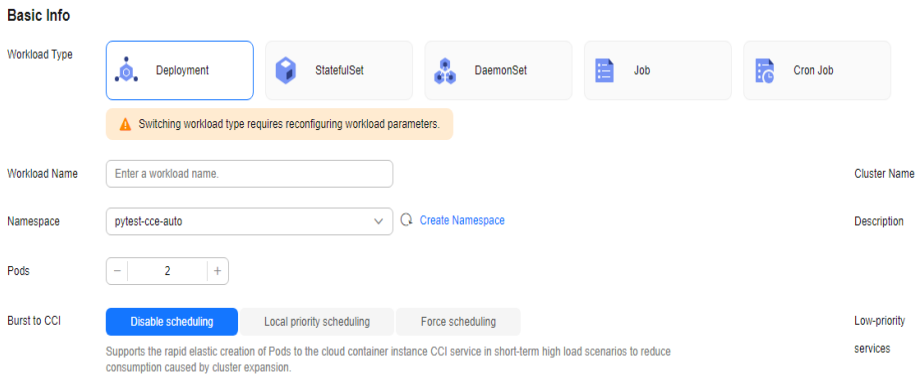
Parameter	Description
Version	Add-on version. There is a mapping between add-on versions and CCE cluster versions. For more details, see "Change History" in **CCE Cloud Bursting Engine for CCI**.

Specifications	Number of pods required for running the add-on.
	• If you select **Preset**, you can select **Single** or **HA**.
	• If you select **Custom**, you can modify the number of replicas, vCPUs, and memory of each add-on component as required.
	NOTE
	• The bursting add-on 1.5.2 or later uses more node resources. You need to reserve sufficient pods before upgrading the add-on.
	  • **Single** (only one pod for the add-on): There must be a node that has at least seven schedulable pods. If **Networking** is enabled, eight schedulable pods are required.
	  • **HA** (two pods for the add-on): There must be two nodes, each of which must have at least seven schedulable pods, a total of 14 schedulable pods. If **Networking** is enabled, eight schedulable pods are required on each node, a total of 16 schedulable pods.
	• The resource usage of the add-on varies depending on the workloads scaled to CCI. The pods, secrets, ConfigMaps, PVs, and PVCs requested by the services occupy VM resources. You are advised to evaluate the service usage and apply for VMs based on the following specifications: For 1,000 pods and 1,000 ConfigMaps (300 KB), nodes with 2 vCPUs and 4-GiB memory are recommended. For 2,000 pods and 2,000 ConfigMaps, nodes with 4 vCPUs and 8-GiB memory are recommended. For 4,000 pods and 4,000 ConfigMaps, nodes with 8 vCPUs and 16-GiB memory are recommended.

Networking	If this option is enabled, pods in the CCE cluster can communicate with pods in CCI through Services. The component proxy will be automatically deployed upon add-on installation. For details, see **Networking**.

Creating a Workload

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Workloads**.
4. Click **Create Workload**. For details, see **Creating a Workload**.
5. Specify basic information. Set **Burst to CCI** to **Force scheduling**. For more information about scheduling policies, see **Scheduling Pods to CCI**.



CAUTION

When you schedule a workload in a CCE cluster to CCI, TCP probes cannot be used for health check.

6. Configure the container parameters.
7. Click **Create Workload**.
8. On the **Workloads** page, click the name of the created workload to go to the workload details page.
9. View the node where the workload is running. If the workload is running on a CCI node, it has been scheduled to CCI.

Uninstalling the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

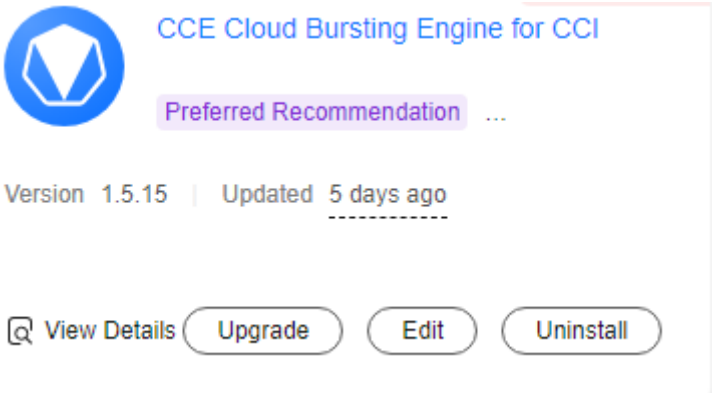


Table 9-21 Special scenarios for uninstalling the add-on

Scenario	Symptom	Description
There are no nodes in the CCE cluster that the bursting add-on needs to be uninstalled from.	Failed to uninstall the bursting add-on.	If the bursting add-on is uninstalled from the cluster, a job for clearing resources will be started in the cluster. To ensure that the job can be started, there is at least one node in the cluster that can be scheduled.
The CCE cluster is deleted, but the bursting add-on is not uninstalled.	There are residual resources in the namespace on CCI. If the resources are not free, additional expenditures will be generated.	The cluster is deleted, but the resource clearing job is not executed. You can manually clear the namespace and residual resources.

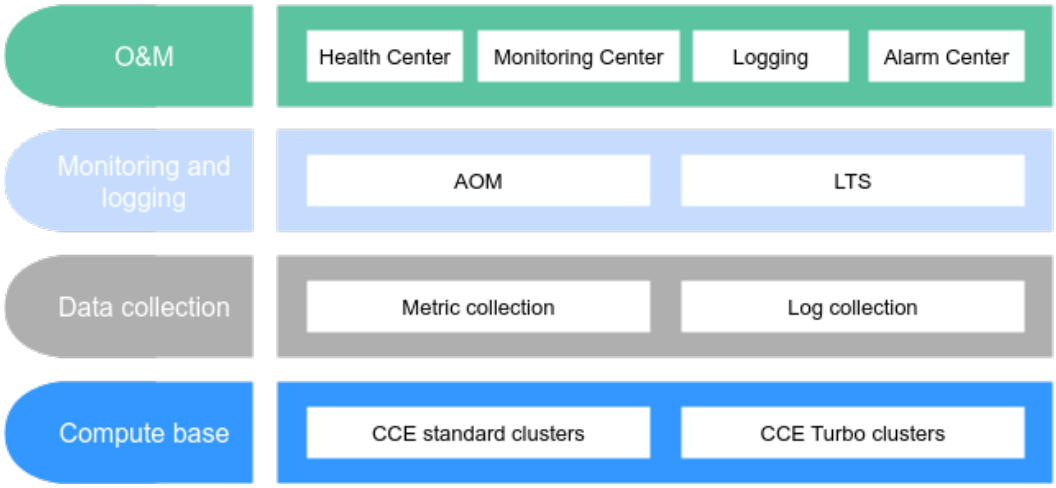
For more information about the bursting add-on, see **CCE Cloud Bursting Engine for CCI**.

10 O&M

10.1 Overview

Observability is an approach that engineers use to monitor the infrastructure and applications in a cloud native environment with the help of a variety of tools and techniques. By analyzing the collected metrics, logs, and traces, engineers can gain insights into the applications for easier troubleshooting. This section describes the observability architecture of CCE and main observability capabilities.

Figure 10-1 Observability architecture



The observability architecture consists of four parts: **compute base**, **data collection**, **monitoring and logging**, and **O&M**.

Compute Base

CCE allows you to create CCE Turbo clusters or CCE standard clusters as required. CCE provides a unified data collection solution for different cluster types, which ensures a consistent experience in cloud native observability. For details about CCE clusters, see **CCE Service Overview**.

Data Collection

Metric collection: An add-on based on Prometheus is provided for cloud native cluster monitoring. This add-on is much more lightweight and can be used out of the box. For details, see **Cloud Native Cluster Monitoring**.

Log collection: An add-on based on Fluent Bit and OpenTelemetry is provided for cloud native logging. This add-on features high performance and low resource usage. There are also CRD-based log collection policies, which are more flexible and easy to use. For details, see **Cloud Native Log Collection**.

Monitoring and Logging

Application Operations Management (AOM) is a one-stop, multi-dimensional O&M management platform for cloud applications. It monitors applications and related cloud resources in real time, analyzes application health, and provides flexible data visualization functions to help you detect faults in a timely manner.

Log Tank Service (LTS) collects log data from hosts and cloud services. LTS can process a massive number of logs efficiently, securely, and in real time, which enables you to gain insights into cloud services and applications and optimize their availability and performance. It also helps you in real-time decision-making, device O&M management, and service trend analysis.

O&M

CCE provides Health Center, Monitoring Center, Logging, and Alarm Center for O&M.

- **Health Center**

  Health diagnosis carefully monitors cluster health by leveraging the experience of our container O&M experts to detect cluster faults and identify risks in a timely manner. It provides rectification suggestions too.

- **Monitoring Center**

  Monitoring Center provides functions such as multi-dimensional data insights and dashboard. Monitoring Center provides monitoring views from dimensions such as clusters, nodes, workloads, and pods. It supports multi-level drill-down and association analysis. Dashboard gives you monitoring graphs for items such as the API server, CoreDNS, and PVC.

- **Logging**

  CCE works with LTS to collect logs of control plane components (kube-apiserver, kube-controller-manager, and kube-scheduler), Kubernetes audit logs, Kubernetes events, and container logs (stdout logs, text logs, and node logs).

- **Alarm Center**

  Alarm Center works with AOM 2.0 to allow you to create alarm rules and view alarms of clusters and containers.

10.2 Agency Permissions

O&M modules work closely with cloud services for cluster monitoring, alarm reporting, and notification. When you access the O&M modules for the first time,

the system will request permissions to access the cloud services in the region where you run your applications.

To minimize authorization, CCE fine-grained permissions are optimized. The permissions defined by system policies are now defined using API actions. (Each API has one API action.) If you have authorized the cloud services, you can optimize the permissions in one click.

After you agree to the authorization, agencies are automatically created in IAM to delegate required resource operation permissions in your account to Huawei Cloud CCE and AOM. For details about agencies, see **Cloud Service Delegation**. The following are agencies automatically created in IAM:

- cia_admin_trust

  This agency is used to to delegate the permissions required by the O&M modules to access other cloud services.

  To use the O&M modules in multiple regions, you need to apply for the Tenant Guest, CCE Administrator, and SWR Administrator permissions in each region. (Go to the IAM console, choose **Agencies**, and click **cia_admin_trust** to view the authorization records in each region.)

- aom_admin_trust

  For details about this agency, see **AOM Cloud Service Authorization**.

NOTE

The O&M modules may fail to run as expected if the required permissions are not granted. When using the O&M modules, do not delete or modify **cia_admin_trust** and **aom_admin_trust**.

Permissions Before Optimization

Table 10-1 cia_admin_trust permissions

Granted To	Policy/Role	Description
CCE	IAM ReadOnlyAccess	IAM users need to be able to access Monitoring Center and Alarm Center.
CCE	Tenant Guest	Monitoring Center and Alarm Center check the configurations of global resources (for example OBS or DNS) associated with clusters to identify invalid or inappropriate configurations.
CCE	CCE Administrator	Monitoring Center and Alarm Center need to be able to access CCE to obtain information about clusters, nodes, workloads, and other resources, so that they can help ensure resource health.
CCE	SWR Administrator	Monitoring Center and Alarm Center need to be able to access SWR to obtain image information.
CCE	SMN Administrator	Monitoring Center and Alarm Center need to be able to access SMN to obtain contact groups.

Granted To	Policy/Role	Description
CCE	AOM Administrator	Monitoring Center and Alarm Center need to be able to access AOM to obtain metrics.
CCE	LTS Administrator	Monitoring Center and Alarm Center need to be able to access LTS to obtain logs.

Table 10-2 aom_admin_trust permissions

Granted To	Policy/Role	Description
AOM	DMS UserAccess	AOM obtains subscription data from DMS.
AOM	ECS CommonOperations	AOM obtains system metrics and logs using UniAgents and ICAgents installed on ECSs.
AOM	CES ReadOnlyAccess	AOM synchronizes metrics from Cloud Eye.
AOM	CCE FullAccess	AOM synchronizes container metrics from CCE.
AOM	RMS ReadOnlyAccess	AOM CMDB manages cloud service instances.
AOM	ECS ReadOnlyAccess	AOM obtains system metrics and logs using UniAgents and ICAgents installed on ECSs.
AOM	LTS FullAccess	AOM obtains logs from LTS.
AOM	CCI FullAccess	AOM synchronizes container metrics from CCI.

Permissions After Optimization

Table 10-3 cia_admin_trust permissions

Policy Name	Policy Type	Policy Scope	Permission Set	Description
CCE Administrator	System-defined policy	Project	cce:*:*	CCE administrator permissions

Policy Name	Policy Type	Policy Scope	Permission Set	Description
CIAMonitorProjectPolicy	Custom policy	Project	cce:cluster:get	Obtains details about a cluster.
			cce:cluster:list	Lists all clusters.
			cce:addonInstance:list	Lists all add-on instances.
			cce:addonInstance:create	Creates an add-on instance.
			cce:addonInstance:delete	Deletes an add-on instance.
			cce:addonInstance:update	Updates an add-on instance.
			cce:node:get	Obtains details about a node.
			cce:node:list	Lists nodes.
			cce:nodepool:list	Lists all node pools in a cluster.
			aom:metric:set	Modifies monitoring configuration.
			aom:metric:get	Queries details about a metric.
			aom:metric:list	Lists metrics.
			aom:alarm:list	Lists alarms.
			aom:alarm:put	Clears AOM alarms.
			aom:actionRule:get	Queries an alarm linkage rule by ID.
			aom:actionRule:list	Lists alarm linkage rules.
			aom:actionRule:create	Creates an alarm linkage rule.
			aom:actionRule:update	Updates an alarm linkage rule.
			aom:actionRule:delete	Deletes an alarm linkage rule.
			aom:alarmRule:create	Adds a threshold rule.
			aom:alarmRule:list	Lists alarm rules.

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			aom:alarmRule:delete	Delete a threshold rule.
			aom:agency:get	Queries AOM authorization.
			lts:groups:get	Queries a specified log group.
			lts:groups:list	Lists log groups.
			lts:groups:create	Creates a log group.
			lts:logs:list	Lists logs.
			lts:topics:get	Queries a specified log topic.
			lts:topics:create	Creates a log topic.
			lts:topics:put	Updates a log topic.
			smn:topic:list	Lists topics.
			smn:topic:update	Updates a topic, including adding subscriptions to and deleting subscriptions from a topic.
			smn:topic:delete	Deletes a topic.
			smn:topic:create	Creates a topic.
			vpc:securityGroups:get	Queries security groups or details about a security group.
			vpc:vpcs:get	Queries details about a VPC.
			vpc:subnets:get	Queries subnets or details about a subnet.
			vpc:vpcs:list	Lists VPCs.
			vpcep:endpoints:list	Lists VPC endpoints.
			evs:quotas:get	Queries EVS disk quotas.

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			ecs:cloudServerQuotas:get	Queries tenant quotas.
			apm:icmgr:get	Obtains AOM 2.0 permissions.
			apm:icmgr:create	Grants AOM 2.0 permissions.

Table 10-4 aom_admin_trust permissions

Policy Name	Policy Type	Policy Scope	Permission Set	Description
AOM Global Access	Custom policy	Global	rms:*:list	Lists RMS resources.
			rms:*:get	Queries details about an RMS resource.
			rms:resources:listTagsForResource	Lists resource tags.
			rms:resources:listTags	Lists project tags.
			rms:resources:listResourcesByTag	List resource instances.
AOM UserAccess	Custom policy	Project	lts:topics:*	Full permissions for performing operations on log topics
			lts:groups:*	Full permissions for performing operations on log groups
			aom:metric:*	Full permissions for performing operations on a metric (AOM)
			aom:cmdbSubApplication:*	Full permissions for performing operations on a sub-application (AOM)

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			aom:cmdbResources:*	Full permissions for performing operations on resources (AOM)
			aom:cmdbEnvironment:*	Full permissions for performing operations on the environment (AOM)
			aom:cmdbComponent:*	Full permissions for performing operations on a component (AOM)
			aom:cmdbApplication:*	Full permissions for performing operations on an application (AOM)
			ecs:cloudServers:showServer	Queries details about an ECS.
			ecs:cloudServers:list	Lists ECSs.
			dms:instance:get	Queries details about a DMS instance.
			ces:metrics:list	Lists metrics (Cloud Eye).
			ces:metricData:list	Queries metrics (Cloud Eye).
			cci:namespace:list	Lists all namespaces.
			cce:cluster:list	Lists all clusters.
			cce:cluster:get	Obtains details about a cluster.
			cce:node:list	Lists nodes.
			cce:node:get	Obtains details about a node.
			apm:icmgr:*	Full permissions for performing operations on the APM collection component

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			lts:*:*	Full permissions for performing operations on LTS logs
			aom:*:list	Lists AOM instances.

10.3 Health Center

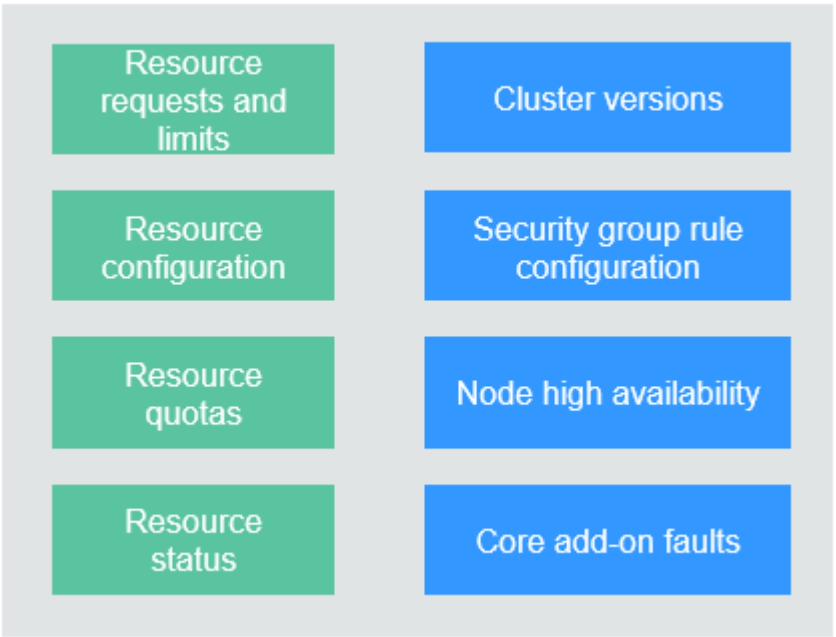
10.3.1 Overview

By leveraging the experience of our container O&M experts, health diagnosis monitors cluster health to detect cluster faults and identify risks in a timely manner. It also provides rectification suggestions.

Health Diagnosis Coverage

The following figure shows the health diagnosis coverage.

Figure 10-2 Health diagnosis coverage



Health Diagnosis Capabilities

- Out-of-the-box diagnosis (without Monitoring Center enabled)
- Comprehensive check of cluster health (with Monitoring Center enabled)

- Health scores based on diagnosis results
- Scheduled inspection and visualized inspection results
- Inspection history for analyzing fault causes
- Risk levels and rectification suggestions

Application Scenarios

- You can check the cluster health before a cluster change and perform health diagnosis at any time.
- You can set a scheduled inspection to identify cluster risks on schedule.

The following table lists the inspection items.

Dimension	Inspection Item
O&M	• Cluster O&M • Cluster security group configuration • Cluster resource planning • Cloud service quota
Resources and services	• Storage add-on (everest) status • Logging add-on (log-agent) status • Domain name resolution add-on (coredns) status • Worker node load status • Worker node status • Pod configuration • Pod workload • Pod status

For more information, see **Diagnosis Items and Rectification Solutions**.

10.3.2 Cluster Diagnosis

CCE provides one-click health diagnosis on clusters, nodes, workloads, core add-ons, and external dependencies to help you quickly locate cluster faults (if any). This section describes how to perform health diagnosis in a cluster.

Prerequisites

- You have **obtained resource permissions**.
- The cluster version is v1.17 or later.
- The cluster is in the **Running** state.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Health Center**.

You can perform basic diagnosis without enabling Monitoring Center. To experience more diagnosis services, enable Monitoring Center by referring to **Enabling Monitoring Center**.

----End

Scheduled Inspection

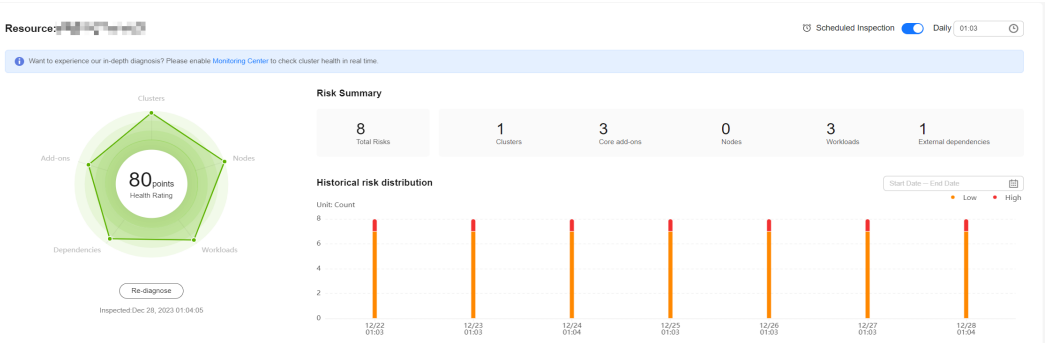
Enable **Scheduled Inspection** in the upper right corner and configure the start time of the inspection. The inspection task will automatically start at the specified time. A cluster can be inspected only once every day.

Figure 10-3 Scheduled inspection

Manual Health Diagnosis

When you use health diagnosis for the first time, click **Diagnose Now** to start the diagnosis and wait for a while. The health diagnosis page displays the health score, risk distribution radar chart, diagnosis risk summary, historical risk distribution, and diagnosis result.

Figure 10-4 Diagnosis overview

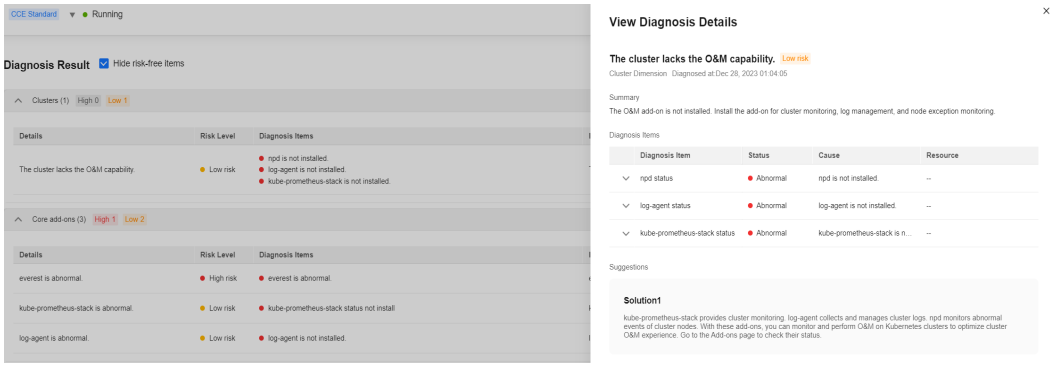


Diagnosis Result

After the diagnosis is complete, the page will be automatically refreshed to display the diagnosis result. Normal items are hidden by default in the result.

Kubernetes problems will be summarized from the abnormal items. Troubleshooting suggestions will also be provided. You can click **View Diagnosis Details** to view details about a specific diagnosis item and related abnormal resources. In some cases, there are also troubleshooting documents on the diagnosis details page for your reference.

Figure 10-5 Diagnosis result



10.3.3 Workload Diagnosis

Based on previous O&M experience, load faults usually occur frequently, and handling these faults is time-consuming. To improve O&M efficiency, CCE introduces single-pod diagnosis to help O&M teams locate and rectify faults more efficiently.

The diagnosis tool uses Kubernetes native APIs, Kubernetes events, logs, and metrics to comprehensively analyze and determine problems and provide specific rectification suggestions. This comprehensive diagnosis method not only improves the accuracy of fault locating, but also significantly reduces the workload of O&M personnel, thereby improving the overall O&M efficiency.

Prerequisites

- You have **obtained resource permissions**.
- The cluster version is v1.17 or later.
- The cluster is in the **Running** state.

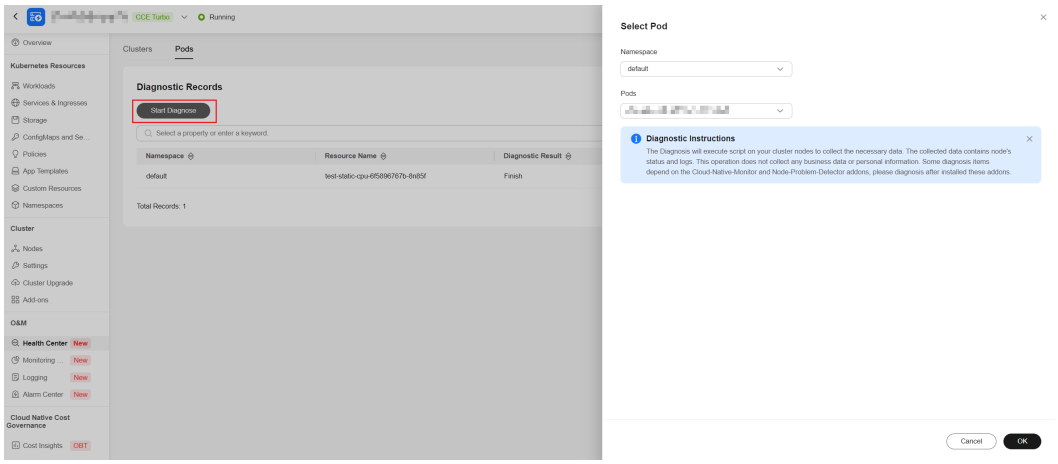
Selecting a Pod for Diagnosis

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Health Center**. Then, click the **Pods** tab.

Step 3 On the displayed tab, click **Start Diagnosis**. Then, select the pod to be diagnosed and click **OK**.

Figure 10-6 Start Diagnosis



Step 4 After the diagnosis is complete, click **View Details** to view the diagnosis result.

Figure 10-7 Diagnosis completed

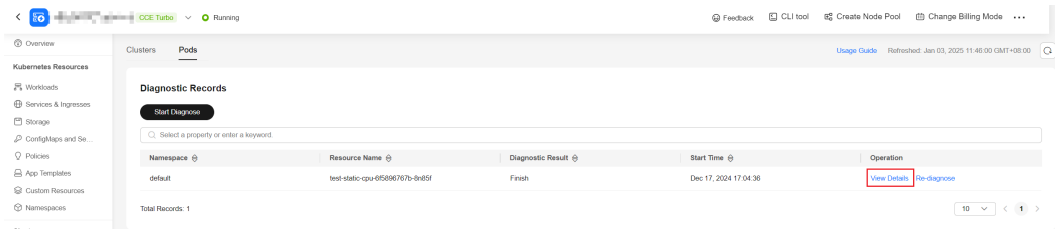
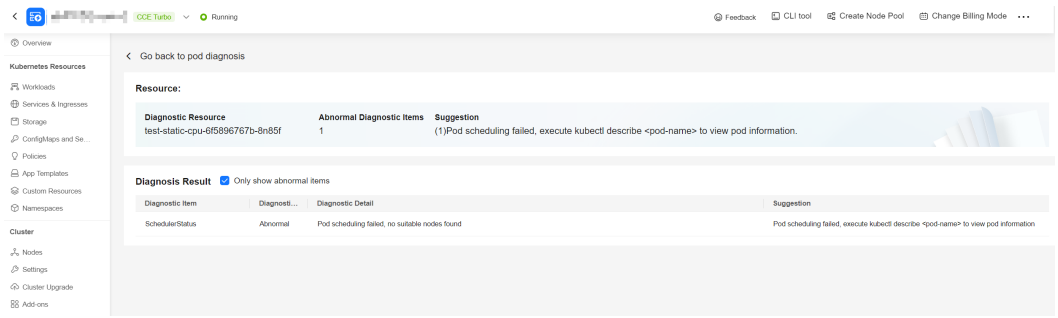


Figure 10-8 Viewing the diagnosis result



----End

10.3.4 Diagnosis Items and Rectification Solutions

Cluster Diagnosis Items and Rectification Solutions

Clusters

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Cluster resource planning	Whether HA is enabled for the master node	Yes	The cluster has only one master node or a master node is abnormal. If another master node is faulty, the cluster is unavailable, affecting service reliability. To improve service resilience, use an HA cluster or rectify node exceptions. When a master node is faulty, the cluster is still available.
	Whether the cluster CPU request has exceeded 80%	Yes	A request is the minimum CPU or memory a workload needs. Plan required resources based on service requirements. For details, see **Configuring Container Specifications**.
	Whether the cluster memory request has exceeded 80%	Yes	
	Whether the cluster version has reached the end of service	No	After the cluster version has reached the end of service, CCE will no longer support the creation of new clusters. You will also no longer be provided with technical support, including new feature updates, vulnerability or issue fixes, new patches, work order guidance, and online checks for the cluster version. The CCE SLA is not valid for such clusters. Go to the **Clusters** page to upgrade the cluster version. For details, see **Process and Method of Upgrading a Cluster**.
Cluster O&M	Whether the Cloud Native Cluster Monitoring add-on is normal	No	The Cloud Native Cluster Monitoring add-on provides one-stop cluster monitoring. Go to the **Add-ons** page to install this add-on and check its status. For details, see **Cloud Native Cluster Monitoring**.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether the Cloud Native Log Collection add-on is normal	No	The Cloud Native Log Collection add-on collects and manages workload logs. Go to the **Add-ons** page to install this add-on and check its status.
	Whether the CCE Node Problem Detector add-on is normal	No	The CCE Node Problem Detector add-on monitors the nodes. Go to the **Add-ons** page to install this add-on and check its status. For details, see **CCE Node Problem Detector**.
Cluster configuration	Whether the security group is correctly configured	No	Invalid cluster security group configuration makes it impossible for the nodes to communicate with each other. Retain the default security group configuration.

Core Add-ons

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
CoreDNS status	Whether CoreDNS is normal	No	CoreDNS is a mandatory add-on that provides domain name resolution for clusters. If this add-on is not installed or is abnormal, the overall service response of the cluster will be affected. Go to the **Add-ons** page to install this add-on or check the add-on status.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether the CPU usage of CoreDNS has exceeded 80% in the last 24 hours	Yes	CoreDNS provides domain name resolution for clusters. If the resource usage is too high, the add-on may be overloaded. Domain name resolution will be affected, and the latency is increased. To prevent services from being affected, analyze the recent QPS of CoreDNS. Go to Monitoring Center, click the **Dashboard** tab, and select the CoreDNS view to view the instance metrics. If the metric values reach the thresholds, adjust the specifications.
	Whether the memory usage of CoreDNS has exceeded 80% in the last 24 hours	Yes	
	Whether CoreDNS failed to resolve domain names in the last 24 hours	Yes	If CoreDNS failed to resolve domain names, services are affected.
	Whether the P99 latency of CoreDNS has exceeded 5s in the last 24 hours	Yes	If the latency increases, responses to DNS requests become slow.
CCE Container Storage (Everest) status	Whether Everest is normal	No	Everest is a mandatory add-on that provides cloud storage services for clusters. If this add-on is not installed or is abnormal, the cluster storage capability is affected. Go to the **Add-ons** page to install this add-on or check the add-on status.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether the CPU usage of everest-controller has exceeded 80% in the last 24 hours	Yes	Everest provides cloud storage services for clusters. If the resource usage is too high, the add-on may be overloaded, and cluster cloud storage is affected. To prevent cloud storage from being affected, analyze the recent load of everest-controller. Choose **Monitoring Center** > **Workloads** to view the Everest instance metrics. If the metric values reach the thresholds, adjust the specifications. For details, see **CCE Container Storage (Everest)**.
	Whether the memory usage of everest-controller has exceeded 80% in the last 24 hours	Yes	
Cloud Native Cluster Monitoring status	Whether Cloud Native Cluster Monitoring is normal	No	The Cloud Native Cluster Monitoring add-on provides one-stop cluster monitoring. Go to the **Add-ons** page to install this add-on and check its status.
	Whether the CPU usage of the prometheus workload has exceeded 80% in the last 24 hours	Yes	The Cloud Native Cluster Monitoring add-on provides cluster monitoring. If the resource usage is too high, this add-on may be overloaded, and cluster monitoring is affected. Choose **Monitoring Center** > **Workloads** to view the prometheus instance metrics. If the metric values reach the thresholds, adjust the specifications. **NOTE** The PVC resource usage is checked when this add-on is deployed with local data storage enabled. In this mode, the collected metrics are stored in the cluster PV.
	Whether the memory usage of the prometheus workload has exceeded 80% in the last 24 hours	Yes	
	Whether the PVC usage of prometheus-server exceeded 80% when the prometheus workload is deployed in server mode	Yes	

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether OOM has occurred for the prometheus workload in the last 24 hours	No	The Cloud Native Cluster Monitoring add-on provides cluster monitoring. OOM occurs when the memory usage of the add-on instance reaches the limit. As a result, metric reporting will be affected, and non-HA cluster monitoring will be unavailable. Adjust the specifications of the prometheus instance.
CCE Cluster Autoscaler status	Whether the CCE Cluster Autoscaler add-on is available when auto scaling is enabled for node pools	No	The CCE Cluster Autoscaler add-on provides auto scalability for clusters. If this add-on is abnormal, auto scaling that has been enabled for a node pool becomes unavailable. Check the add-on status on the **Add-ons** page. **NOTE** The add-on status is checked only when auto scaling is enabled for node pools.
Cloud Native Log Collection status	Whether the Cloud Native Log Collection add-on is normal	No	The Cloud Native Log Collection add-on collects and manages workload logs. Go to the **Add-ons** page to install this add-on and check its status.
	Whether default LTS log group and log streams are created	No	The default event log group and log streams are the basic units for event reporting in Monitoring Center. If there are no log group and log streams, event reporting is unavailable. For details about how to create a log group and log streams, see **Collecting Container Logs Using Cloud Native Log Collection**.

Nodes

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Node status	Whether nodes are ready	Yes	If a node is not ready, services running on the node may be affected. Rectify the fault in a timely manner.
	Whether nodes can be scheduled	Yes	If a node cannot be scheduled, node resources cannot be used. Go to the CCE node management page to check whether the node status meets the expectation.
	Whether kubelet is normal	Yes	kubelet is a key component of the nodes. If kubelet is abnormal, the nodes may be abnormal and the pod status is inconsistent with that on the API server. Run the **journalctl -l -u kubelet** command on each node to view the kubelet log and locate the cause.
Node configuration	Whether the memory requests of pods on a node have exceeded 80% of the node memory	Yes	The minimum CPU and memory requested by a node determine whether new applications can be scheduled to the node. If the request is higher than the available resource, no applications will be scheduled to the node. The results show that the resource requests have exceeded the minimum values. Plan required resources for your applications based on the results.
	Whether the CPU requests of pods on a node have exceeded 80% of the node CPU	Yes	
Resource usage of nodes	Whether the CPU usage of a node has exceeded 80% in the last 24 hours	Yes	If the node CPU usage is too high, the workloads running on the node will be affected. Go to Monitoring Center to view the node CPU usage. Then plan required node resources or expand the node capacity.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether the memory usage of a node has exceeded 80% in the last 24 hours	Yes	If the memory usage of a node is too high, there are OOM risks, affecting service availability on the node. Go to Monitoring Center to view the node memory usage. Then plan required node resources or expand the node capacity.
	Whether the disk usage of a node has exceeded 80%	Yes	If the node disk usage is too high, the pods will be affected. Expand capacity in a timely manner. Run the following commands to view disk details: • **lsblk**: information about all available block devices • **df -h**: available disk space of each mounted disk • **fdisk -l**: all partitions
	Whether the number of PIDs for a node exceeds the limit	Yes	The node is experiencing PID pressure, making it unstable. Release unnecessary processes on the node or **modify the PID limit**. Run the following commands to view PID details: • **sysctl kernel.pid_max**: the maximum number of PIDs • **ps -eLf|awk '{print $2}' | sort -rn| head -n 1**: the current maximum PID • **ps -elT | awk '{print $4}' | sort | uniq -c | sort -k1 -g | tail -5**: the top five processes that occupy the most SPIDs
	Whether OOM has occurred on a node in the last 24 hours	Yes	If OOM occurs on a node, service functions on the node are affected. Go to Monitoring Center to view the node memory. Then plan required resources or expand the capacity.

Workloads

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Pod status	Whether pods are normal	No	If a pod fails to function normally, the workload performance for that pod may deteriorate. If there are no replicas available, the pod may be inaccessible. Run the following commands to view pod details: • **kubectl get pod <PodName> -n <Namespace> -o yaml**: pod configuration • **kubectl describe pod <PodName> -n <Namespace>**: pod events • **kubectl logs <PodName> -n<Namspace> -c <ContainerName>**: container logs
Pod workload	Whether OOM has occurred on a pod in the last 24 hours	No	If OOM occurs on a pod, service functions of the pod are affected. Go to Monitoring Center to view the pod memory and adjust the workload specifications.
	Whether the CPU usage of a pod has exceeded 80% in the last 24 hours	Yes	If the resource usage is too high, the pod may be overloaded. This increases the latency and slows down service responses. Choose **Monitoring Center** > **Pods** to view the instance metrics. If the metric values reach the thresholds, adjust the container specifications.
	Whether the memory usage of a pod has exceeded 80% in the last 24 hours	Yes	
Pod configuration	Whether requests are configured for containers in a pod	No	If requests are not configured, Scheduler will be affected, and pods may be scheduled to nodes whose resources cannot meet requirements. High requests will also reduce the resource usage of nodes.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Pod probe configuration	Whether liveness probes are configured for containers in a pod	No	If no liveness probes are configured, application exceptions in a pod cannot be detected, and the pod cannot be restarted in a timely manner, which will affect the QoS. Configure liveness probes for the pod to avoid abnormal applications and restart the pod in a timely manner if applications fail to function normally.
	Whether readiness probes are configured for containers in a pod	No	If no readiness probes are configured, requests are still sent to the pod even if it becomes abnormal, which will affect the QoS. Configure readiness probes for the pod so that requests can still be handled even if applications are abnormal.

External Dependencies

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Resource quotas of a node	Whether 90% or more of the EVS disk quota has been used	Yes	Sufficient resource quotas are required for node creation in a cluster. If there are insufficient resource quotas, choose **Resources** > **My Quotas** and contact customer service to apply for account quotas.
	Whether 90% or more of the ECS quota has been used	Yes	

Pod Diagnosis Items and Rectification Solutions

Table 10-5 Pod diagnosis items and rectification solutions

Diagnosis Item		Rectification Solution
FailedScheduling	Insufficient memory	The available memory of the node is insufficient. Expand the memory capacity.
	Insufficient cpu	The available CPU of the node is insufficient. Expand the CPU capacity.
	skip schedule deleting pod	The pod is being deleted.
	Other information	If the pod fails to be scheduled, view the pod information. `kubectl describe <pod-name>`
FailedAttachVolume		Check the status of the Everest add-on and node network connection, and ensure that the node has required permissions.
FailedMount		Check the status of the Everest add-on and node network connection, and ensure that the node has required permissions.
InvalidDiskCapacity		Check the disk capacity of the node and the actual available space. Ensure that the disk capacity is correctly set and meets the storage requirements of applications or services. Delete unnecessary files to release disk space. If a dynamic volume is used, ensure that the storage backend configuration is correct and available. Expand the disk capacity or adjust the storage requirements of applications or services as needed.
BackOffPullImage		Ensure that the image tag is correct.
FailedPullImage		Ensure that the image tag is correct.
ErrImageNeverPull		Check the local image. You are advised to set the image pull policy to **IfNotPresent** or **Always**.
InspectFailed		Check the integrity of the image.
FailedPostStartHook		Check the configuration and script of the post-start hook to ensure that they are correct. View the hook execution log to obtain the error information and rectify the fault in the hook script based on the error information. If possible, manually execute the post-start hook script to check whether the environment or permissions are correct.

Diagnosis Item	Rectification Solution
FailedPreStopHook	Check the configuration and script of the pre-stop hook to ensure that they are correct. View the hook execution log to obtain the error information and rectify the fault in the hook script based on the error information. If possible, manually execute the pre-stop hook script to check whether the environment or permissions are correct.
ProbeWarning	Check the probe configuration to ensure that the probe is correctly configured and can correctly evaluate the container health status. View the alarm information to find the possible faults, and adjust the probe configuration or rectify the faults in the container as needed.
Unhealthy	Check the pod or container logs to find error information. Ensure that applications or services are correctly started and running in the container. Check the container resource usage to determine whether resources are insufficient. Take measures based on logs and monitoring information, such as restarting pods or containers to rectify application or service faults.
FailedCreatePodContainer	Check the pod and container configurations to ensure that the YAML file is correct, including the container image, resource request, and limit.
Preempting	You are advised to set proper resource requests and limits for the load to prevent preemption caused by insufficient resources.
Killing	Check the resource usage and ensure that the resource requests and limits of pods and nodes are properly set to prevent containers from being terminated due to insufficient resources.

10.4 Monitoring Center

10.4.1 Overview

Monitoring Center is a next-generation O&M platform for cloud native containers. It monitors applications and resources in real time, collects metrics and events to analyze application health statuses, and visualizes multi-dimensional data.

Compatible with mainstream open source components, Monitoring Center supports quick fault locating.

Functions

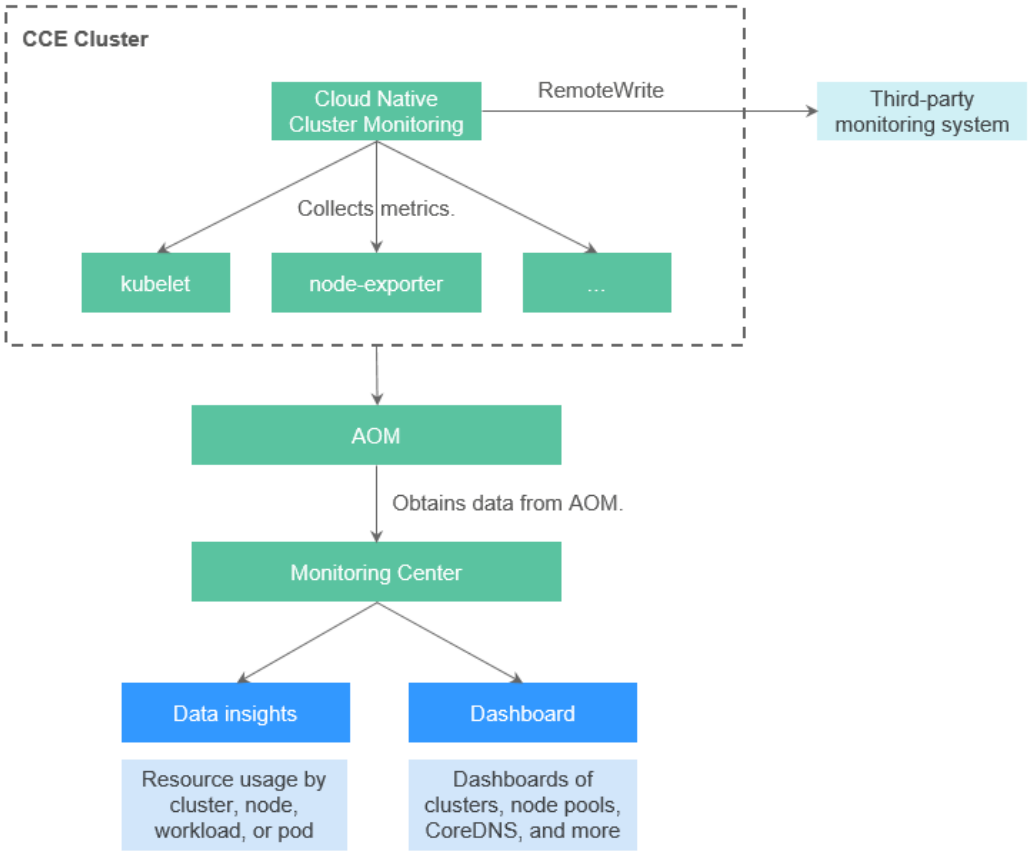
- Multi-dimensional data insights: comprehensively monitor Kubernetes native containers and displays the metrics of **clusters**, **nodes**, **workloads**, **events**, and **pods** for the health status and load of clusters.
- **Dashboard**: shows various graphs such as line graphs and digit graphs on the same screen for comprehensive data display.

Advantages

- Monitoring Center is deeply integrated with Prometheus, a mature monitoring project of the Cloud Native Computing Foundation (CNCF). It brings in observability for your cloud native applications by collecting, storing, and visually presenting O&M data, such as key metrics and events.
- Monitoring Center provides full-stack monitoring from cloud native infrastructure resources to application workloads, enabling you to clearly perceive the infrastructure and application load status anytime anywhere.
- Monitoring Center monitors Kubernetes clusters, nodes, and pods, enables end-to-end tracing and visualization for services, and provides cluster health diagnosis, greatly speeding up fault analysis and locating.
- Monitoring Center provides ready-to-use add-ons, data collection, and cloud native cluster monitoring. Compared with monitoring developed based on open source components, it is more competitive in reliability, availability, and deployment.
- Monitoring Center provides lightweight add-ons for metric collection. Compared with Prometheus, it greatly reduces resource usage and facilitates deployment.

Monitoring Center Architecture

Figure 10-9 Monitoring Center architecture



The cloud native cluster monitoring add-on collects metrics exposed by exporters in CCE clusters and writes the data to the AOM instance in Prometheus RemoteWrite mode.

Monitoring Center provides multi-dimensional data insights and dashboard based on the metrics stored in the AOM instance.

Based on the RemoteWrite feature, the cloud native cluster monitoring add-on reports monitoring metrics in a cluster to the third-party monitoring platform through Bearer Token authentication.

Prometheus Monitoring

Prometheus has become the most common tool for cloud native observability. Its powerful monitoring capability and active community ecosystem enable Prometheus projects to be hosted under the CNCF. Currently, the CCE add-on page provides an add-on (**Cloud Native Cluster Monitoring**) for monitoring Kubernetes clusters.

Based on the Prometheus monitoring ecosystem, AOM provides hosted Prometheus instances for CCE, which are suitable for monitoring CCE clusters and applications running on them. By default, AOM instances integrate the cloud-

native monitoring add-ons of CCE clusters. After Monitoring Center is enabled, metrics are automatically reported to the specified AOM instances.

AOM ICAgent Monitoring

As the collector of AOM, ICAgent runs on hosts to collect metrics, logs, and application performance data in real time. For hosts purchased on the ECS or BMS console, install the ICAgent manually. For cluster nodes, ICAgent is automatically installed.

10.4.2 Enabling Monitoring Center

After Monitoring Center is enabled for a cluster, the Cloud Native Cluster Monitoring add-on used to collect metrics is installed. Cluster metrics are collected and reported to AOM instances. This section describes how to enable Monitoring Center.

NOTICE

- After Monitoring Center is enabled, cluster metrics are reported to the selected AOM instance. Basic metrics can be monitored for free, but custom metrics are billed based on the standard pricing of AOM. For details, see **Pricing Details**.
- The Cloud Native Cluster Monitoring add-on consumes cluster resources. Ensure that there are required cluster resources for installing the add-on. For details about the resource consumption, go to the add-on details page.

Prerequisites

You have an account in the **admin** user group to delegate CCE and its dependent services.

The authorization dialog box is automatically displayed on the **Monitoring Center** page. After you confirm the authorization, the system automatically completes the authorization. For details about permission types, see **Agency Permissions**.

Constraints

- The cluster version must be v1.17 or later.
- Before using Monitoring Center, you need to use an account in the **admin** user group to delegate CCE and its dependent services. After the authorization is complete, users with the CCE Administrator role or CCE FullAccess permission can perform all operations on Monitoring Center. Users with the CCE ReadOnlyAccess permission can view all resource information but cannot perform any operations.
- Self-built Prometheus or the Prometheus add-on (**Prometheus (EOM)**) is not installed in the cluster.

Enabling Monitoring Center

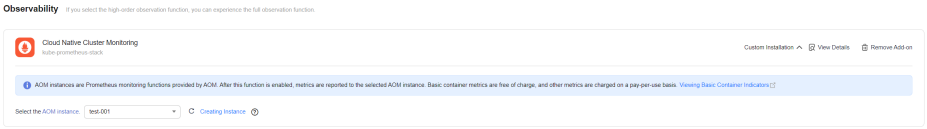
- **Enabling Monitoring Center during cluster purchase**

a. Log in to the CCE console and purchase a cluster.

b. On the **Select Add-on** page, select the **Cloud Native Cluster Monitoring** add-on.

c. On the **Add-on Configuration** page, select the AOM instance to be interconnected with the add-on. If there is no access code, create one first.

Figure 10-10 Enabling cluster monitoring



d. After the cluster is created, create a node on the **Nodes** tab. After the node is created, the Cloud Native Cluster Monitoring add-on will be automatically deployed on the node.

- **Enabling Monitoring Center on the Monitoring Center page**

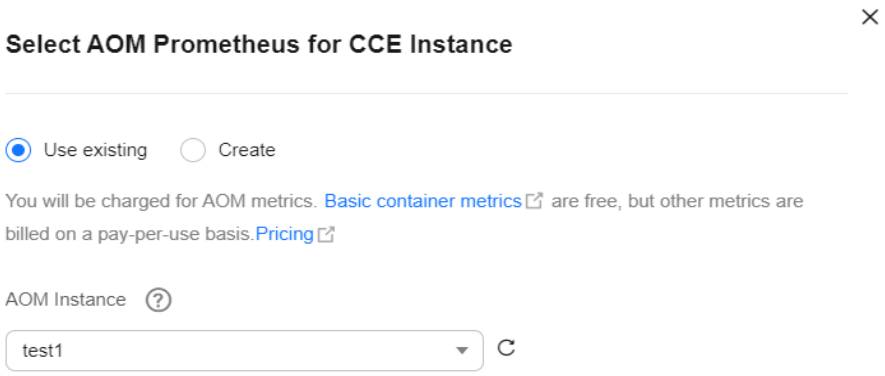
a. Click the cluster name to access the cluster console. In the navigation pane, choose **Monitoring Center**.

b. Check whether there is self-built Prometheus in the cluster. The Cloud Native Cluster Monitoring add-on installed when Monitoring Center is enabled may conflict with self-built Prometheus.

If your cluster already has self-built Prometheus, you can select **Compatibility Mode** to enable the compatibility mode. The Cloud Native Cluster Monitoring add-on will be installed in the **cce-monitoring** namespace and works with self-built Prometheus. However, there are some restrictions on the compatibility mode. For details, see **Cloud Native Cluster Monitoring Is Compatible with Self-Built Prometheus**.

c. Click **Enable** and select the AOM instance that metrics are reported to.

Figure 10-11 Enabling Monitoring Center



d. Wait for 3 to 5 minutes until the monitoring data is reported to the AOM instance.

The functions of Monitoring Center are available.

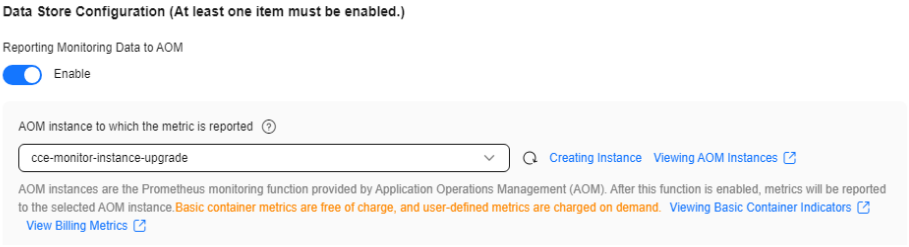
- **Enabling Monitoring Center on the Add-ons page**

a. Click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.

b. Select the **Cloud Native Cluster Monitoring** add-on and click **Install**.

c. Select **Reporting Monitoring Data to AOM**. The other two data storage configuration items can be selected as needed.

Figure 10-12 Installing the Cloud Native Cluster Monitoring add-on



d. Wait for 3 to 5 minutes until the monitoring data is reported to the AOM instance.

The functions of Monitoring Center are available.

NOTE

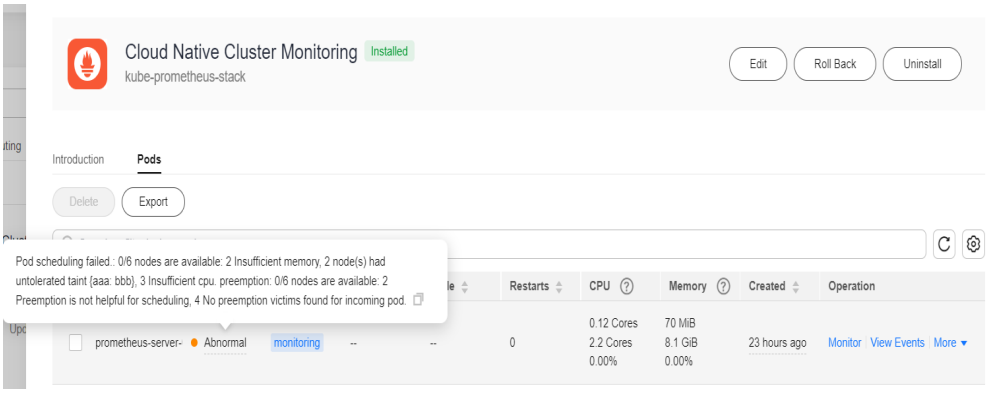
To disable Monitoring Center, uninstall the Cloud Native Cluster Monitoring add-on on the **Add-ons** page or disable the option for interconnecting with AOM.

FAQ

- Failed to enable Monitoring Center because the add-on is abnormal.

  Solution: Go to the **Add-ons** page to view the list of installed add-ons. Click the name of the Cloud Native Cluster Monitoring add-on to expand the instance list. Check the events of abnormal pods and locate the fault based on the error information.

Figure 10-13 Abnormal add-on



- There is no data on the **Monitoring Center** page.

  Solution:

a. Go to the **Add-ons** page to view the list of installed add-ons. Click the name of the Cloud Native Cluster Monitoring add-on to expand the instance list and check whether the Prometheus instance is running normally. If the Prometheus instance is not running normally, query the events of pods to obtain the exception information.

   For example, if "0/6 nodes are available: 1 Insufficient cpu, 2 node(s) had taint {cie.manage: proxy}, that the pod didn't tolerate, 3 node(s) had taint {node.kubernetes.io/unreachable: }, that the pod didn't tolerate" is displayed, the CPU of one node is insufficient and the remaining five nodes are marked with taints. As a result, pods cannot be scheduled.

b. If the add-on is normal, you can query the logs of the Prometheus instance and check whether the logs contain error information. Error information related to remote_write indicates that metrics fail to be reported. In this case, check whether the network for reporting the metrics is normal.

10.4.3 Managing Collection Tasks

Monitoring Center provides visualized collection task management. All settings can be retained during the Cloud Native Cluster Monitoring add-on upgrade.

Prerequisites

Cloud Native Cluster Monitoring 3.11.0 or later has been installed.

Managing Collection Tasks

NOTICE

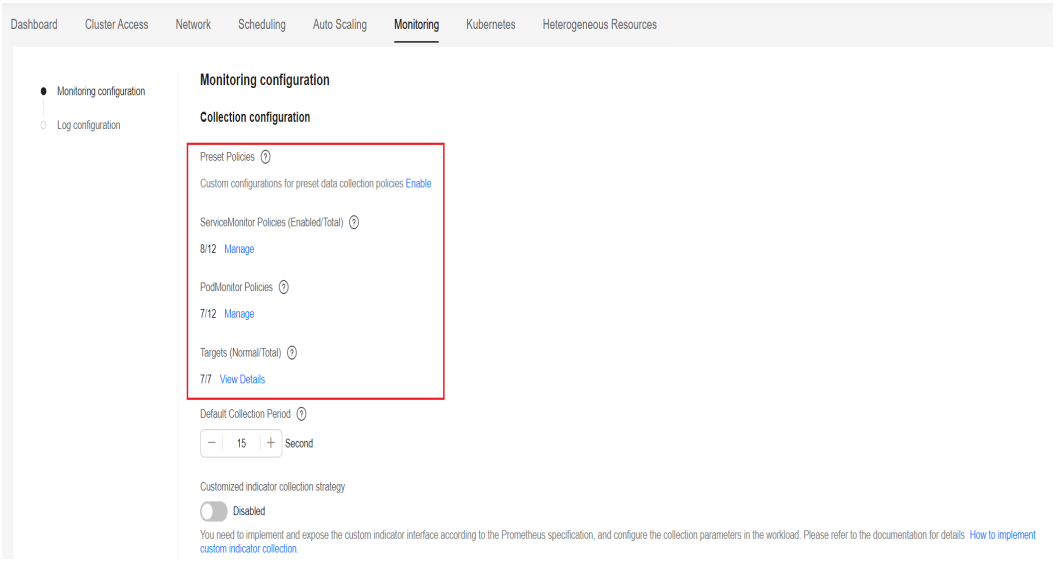
After you enable collection tasks that are disabled by default and add metrics except **free basic metrics**, you will be billed for the metrics based on AOM's standard pricing if your cluster is connected to AOM. For details, see **Pricing Details**.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Settings**. Then, click the **Monitoring** tab.

Step 3 Modify the configuration items in **Collection configuration**.

The collection task configuration consists of the **Preset Collection Configuration**, **ServiceMonitor Collection Configuration**, **PodMonitor Collection Configuration**, and **Targets Collection Configuration** configuration items.



----End

Preset Collection Configuration

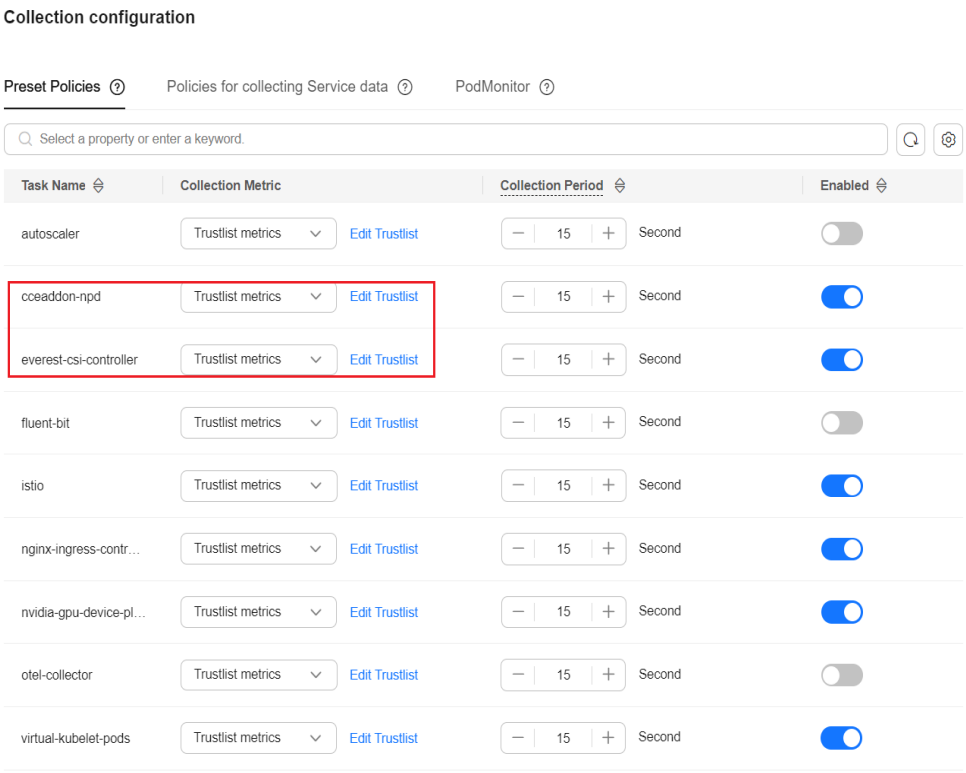
NOTE

To ensure consistent default behavior of add-ons, preset collection is disabled by default. You are advised to enable preset collection.

Enabling preset collection will convert tasks from ServiceMonitor or PodMonitor to visualized collection tasks. The Cloud Native Cluster Monitoring add-on allows for easy management of these tasks, enabling or disabling them based on service needs. You can also add metrics except the **basic free ones**.

The configuration for managing preset collection tasks can be inherited and retained during the add-on upgrade. In addition, the kube-state-metrics and node-exporter workloads will be centrally managed by Operator. Your custom configuration for the two workloads will be retained to the maximum extent during the add-on upgrade.

- Metric collection management



You can manage the metric collection behavior of each preset collection task as needed.

- If you choose to collect all metrics, all metrics specified by a collection task will be collected.
- If you select the metric collection trustlist, you can add metrics (except **free basic metrics**) to the trustlist as needed to more accurately control the custom collection content, reduce resource consumption of your cluster, and reduce metric reporting costs.

- Collection task period management

  You can customize the collection period of a collection task as needed.

  NOTICE

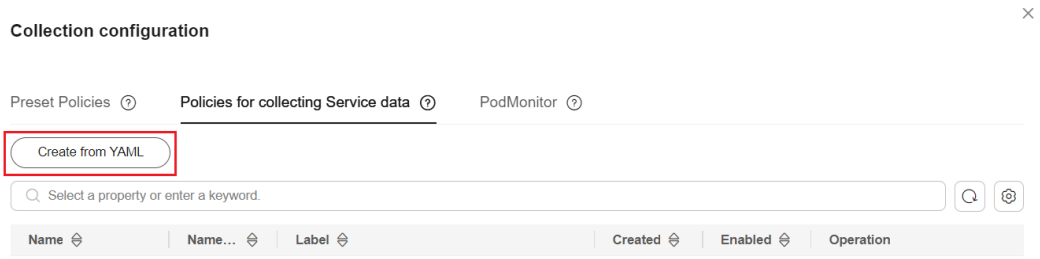
  You are advised to set the collection periods of the kubelet, kubelet-cadvisor, kube-state-metrics, and virtual-kubelet-pods collection tasks to the same value.

- Starting and Stopping a Collection Task

  You can enable or disable a system collection task as needed.

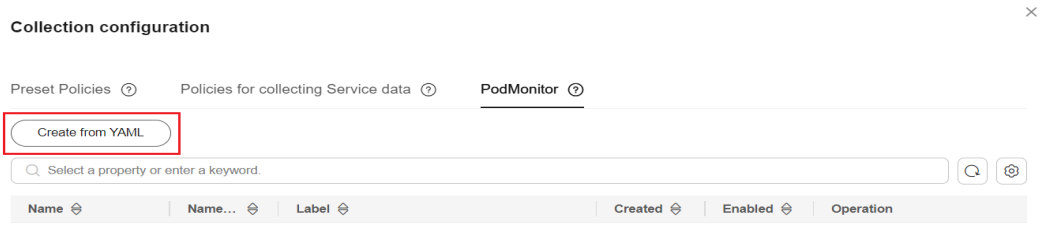
ServiceMonitor Collection Configuration

You can create, modify, delete, start, and stop a ServiceMonitor as needed. For details about how to create a ServiceMonitor, see **Method 4: Configuring ServiceMonitor**.



PodMonitor Collection Configuration

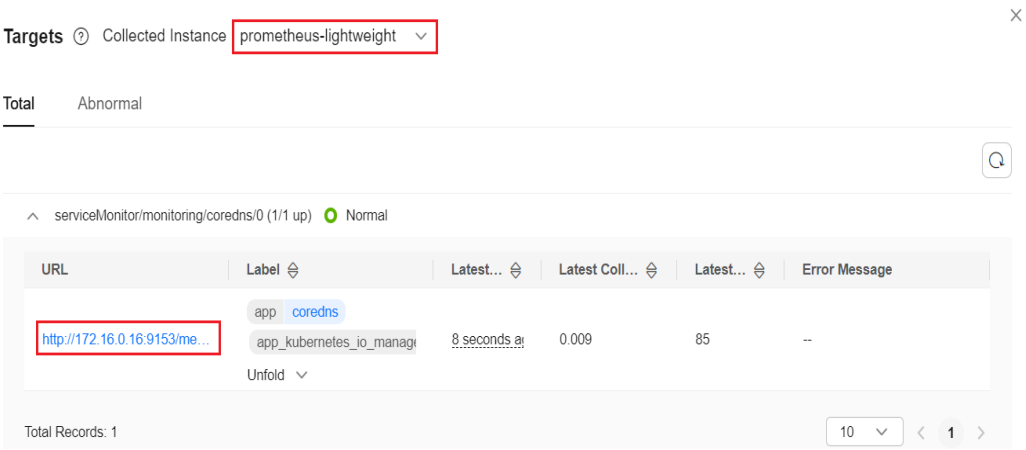
You can create, modify, delete, start, and stop a PodMonitor as needed. For details about how to create a PodMonitor, see **Method 3: Configuring PodMonitor**.



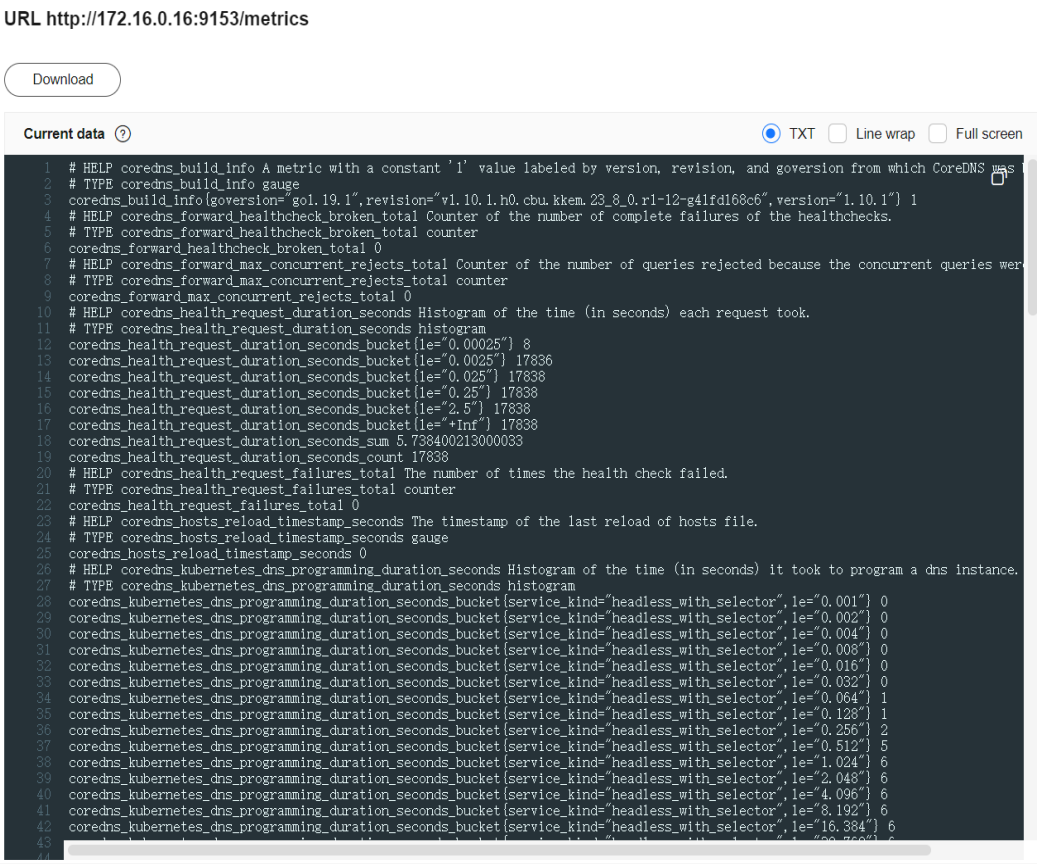
Targets Collection Configuration

On the **Targets** page, you can view the collection task status, including the collection endpoint, label, latest collection time, latest collection duration, number of latest collected samples, and error information.

If **sharding** is enabled for your Cloud Native Cluster Monitoring, there will be multiple collection instances. You can switch one collection instance to another in **Collected Instance**.



When the Cloud Native Cluster Monitoring add-on is deployed with local data storage disabled, you can directly access the collection endpoint to view, analyze, and manage collection results.



Why Is 403 Displayed in Collection Endpoint Access? How Do I Handle It?

Root Cause

Authentication has been configured for collection tasks in the ServiceMonitor or PodMonitor format corresponding to your collection endpoint. For security purposes, the endpoint to be authenticated cannot be accessed by default.

Solution: You can configure to allow access to endpoints with authentication.

NOTICE

If you allow access to endpoints with authentication, the endpoints to be authenticated can be directly accessed by accessing the prometheus-lightweight service in the cluster. For this reason, do not expose the prometheus-lightweight service port outside the cluster.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **ConfigMaps and Secrets**. Then, set **Namespace** to **All namespaces** and locate the **persistent-user-config** configuration item.
3. Click **Update** to edit **lightweight-user-config.yaml** and add **- --target-response-auto-auth=true** under **operatorConfigOverride**.


```
customSettings:
  operatorEnvOverride: []
  operatorConfigOverride:
  - --target-response-auto-auth=true
  promAdapterConfigOverride: []
```

4. Click **OK** to save the configuration. The configuration takes effect in about 1 minute.

10.4.4 Cluster Monitoring

To observe the resource usage and health of a cluster, choose **Monitoring Center** > **Clusters**. The monitoring data is displayed, where you can view the **Cluster Health**, **Health Overview**, **Top Resource Consumption Statistics**, and **Data Plane Monitoring**.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

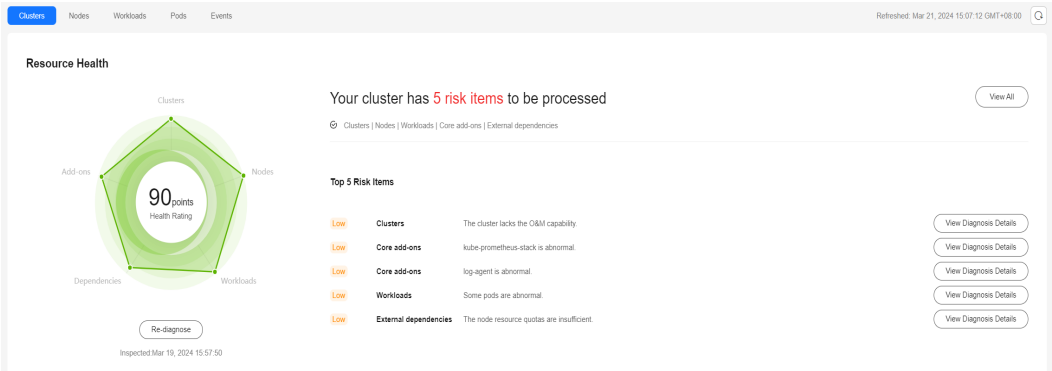
Step 2 In the navigation pane, choose **Monitoring Center**. Then, click the **Clusters** tab.

----End

Cluster Health

Cluster health is evaluated from several dimensions, such as the health score, number of risk items to be processed, risk level, and proportion of diagnosed risk items for master nodes, clusters, worker nodes, workloads, and external dependencies. Abnormal data is displayed in red. For more diagnosis results, go to **Health Center**.

Figure 10-14 Cluster health



Health Overview

Resource Overview

Resource Overview displays the percentage of abnormal resources in nodes, workloads, and pods and the total number of namespaces.

Control Plane Health Overview

Control Plane Health Overview displays the percentage of exceptions on control plane components and master nodes, total QPS of the API server, and request

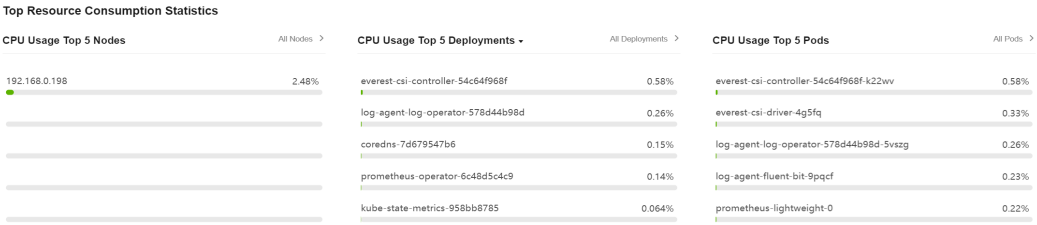
error rate of the API server. If the API server (the API service provider of the cluster) on the control plane is abnormal, the cluster may fail to be accessed, and workloads that depend on the API server may fail to run normally. The QPS and request error rate help you quickly identify and rectify faults.

Figure 10-15 Health overview

Top Resource Consumption Statistics

CCE collects statistics on top 5 nodes, Deployments, StatefulSets, and pods by CPU and memory usages, helping you identify high resource consumption. To view all data, click the **nodes**, **workloads**, or **pods** tab.

Figure 10-16 Top Resource Consumption Statistics



Metrics

- CPU Usage
  - Node CPU usage = Average percentage of the time that the node CPU is not in an idle state
  - Workload CPU usage = Average CPU usage in each pod of the workload
  - Pod CPU usage = CPU cores used by the pod/CPU core limits of all service containers in the pod × 100% (Total CPU cores of the node will be used if no limits are configured.)
- Memory Usage
  - Node memory usage = Memory used by the node/Total memory of the node × 100%
  - Workload memory usage = Average memory usage in each pod of the workload
  - Pod memory usage = The physical memory used by the pod/Memory limits of all containers in the pod × 100% (Total memory of the node will be used if no limits are configured.)

Data Plane Monitoring

By default, the resource usage is collected from each dimension in the last hour, last 8 hours, and last 24 hours. To view more monitoring information, click **View All Metrics** to access the **Dashboard** page. For details, see **Using Dashboard**.

NOTE

You can hover over a chart to view the monitoring data in each minute.

- **CPU**: the CPU used by a cluster in a specified period
- **Memory**: the memory used by a cluster in a specified period
- **PVC Storage Status**: the binding between PVCs and PVs
- **Pod Status and Quantity**: real-time status and number of pods in a cluster
- **Trend of Total Pod Restarts**: the total number of pod restarts in the cluster in the last 5 minutes
- **Node Status Trend**: real-time status of nodes in a cluster

10.4.5 Node Monitoring

To monitor the resource usage of nodes, choose **Monitoring Center** > **Nodes**. This tab provides information about all nodes in a cluster and monitoring data of a single node, such as CPU and memory usages, network inbound and outbound rates, and disk I/O read and write rates.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Monitoring Center**. Then, click the **Nodes** tab.

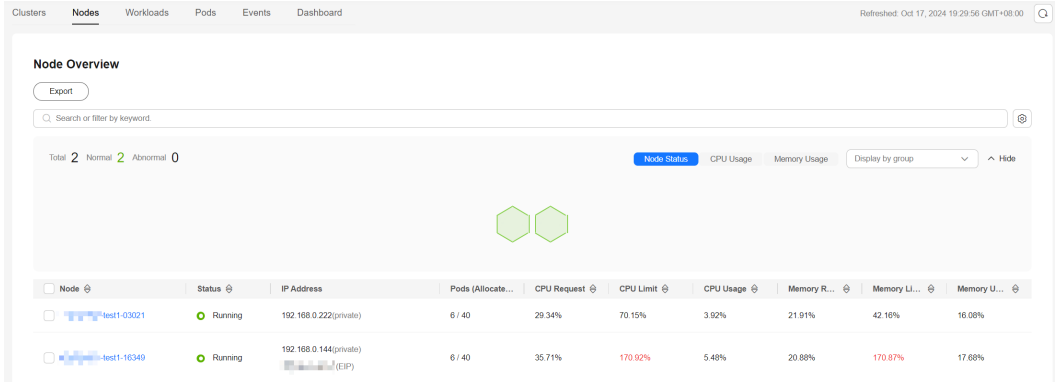
Nodes displays information about all nodes. To view the monitoring information of a node, click the node name to access the **Overview** tab and switch to the **Pods** and **Monitoring** tabs as needed.

----End

Nodes

This tab displays the name, status, IP address, number of pods (allocated/total), CPU request/limit/usage, and memory request/limit/usage of each node.

Figure 10-17 Nodes

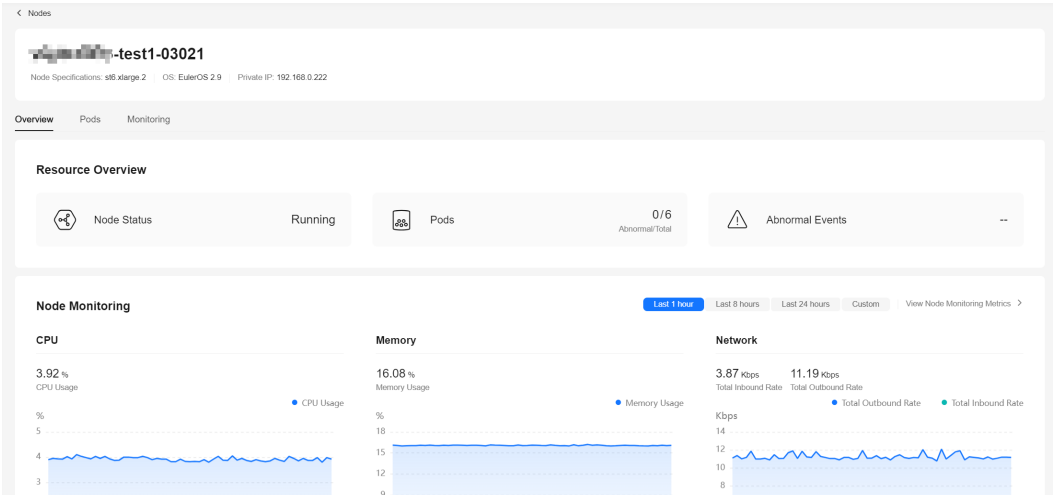


You can search for the desired node by name, status, private IP address, or public IP address. You can click **Export** to export data of all nodes or selected nodes. The exported file is in .xlsx format, and the file name contains the timestamp.

If the CPU limit or memory limit of a node exceeds 100%, the node resources are overcommitted and the sum of workload limits (maximum available values) of the node exceeds the node specifications. If workloads require too many resources, they may preempt resources, causing service exceptions or even node exceptions.

Overview

Figure 10-18 Resource overview and node monitoring



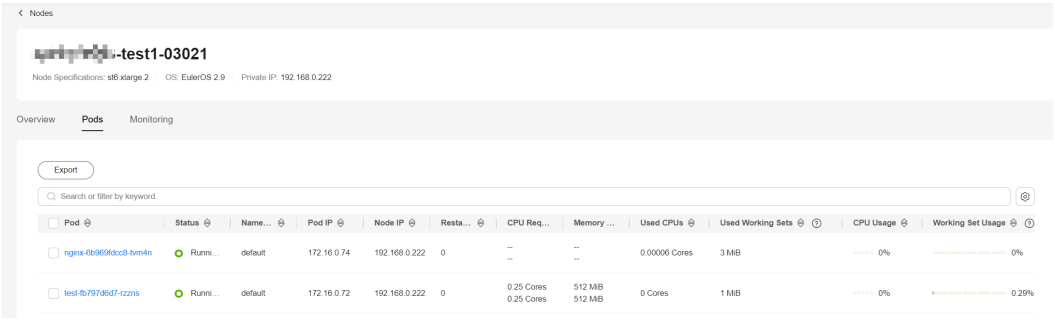
- **Resource Overview**: displays the node status, the number of pods, and abnormal events.
- **Node Monitoring**: displays the monitoring overview of the last hour, including the CPU usage, memory usage, and network inbound and outbound rates.
- **Pod Usage Trend**: displays the resource metrics of each pod on the node. It also displays the top 5 data in descending or ascending order.

For more metrics, go to the **Monitoring** tab.

Pods

This tab displays the name, status, namespace, IP address, node, number of restarts, and resource metrics of each pod.

Figure 10-19 Pods



You can search for the desired pod by name, status, namespace, IP address, or node. You can click **Export** to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **Pod Monitoring**.

Monitoring

This tab displays the resource usage of the node in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access **Dashboard**. For details, see **Using Dashboard**.

Figure 10-20 Node monitoring

- **CPU Metrics**
  - CPU usage: the average percentage of the time that the node CPU is not in an idle state
  - CPU allocation ratio: CPU cores requested by all containers on the node/ Total CPU cores on the node × 100%
  - Single CPU core usage: the percentage of the time that each node CPU core is not in an idle state

- **Memory Metrics**
  - Memory usage: The memory used by the node/Total memory of the node × 100%
  - Memory allocation ratio: The memory requested by all containers on the node/Total memory of the node × 100%
- **Networking Metrics**
  - Outbound rate: the number of bytes sent by the NIC on the node per second during different time periods
  - Inbound rate: the number of bytes received by the NIC on the node per second during different time periods
  - Packet loss (transmit): The packets not received by the receiver/The packets sent from the NIC of the node × 100%
  - Packet loss (receive): The packets not received by the NIC of the node/The packets sent to the NIC × 100%
- **Disk Metrics**
  - Disk read rate: the number of bytes read from each file system on the node per second during different time periods.
  - Disk write rate: the number of bytes written to each file system on the node per second during different time periods.
  - Disk usage: the percentage of used disk space of each file system on the node during different time periods.
- **Pod Metrics**
  - Pod CPU usage: The CPU used by each pod on the node during different time periods/The CPU limit of each pod × 100%
  - Pod memory usage: The memory used by each pod on the node during different time periods/The memory limit of each pod × 100%
  - Pod status and quantity: the total number of pods in the **Unavailable**, **Unready**, **Running**, **Completed**, or **Other** state on the node during different time periods
  - Pod quantity trend: the number of pods on the node during different time periods
- **Other Metrics**
  - Average node load: the average number of running processes on the node in a specified period. This metric is used to check whether the number of processes running on the node reaches its processing capability. Generally, it should be kept within a reasonable range for stability and reliability of the node.
  - iptables connections: the maximum number of entries and the number of allocated entries in the connection tracking table

10.4.6 Workload Monitoring

To monitor the resource usage of workloads, choose **Monitoring Center** > **Workloads**. This tab provides information about all workloads in a cluster and monitoring data of a single workload, such as CPU and memory usages and network inbound and outbound rates.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Monitoring Center**. Then, click the **Workloads** tab.

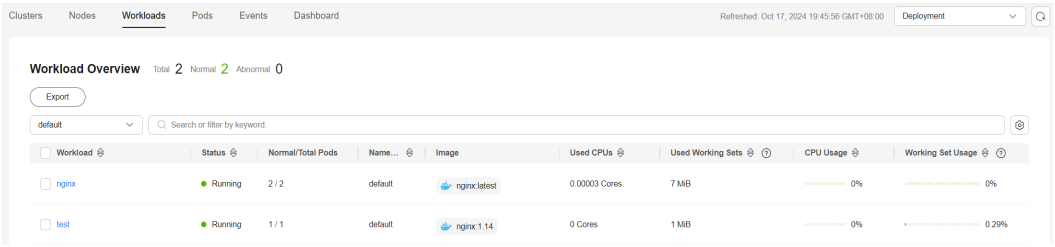
Workloads displays information about all workloads. To view the monitoring information of a workload, click the workload name to access the **Overview** tab and switch to the **Pods** and **Monitoring** tabs as needed.

----End

Workloads

This tab displays the name, status, number of pods (normal/total), namespace, image name, and resource metrics of each workload.

Figure 10-21 Workloads



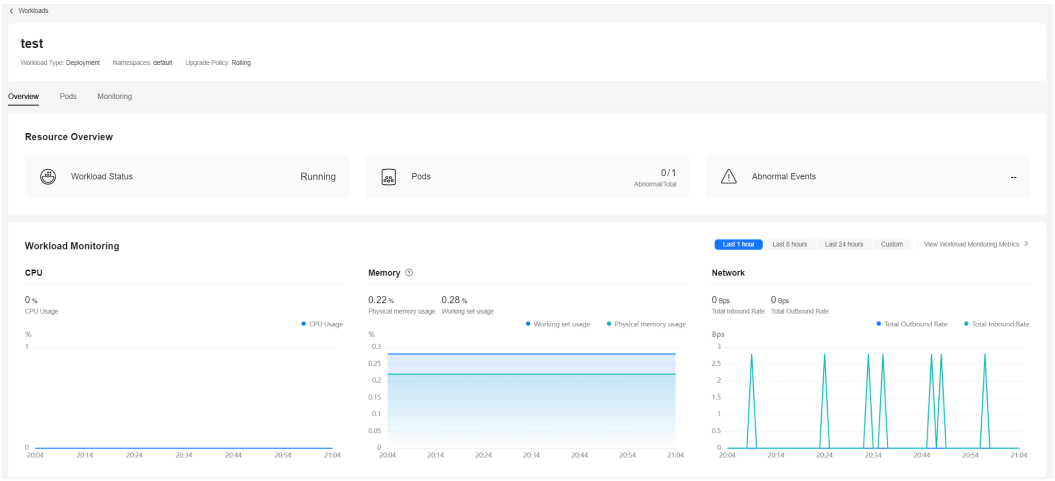
You can search for the desired workload by workload type (in the upper right corner), namespace, and workload name or status.

You can click **Export** to export data of all workloads or selected workloads. The exported file is in .xlsx format, and the file name contains the timestamp.

Overview

You can click the workload name to view the resource overview, including the workload status, number of pods (abnormal/total), and abnormal events. You can also view the monitoring overview of the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 10-22 Resource overview and workload monitoring



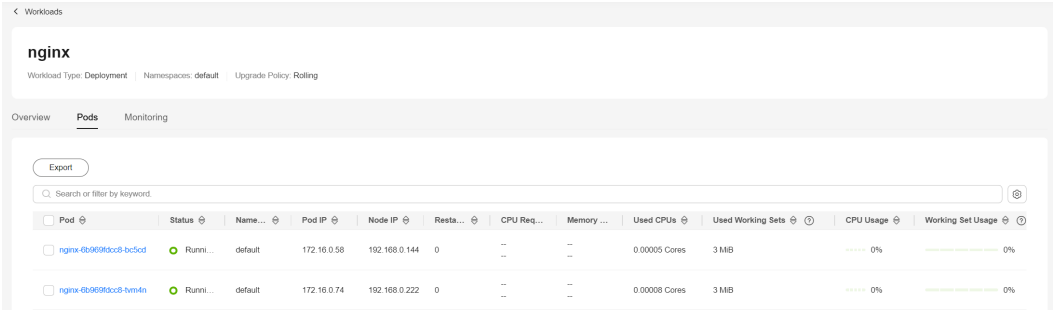
The **Overview** tab also displays the pod usage trend. You can view the resource metrics of each pod of the workload. You can also view the top 5 data in descending or ascending order.

For more metrics, go to the **Monitoring** tab.

Pods

This tab displays the name, status, namespace, IP address, node, number of restarts, and resource metrics of each pod.

Figure 10-23 Pods



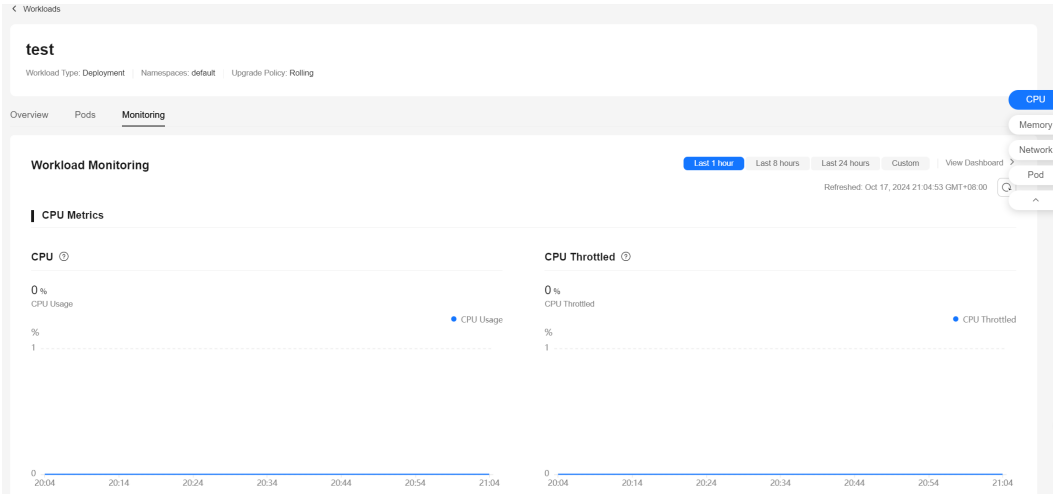
You can search for the desired pod by name, status, namespace, IP address, or node. You can click **Export** to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **Pod Monitoring**.

Monitoring

This tab displays the resource usage of the workload in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access **Dashboard**. For details, see **Using Dashboard**.

Figure 10-24 Workload monitoring



- **CPU Metrics**
  - CPU usage: The CPU used by containers in all pods of the workload during different time periods/CPU limits of all containers × 100%
  - CPU throttled: the average percentage of the time that containers have been throttled in all pods of the workload during different time periods
- **Memory Metrics**
  - Memory usage: The memory used by containers in all pods of the workload during different time periods/Memory limits of all containers × 100%
- **Networking Metrics**
  - Total outbound rate: the total number of bytes sent by containers in all pods of the workload per second during different time periods
  - Total inbound rate: the total number of bytes received by containers in all pods of the workload per second during different time periods
  - Packet loss (transmit): The packets not received by the receiver/The packets sent from containers in all pods of the workload during different time periods × 100%
  - Packet loss (receive): The packets not received by containers in all pods of the workload/The packets sent to the containers during different time periods × 100%
- **Pod Metrics**
  - Pod CPU usage: The CPU used by each pod of the workload during different time periods/The CPU limit of each pod × 100%
  - Pod memory usage: The memory used by each pod of the workload during different time periods/The memory limit of each pod × 100%
  - Pod status and quantity: the total number of pods in the **Unavailable**, **Unready**, **Running**, **Completed**, or **Other** state of the workload during different time periods
  - Pod quantity trend: the number of pods (replicas) of the workload during different time periods

10.4.7 Pod Monitoring

To monitor the resource usage of pods, choose **Monitoring Center** > **Pods**. This tab provides information about all pods in a cluster and monitoring data of a single pod, such as CPU and memory usages and network inbound and outbound rates.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Monitoring Center**. Then, click the **Pods** tab.

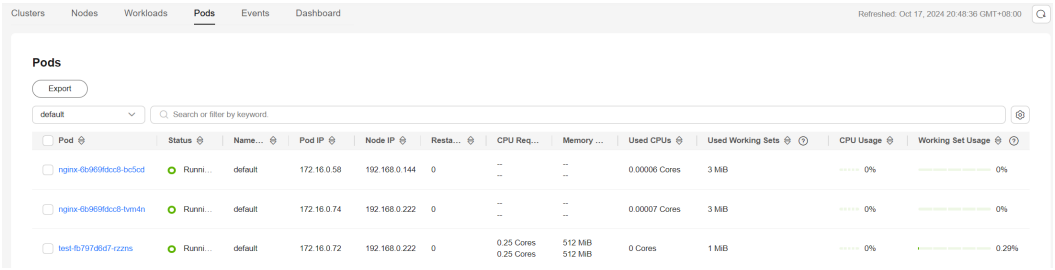
Pods displays information about all pods. To view the monitoring information of a pod, click the pod name to access the **Overview** tab and switch to the **Containers** and **Monitoring** tabs as needed.

----End

Pods

This tab displays the name, status, namespace, IP address, node, number of restarts, and resource metrics of each pod.

Figure 10-25 Pods



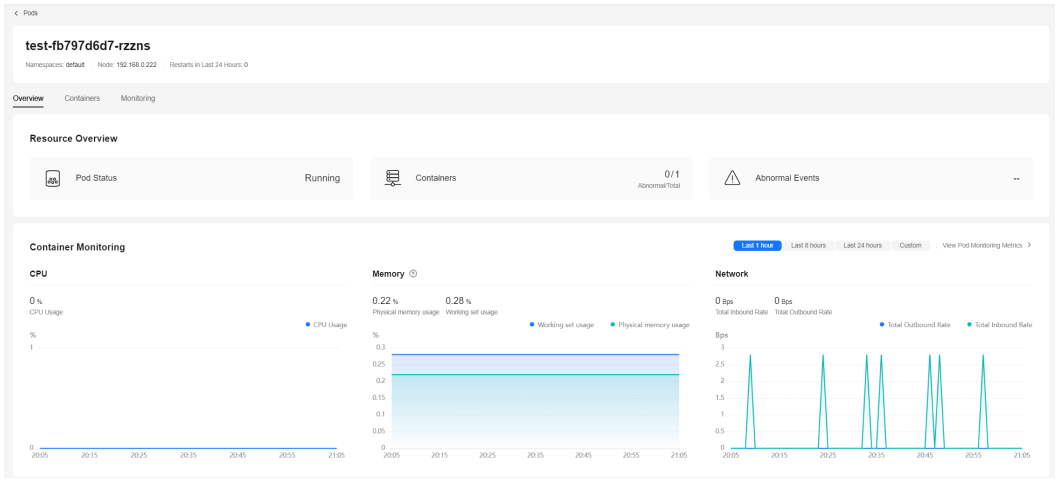
You can search for the desired pod by namespace and pod name, status, pod IP address, or node.

You can click **Export** to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

Overview

You can click the pod name to view the resource overview, including the pod status, number of containers (abnormal/total), and abnormal events. You can also view the monitoring overview of the pod and its node in the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 10-26 Resource overview and container monitoring



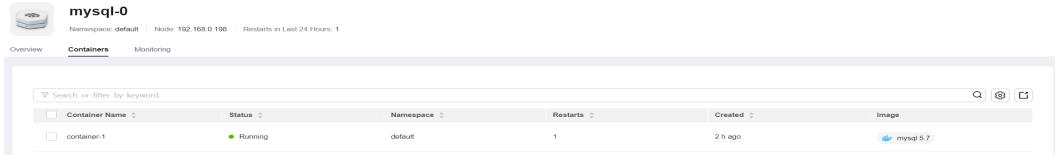
The **Overview** tab also displays the container usage trend. You can view the resource metrics of each container in the pod. You can also view the top 5 data in descending or ascending order.

For more metrics, go to the **Monitoring** tab.

Containers

You can view the name, status, namespace, number of restarts, creation time, and image of each container.

Figure 10-27 Containers

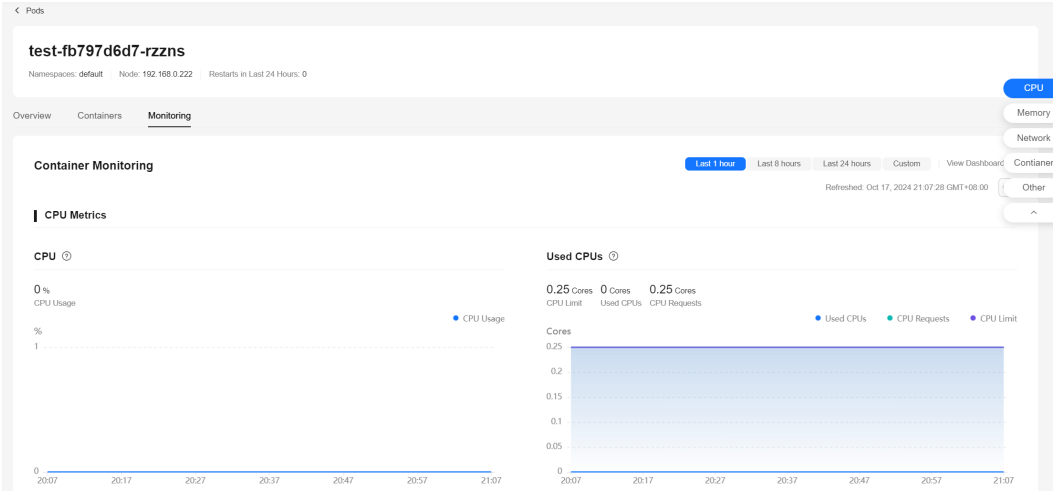


You can search for the desired container by name, status, or namespace. You can click **Export** to export data of all containers or selected containers. The exported file is in .xlsx format, and the file name contains the timestamp.

Monitoring

This tab displays the resource metrics of a pod from the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access **Dashboard**. For details, see **Using Dashboard**.

Figure 10-28 Pod monitoring



- **CPU Metrics**
  - CPU usage: The CPU used by all containers in the pod during different time periods/CPU limits of all containers × 100%
  - Used CPU: the CPU that the pod is using
  - CPU request: the CPU requested by the pod
  - CPU limit: the CPU limit of the pod. When the used CPU is close to this limit, the CPU usage of the containers will be limited, affecting container performance.
- **Memory Metrics**
  - Memory usage: The memory used by all containers in the pod during different time periods/Memory limits of all containers × 100%
  - Used memory: the memory that the pod is using
  - Memory request: the memory requested by the pod
  - Memory limit: the memory limit of the pod. When the used memory is close to this limit, OOM will occur.
- **Networking Metrics**
  - Total outbound rate: the total number of bytes sent by all containers in the pod per second
  - Total inbound rate: the total number of bytes received by all containers in the pod per second
- **Container Metrics**
  - Container CPU usage: The CPU used by each container in the pod during different time periods/The CPU limit of each container × 100%
  - Container memory usage: The memory used by each container in the pod during different time periods/The memory limit of each container × 100%
  - Container CPU throttled: the percentage of the time that each container has been throttled during different time periods
  - Container network packet loss: The packets not received by each container in the pod/The packets sent to each container during different time periods × 100%

- **Other Metrics**
  - Historical pod status: the status of the pod during different periods
  - Historical container status: the status of each container in the pod during different time periods

10.4.8 Event Monitoring

Kubernetes events show the cluster running status and resource scheduling status, helping O&M personnel observe resource changes and locate faults. To enable this function, you need to install the log-agent add-on in the cluster. log-agent can collect Kubernetes events and display them on the **Monitoring Center** > **Events** page.

Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Monitoring Center**. Then, click **Events**.

There are **Overview** and **Events** tabs on **Events**. On the **Overview** tab, you can view the total number, trend, and sorting of events in the cluster. On the **Events** tab, you can view event details, such as the event name, type, content, and information about the resource that triggers the event.

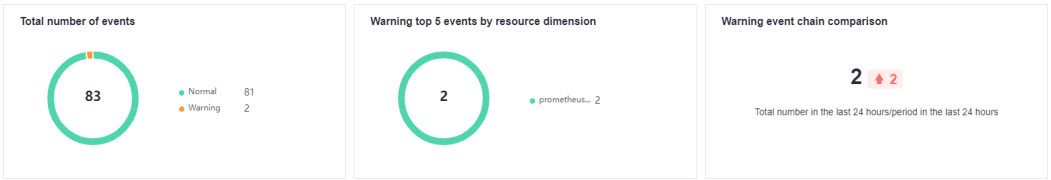
----End

Overview

By default, the **Overview** tab displays the event statistics of all namespaces in the cluster. You can also select a specified namespace from the drop-down list in the upper right corner to view its event data.

Figure 10-29 displays the **Total Events** doughnut chart that shows the distribution of normal and warning events, the **Top 5 Warning Events by Resource** chart that shows the resource information corresponding to the top 5 warning events, and the **Warning Events by Resource Type** chart that shows the comparison between the number of warning events and the number of warning events in the last 24 hours.

Figure 10-29 Event statistics



The bar charts in **Figure 10-30** display the trend of the number of normal events and warning events in the last 24 hours.

Figure 10-30 Warning/Normal event trend

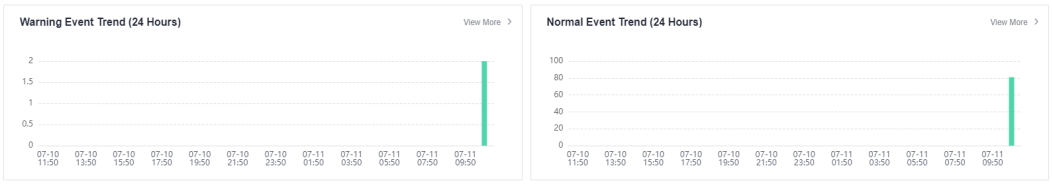
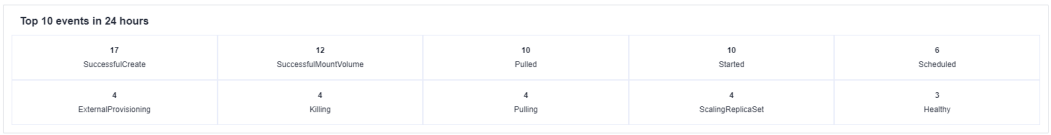


Figure 10-31 displays the names of top 10 events in the last 24 hours.

Figure 10-31 Top 10 events in 24 hours



Events

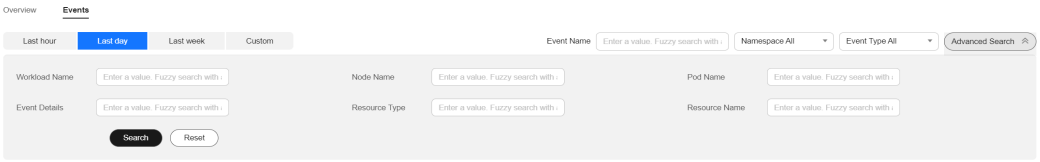
Searching for Events

On the **Events** tab, you can search for a specified resource based on certain criteria to conveniently view its event information, including the trend and details of normal and warning events.

Search for events in any of the following ways:

- Enter the name of the event to be searched for in the text box, select a namespace or event type, and click **Search**.
- Click **Advanced Search** and enter the desired workload, node, pod, event content, resource type, or resource name.
- Select a time interval in the upper left corner to view the events generated in that period, including last hour, last day, last week, and a custom interval.

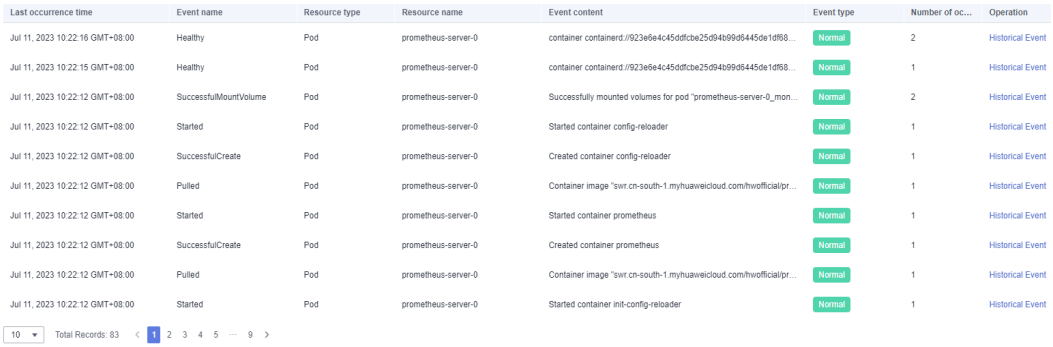
Figure 10-32 Searching for events



Event List

You can view details about events that meet your search criteria in the list. The details include the last occurrence time, event name, resource type, resource name, event content, event type, and occurrence times. Click **Historical Events** in the **Operation** column. A dialog box is displayed to show all events of the current resource type and resource.

Figure 10-33 Event list



Last occurrence time	Event name	Resource type	Resource name	Event content	Event type	Number of oc...	Operation
Jul 11, 2023 10:22:16 GMT+08:00	Healthy	Pod	prometheus-server-0	container containerd://923e6e4c45ddfcbe25d94b99d6445de1df68...	Normal	2	Historical Event
Jul 11, 2023 10:22:15 GMT+08:00	Healthy	Pod	prometheus-server-0	container containerd://923e6e4c45ddfcbe25d94b99d6445de1df68...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulMountVolume	Pod	prometheus-server-0	Successfully mounted volumes for pod "prometheus-server-0_mon...	Normal	2	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container config-reloader	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulCreate	Pod	prometheus-server-0	Created container config-reloader	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Pulled	Pod	prometheus-server-0	Container image "swr.cn-south-1.myhuaweicloud.com/hwofficial/pr...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container prometheus	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	SuccessfulCreate	Pod	prometheus-server-0	Created container prometheus	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Pulled	Pod	prometheus-server-0	Container image "swr.cn-south-1.myhuaweicloud.com/hwofficial/pr...	Normal	1	Historical Event
Jul 11, 2023 10:22:12 GMT+08:00	Started	Pod	prometheus-server-0	Started container init-config-reloader	Normal	1	Historical Event

10 Total Records: 83 1 2 3 4 5 ... 9

10.4.9 Dashboard

10.4.9.1 Using Dashboard

A dashboard integrates high-frequency monitoring metrics of different components from different perspectives. Different metrics are displayed on the same screen in charts, helping you monitor the cluster running in real time.

The dashboard displays monitoring metrics in various views such as the cluster view, API server view, pod view, host view, and node view.

Prerequisites

- The cluster version is v1.17 or later.
- The cluster is in the **Running** state.
- Monitoring Center has been enabled for the cluster.

Checking and Switching Views

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Monitoring Center**. Then, click the **Dashboard** tab.

The cluster view is displayed by default.

Step 3 The dashboard provides preset views. You can click the **Switch View** button next to the view name to select monitoring data to view.

Step 4 Configure related parameters for checking views.

Step 5 Specify the view window.

Select or customize time segments in the upper right corner of the page, and click to refresh the page.

----End

10.4.9.2 Cluster View

A cluster view is generated based on metrics and Prometheus Query Language (PromQL) statements. It displays information about nodes, CPUs, memory,

networks, and disks, so you can better monitor a cluster. The following describes cluster resources from two parts: metric description and metric list. In the resource diagrams, greater values in bytes can be converted to ones in MB, KB, or GB.

Metric Description

Cluster metrics include basic metrics, network metrics, and disk metrics. You can view these metrics in the following tables.

Figure 10-34 Basic metrics

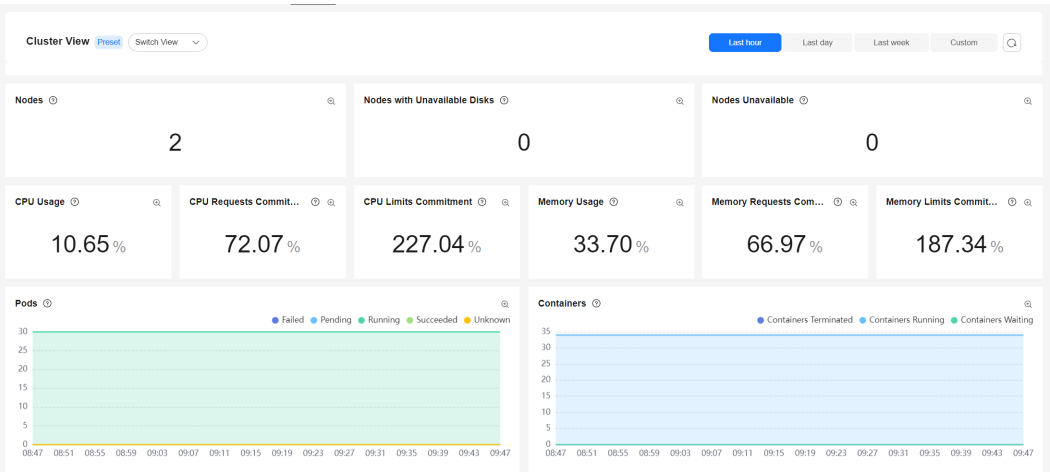


Table 10-6 Basic metrics

Metric	Unit	Description
Nodes	N/A	The number of nodes in a cluster.
Nodes with Unavailable Disks	N/A	The number of nodes with unavailable disks in a cluster.
Nodes Unavailable	N/A	The number of unready nodes in a cluster.
CPU Usage	%	Total CPUs used by all containers/CPU limits in the cluster
CPU Requests Commitment	%	Total CPU requests/Total CPU of the cluster
CPU Limits Commitment	%	Total CPU limits/Total CPU of the cluster
Memory Usage	%	Total memory used by all containers/Memory limits in the cluster
Memory Requests Commitment	%	Total memory requests/Total memory of the cluster

Metric	Unit	Description
Memory Limits Commitment	%	Memory limits/Total memory of the cluster
Pods	N/A	The number of pods in different states (**Failed**, **Pending**, **Running**, **Succeeded**, and **Unknown**) in a cluster.
Containers	N/A	The number of containers in different states (**Running**, **Waiting**, and **Terminated**) in a cluster.
Used CPU	core	Total CPUs used by all containers in each namespace.
Used Memory	byte	Total memory used by all containers in each namespace.

Figure 10-35 Network metrics

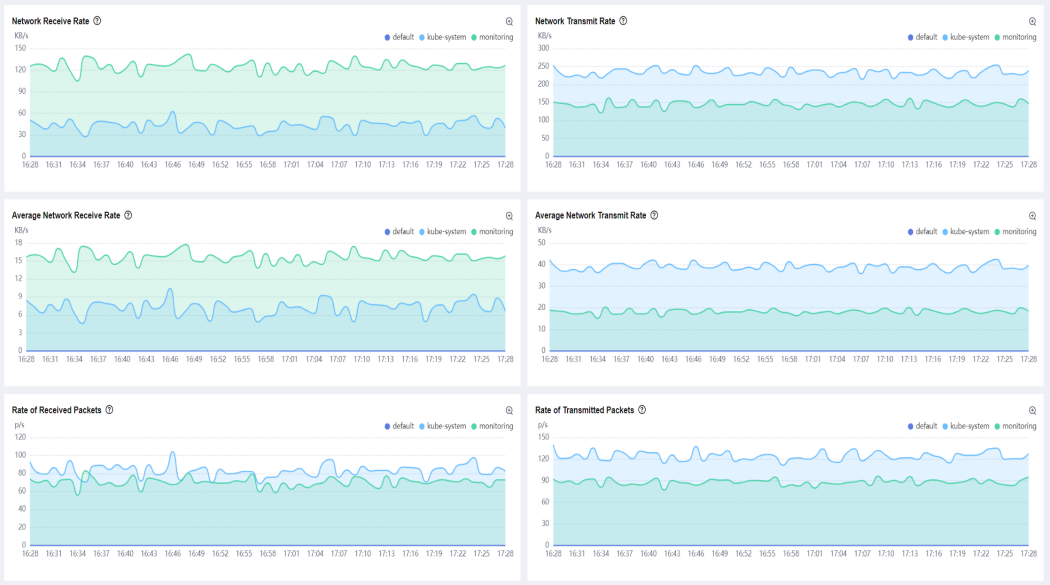


Table 10-7 Network metrics

Metric	Unit	Description
Network Receive Rate	byte/s	Total number of bytes received by all containers in each namespace per second.
Network Transmit Rate	byte/s	Total number of bytes sent by all containers in each namespace per second.

Metric	Unit	Description
Average Network Receive Rate	byte/s	Average number of bytes received by all containers in each namespace per second.
Average Network Transmit Rate	byte/s	Average number of bytes sent by all containers in each namespace per second.
Packet Receive Rate	N/A	Total number of packets received by all containers in each namespace per second.
Packet Transmit Rate	N/A	Total number of packets sent by all containers in each namespace per second.
Packet Loss (Receive)	N/A	Total number of packets not received by all containers in each namespace per second.
Packet Loss (Transmit)	N/A	Total number of packets sent from all containers but not received by the receiver in each namespace per second.

Figure 10-36 Disk metrics

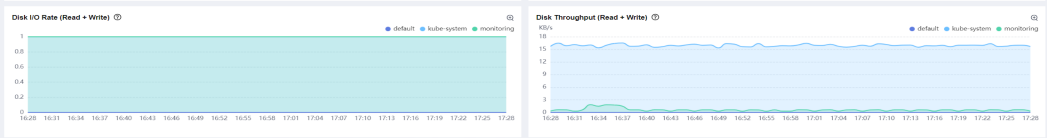


Table 10-8 Disk metrics

Metric	Unit	Description
Disk I/O Rate (Read + Write)	N/A	Total number of read and write operations on the disk by all containers in each namespace per second.
Disk Throughput (Read + Write)	byte/s	Total number of bytes read from and written to the disk by all containers in each namespace per second.

Metric List

The following is the metric list of the cluster view.

Table 10-9 Metric description

Metric	Type	Description
kube_pod_container_resource_requests	Gauge	Resource requests of a container.
kube_pod_container_resource_limits	Gauge	Resource limits of a container.
kube_node_status_allocatable	Gauge	Resources allocatable on a node.
kube_pod_status_phase	Gauge	The current phase of a pod.
node_memory_MemAvailable_bytes	Gauge	The amount of memory available on a node.
node_memory_MemTotal_bytes	Gauge	Cumulative count of bytes that can be consumed by a node.
node_cpu_seconds_total	Counter	Seconds the CPUs spent in each mode.
kube_node_info	Gauge	Node information.
kube_node_status_condition	Gauge	Node status.
kube_pod_container_status_waiting	Gauge	Whether the container is in the **waiting** state.
kube_pod_container_status_terminated	Gauge	Whether the container is in the **terminated** state.
container_cpu_usage_seconds_total	Counter	Cumulative CPU time consumed by a container, in seconds.
container_memory_rss	Gauge	Resident set size (RSS), which is the amount of space of physical memory (RAM) held by a process.
container_network_receive_bytes_total	Counter	Cumulative count of bytes received by a container.
container_network_transmit_bytes_total	Counter	Cumulative count of bytes sent by a container.
container_network_receive_packets_total	Counter	Cumulative count of packets received by a container.
container_network_transmit_packets_total	Counter	Cumulative count of packets sent by a container.
container_network_receive_packets_dropped_total	Counter	Cumulative count of packets not received by a container.

Metric	Type	Description
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets sent from a container but not received by the receiver.
container_fs_reads_total	Counter	Cumulative count of reads completed by a container.
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read by a container.

10.4.9.3 API Server View

The API server view displays information about requests, work queues, and resources, so you can better monitor the API server.

Metric Description

API server metrics include request metrics, work queue metrics, and resource metrics. You can view these metrics in the following tables.

Figure 10-37 Request metrics

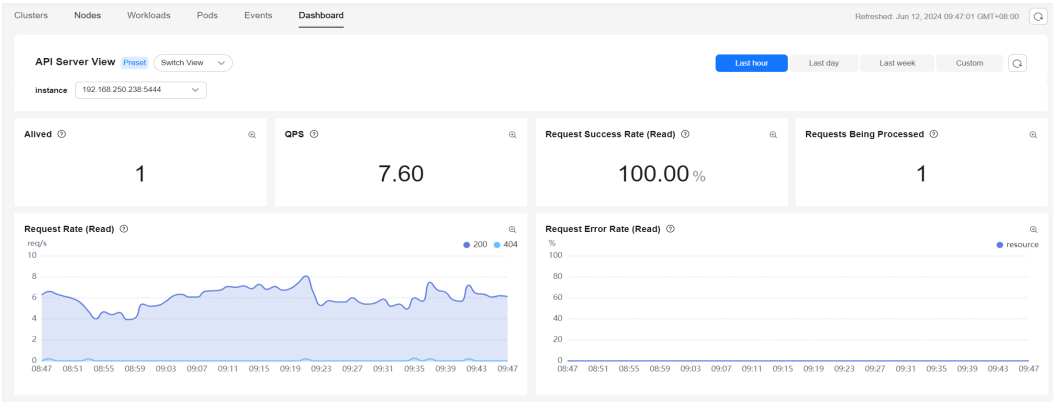


Table 10-10 Request metrics

Metric	Unit	Description
Alived	N/A	The number of live component instances.
QPS	request/s	The number of requests with different response codes per second.
Request Success Rate (Read)	%	The percentage of read requests whose response code is 20x per second.
Requests Being Processed	N/A	The number of requests being processed by the API server.

Metric	Unit	Description
Request Rate (Read)	request/s	The number of read requests with different response codes per second.
Request Error Rate (Read)	%	The percentage of read requests that are rejected per second.
P99 Request Latency (Read)	ms	P99 latency of read operations.
Request Rate (Write)	request/s	The number of write requests with different response codes per second.
Request Error Rate (Write)	%	The percentage of write requests that are rejected per second.
P99 Request Latency (Write)	ms	P99 latency of write operations.

Figure 10-38 Work queue metrics

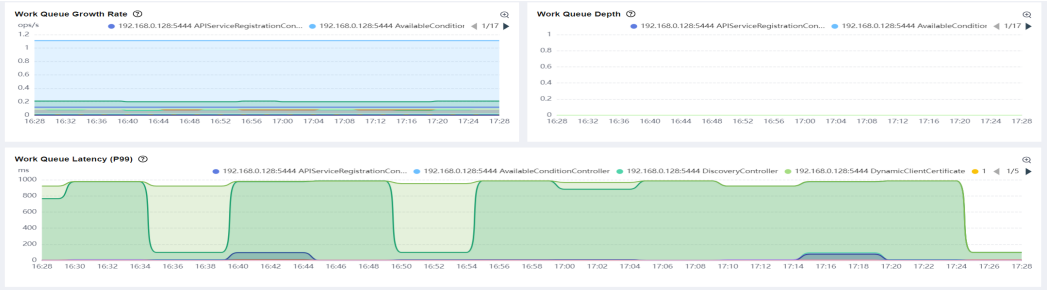


Table 10-11 Work queue metrics

Metric	Unit	Description
Work Queue Growth Rate	N/A	The number of additions handled by the API server work queue per second.
Work Queue Depth	N/A	Depth of a work queue.
Work Queue Latency (P99)	ms	P99 latency of each API server request in a work queue.

Figure 10-39 Resource metrics

Table 10-12 Resource metrics

Metric	Unit	Description
Used Memory	byte	Memory used by the API server.
Used CPU	core	CPU used by the API server.
Goroutines	N/A	The number of goroutines.

Metric List

The following is the metric list of the API server view.

Table 10-13 Metric description

Metric	Type	Description
up	Gauge	Component status.
apiserver_request_total	Counter	Counter of API server requests broken out for code and other items.
go_goroutines	Gauge	The number of goroutines that currently exist.
apiserver_current_inflight_requests	Gauge	Maximum number of requests that were being actively served in the last one-second window.
apiserver_request_duration_seconds_bucket	Histogram	Latency for each request to the API server in seconds.
workqueue_depth	Gauge	Current depth of a work queue.
workqueue_adds_total	Counter	Total number of additions handled by a work queue.
workqueue_queue_duration_seconds_bucket	Histogram	How long in seconds an item stays in a work queue before being requested.
process_resident_memory_bytes	Gauge	Resident memory size in bytes.
process_cpu_seconds_total	Counter	Total user and system CPU time spent in seconds.

10.4.9.4 Pod View

The pod view displays information about resources, networks, and disks, so you can better monitor a pod.

Metric Description

Pod metrics include resource metrics, network metrics, and disk metrics. You can view these metrics in the following tables.

Figure 10-40 Resource metrics

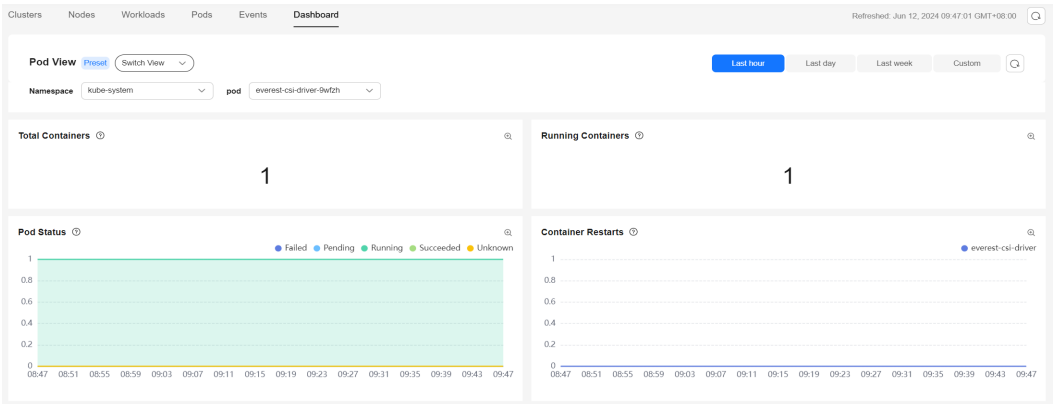


Table 10-14 Resource metrics

Metric	Unit	Description
Total Containers	N/A	Total number of containers in a pod.
Running Containers	N/A	The number of running containers in a pod.
Pod Status	N/A	The number of pods in different states.
Container Restarts	N/A	The number of container restarts.
Used CPU	core	CPU used by the pod.
CPU Efficiency & Usage	%	- Efficiency: Used CPU/Requested CPU - Usage: Used CPU/Total CPU
Used Memory	byte	Memory used by a pod.
Memory Efficiency & Usage	%	- Efficiency: Used memory/Requested memory - Usage: Used memory/Total memory
CPU Throttling	%	The percentage of the CPU time that is in throttling status.

Figure 10-41 Network metrics

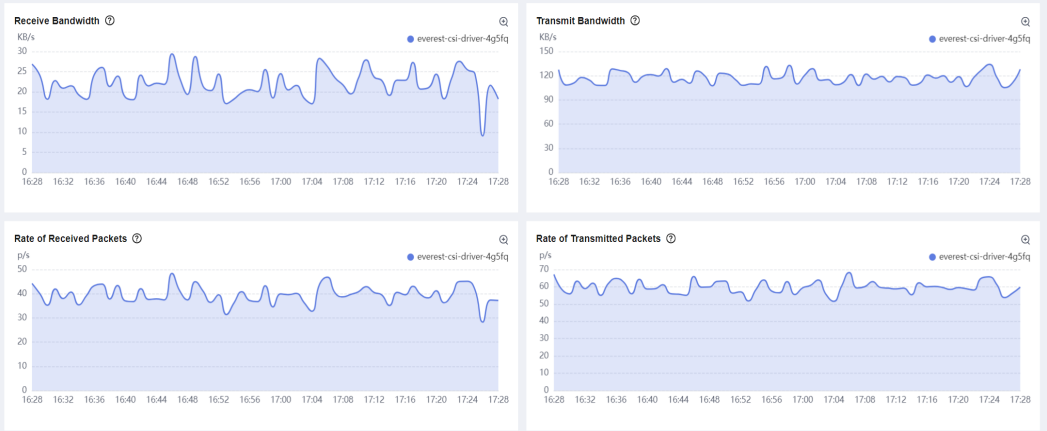


Table 10-15 Network metrics

Metric	Unit	Description
Network Receive Rate	byte/s	The number of bytes received by a container per second.
Network Transmit Rate	byte/s	The number of bytes sent by a container per second.
Packet Receive Rate	N/A	The number of packets received by a container per second.
Packet Transmit Rate	N/A	The number of packets sent by a container per second.
Packet Loss (Receive)	N/A	The number of packets not received by a container per second.
Packet Loss (Transmit)	N/A	The number of packets sent from a container but not received by the receiver per second.

Figure 10-42 Disk metrics

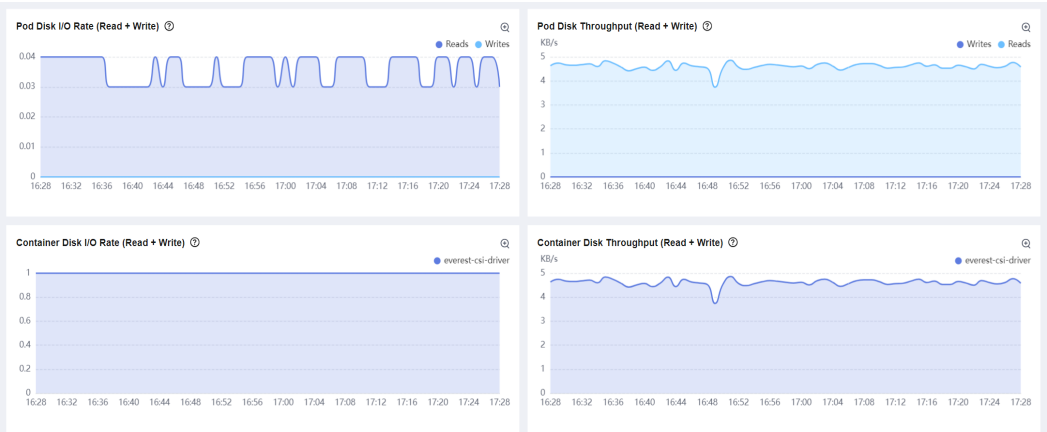


Table 10-16 Disk metrics

Metric	Unit	Description
Pod Disk I/O Rate (Read + Write)	N/A	The number of read/write on the disk by a pod per second.
Pod Disk Throughput (Read + Write)	byte/s	Total number of bytes read to or written from the disk by a pod per second.
Container Disk I/O Rate (Read + Write)	N/A	The number of read/write operations on the disk by a pod per second.
Container Disk Throughput (Read + Write)	byte/s	Total number of bytes read to and written from the disk by the pod per second.
File System Usage	%	The space usage of the file system.
File System Used	byte	The number of bytes used by the file system.

Metric List

The following is the metric list of the pod view.

Table 10-17 Metric description

Metric	Type	Description
kube_pod_container_status_running	Gauge	Whether the container is currently in running state.
kube_pod_container_info	Gauge	Information about a container in a pod.
kube_pod_status_phase	Gauge	The current phase of a pod.
kube_pod_container_status_restarts_total	Counter	The number of container restarts.
container_cpu_usage_seconds_total	Counter	Cumulative CPU time consumed by a container, in seconds.
kube_pod_container_resource_requests	Gauge	The number of request resources requested by a container.
container_spec_cpu_quota	Gauge	CPU quota of a container.
container_memory_working_set_bytes	Gauge	Memory used by a container.
container_spec_memory_limit_bytes	Gauge	Memory limit for a container.

Metric	Type	Description
container_cpu_cfs_throttled_periods_total	Counter	The number of throttled periods.
container_cpu_cfs_periods_total	Counter	The number of elapsed enforcement periods.
container_network_receive_bytes_total	Counter	Cumulative count of bytes received by a container.
container_network_transmit_bytes_total	Counter	Cumulative count of bytes sent by a container.
container_network_receive_packets_total	Counter	Cumulative count of packets received by a container.
container_network_transmit_packets_total	Counter	Cumulative count of packets sent by a container.
container_network_receive_packets_dropped_total	Counter	Cumulative count of packets not received by a container.
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets sent from a container but not received by the receiver.
container_fs_reads_total	Counter	Cumulative count of reads completed by a container.
container_fs_writes_total	Counter	Cumulative count of writes completed by a container.
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read by a container.
container_fs_writes_bytes_total	Counter	Cumulative count of bytes written by a container.
container_fs_usage_bytes	Gauge	The number of bytes that are consumed by a container on the file system.
container_fs_limit_bytes	Gauge	The number of bytes that can be consumed by a container on the file system.

10.4.9.5 Host View

The host view displays information about host resource usage and health status, as well as common metrics of system devices such as disks and file systems, so you can better monitor a node.

Metric Description

You can view the host metrics in the following table.

Figure 10-43 Host metrics

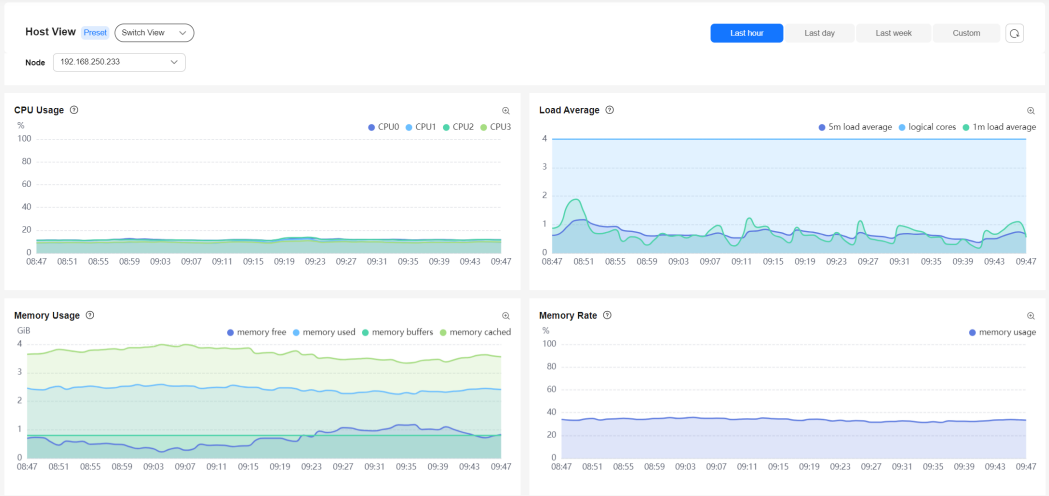


Table 10-18 Host metrics

Metric	Unit	Description
CPU Usage	%	Usage of each CPU core.
Load Average	N/A	CPU contention. • If the value is less than **1**, some CPU resources are used for processing requests. • If the value is **1**, all CPU resources are used for processing requests. • If the value is greater than **1**, some threads are waiting for processing.
Used Memory	byte	Memory usage in each mode.
Memory Usage	%	Host memory usage.
Disk Written	byte/s	Write rate of each disk.
Disk Read	byte/s	Read rate of each disk.
Used Disk Space	byte	Used and available disk space.
Disk Space Usage	%	Disk usage of each device.
Disk I/O Latency	second	Disk I/O read/write latency, in seconds.

Metric	Unit	Description
TCP Connection	N/A	The number of TCP sockets. • **alloc**: the number of TCP sockets allocated (established, and sk_buff obtained). • **inuse**: the number of TCP sockets in use (listening) • **orphan**: the number of TCP sockets not associated with applications. • **tw**: the number of TCP sockets waiting to be closed.
UDP Usage	N/A	UDP usage. • **liteInuse**: the number of UDP-Lite sockets in use. • **inuse**: the number of UDP sockets in use. • **useMemory**: the usage of the UPD buffer.
File Descriptor	exabyte	Maximum number of file descriptors.
Used File Descriptor	N/A	The number of allocated file descriptors.
Socket Usage	N/A	Socket usage. • **socketsUsed**: the total number of used protocol sockets. • **fragInuse**: the number of fragment sockets in use. • **fragMemroy**: the usage of the fragment buffer. • **rawInuse**: the number of raw sockets in use.
Abnormal File System	N/A	File system status. • **readonly**: The file system is read-only. • **deviceError**: A file system error occurred.
Disk I/O Rate	N/A	The number of read/write operations on a disk per second.
Disk Read/Write Latency	second	Disk read/write latency.
I/O Queues	N/A	Average I/O queue length of the disk device, which is the weighted number of seconds spent doing I/O operations. A larger value indicates better disk performance of the node.

Metric	Unit	Description
Process State	N/A	The number of processes in each state.
Connection Tracking Table Entries	N/A	• **Allocated**: the number of allocated entries in the connection tracking table. • **Total**: the maximum number of entries in the connection tracking table.

Metric List

The following is the metric list of the host view.

Table 10-19 Metric description

Metric	Type	Description
node_cpu_seconds_total	Counter	Seconds the CPUs spent in each mode.
node_load1	Gauge	Average CPU load within 1 minute, which reflects the competition of CPU resources. • If the value is less than **1**, some CPU resources are processing requests. • If the value is **1**, all CPU resources are processing requests. • If the value is greater than **1**, some threads are waiting for processing.
node_load15	Gauge	Average CPU load within 15 minutes.
node_memory_MemTotal_bytes	Gauge	Total memory of a node.
node_memory_MemAvailable_bytes	Gauge	Available memory of a node.
node_disk_written_bytes_total	Gauge	Total number of bytes written successfully.
node_disk_read_bytes_total	Gauge	Total number of bytes read successfully.
node_filesystem_size_bytes	Gauge	File system size in bytes.

Metric	Type	Description
node_filesystem_avail_bytes	Gauge	Available file system size in bytes.
node_disk_io_time_seconds_total	Counter	Total seconds spent doing I/O operations.
node_sockstat_TCP_alloc	Gauge	The number of allocated TCP sockets.
node_sockstat_UDPLITE_inuse	Gauge	The number of used UDPLITE sockets.
node_filefd_maximum	Gauge	File descriptor statistics: maximum.
node_filefd_allocated	Gauge	File descriptor statistics: allocated.
node_sockstat_sockets_used	Gauge	The number of used IPv4 sockets.
node_filesystem_readonly	Gauge	Read-only status of the file system.
node_disk_reads_completed_total	Counter	The number of disk reads completed.
node_disk_read_time_seconds_total	Counter	Total number of seconds spent by all reads.
node_disk_io_time_weighted_seconds_total	Counter	The weighted number of seconds spent doing I/O operations. A larger value indicates better disk performance of the node.
node_procs_blocked	Gauge	The number of processes blocked waiting for I/O to complete.
node_nf_conntrack_entries	Gauge	The number of currently allocated flow entries for connection tracking.

10.4.9.6 Node View

The node view displays node resource, network, and disk metrics, so you can better monitor a node.

Metric Description

You can view the node metrics in the following tables.

Figure 10-44 Resource metrics

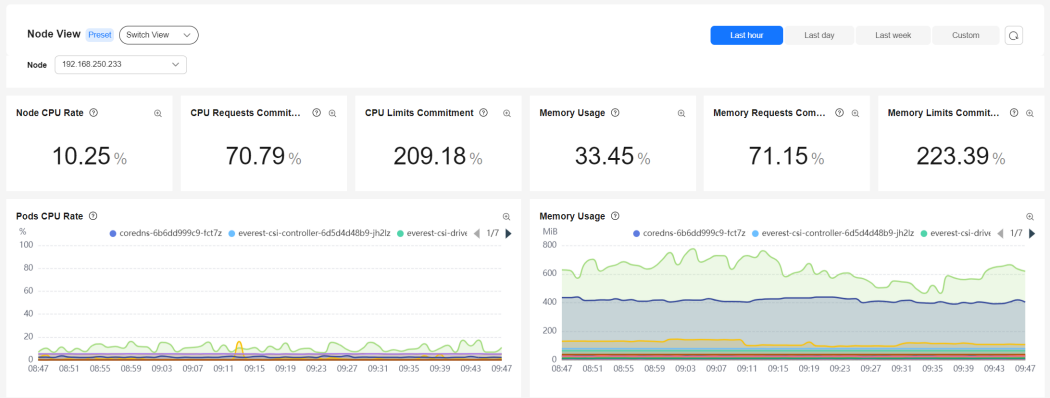


Table 10-20 Resource metrics

Metric	Unit	Description
Node CPU Usage	%	CPU usage of a node.
CPU Requests Commitment	%	Total CPU requests/Total CPU of the node
CPU Limits Commitment	%	CPU limits/Total CPU of the node
Memory Usage	%	Memory usage of a node.
Memory Requests Commitment	%	Total memory requests/Total memory of the node
Memory Limits Commitment	%	Memory limits/Total memory of the node
Pod CPU Usage	%	CPU usage of the pod on the node
Used Memory	byte	Memory usage of the pod on the node

Figure 10-45 Network metrics

Table 10-21 Network metrics

Metric	Unit	Description
Network Receive Rate	byte/s	The number of bytes received by the node per second.
Network Transmit Rate	byte/s	The number of bytes sent by the node per second.
Network Receive Rate (Pod)	byte/s	The number of bytes received by the pod on the node per second.
Network Transmit Rate (Pod)	byte/s	The number of bytes sent by the pod on the node per second.
Packet Receive Rate	N/A	The number of packets received by the pod on the node per second.
Packet Transmit Rate	N/A	The number of packets sent by the pod on the node per second.
Packet Loss (Receive)	N/A	The number of packets not received by the pod on the node per second.
Packet Loss (Transmit)	N/A	The number of packets sent from the pod but not received by the receiver on the node per second.

Figure 10-46 Disk metrics

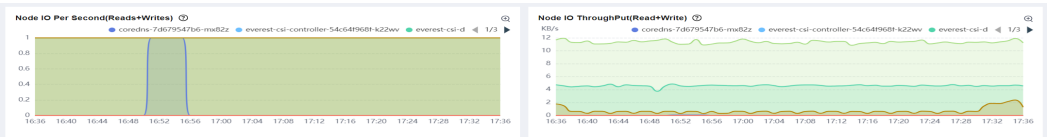


Table 10-22 Disk metrics

Metric	Unit	Description
Node Disk I/O Rate (Read + Write)	N/A	The number of read/write operations on the disk by the pod on the node per second.
Node Disk Throughput (Read + Write)	byte/s	Total number of bytes read from and written to the disk by the pod on the node per second.

Metric List

The following is the metric lis of the node view.

Table 10-23 Metric description

Metric	Type	Description
kube_pod_container_resource_limits	Gauge	The number of requested limit resources by a container.
kube_pod_status_phase	Gauge	The current phase of a pod.
kube_node_status_allocatable	Gauge	The amount of resources allocatable for pods.
kube_pod_container_resource_requests	Gauge	The number of requested request resources by a container.
node_memory_MemAvailable_bytes	Gauge	Number of bytes that can be consumed by a node.
node_memory_MemTotal_bytes	Gauge	Cumulative count of bytes that can be consumed by a node.
node_cpu_seconds_total	Counter	Seconds the CPUs spent in each mode.
container_cpu_usage_seconds_total	Counter	Cumulative CPU time consumed in seconds.
container_memory_rss	Gauge	Resident set size (RSS), which is the amount of space of physical memory (RAM) held by a process.
container_network_receive_bytes_total	Counter	Cumulative count of bytes received by a container.
container_network_transmit_bytes_total	Counter	Cumulative count of bytes sent by a container.
container_network_receive_packets_total	Counter	Cumulative count of packets received by a container.

Metric	Type	Description
container_network_transmit_packets_total	Counter	Cumulative count of packets sent by a container.
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets sent from a container but not received by the receiver.
container_fs_reads_total	Counter	Cumulative count of reads completed by a container.
container_fs_writes_total	Counter	Cumulative count of writes completed by a container.
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read by a container.
container_fs_writes_bytes_total	Counter	Cumulative count of bytes written by a container.

10.4.9.7 Node Pool View

The node pool view displays the usage and allocation of node pool resources, so you can monitor the load of a node pool.

Metric Description

You can view the node pool metrics in the following table.

Figure 10-47 Node pool metrics

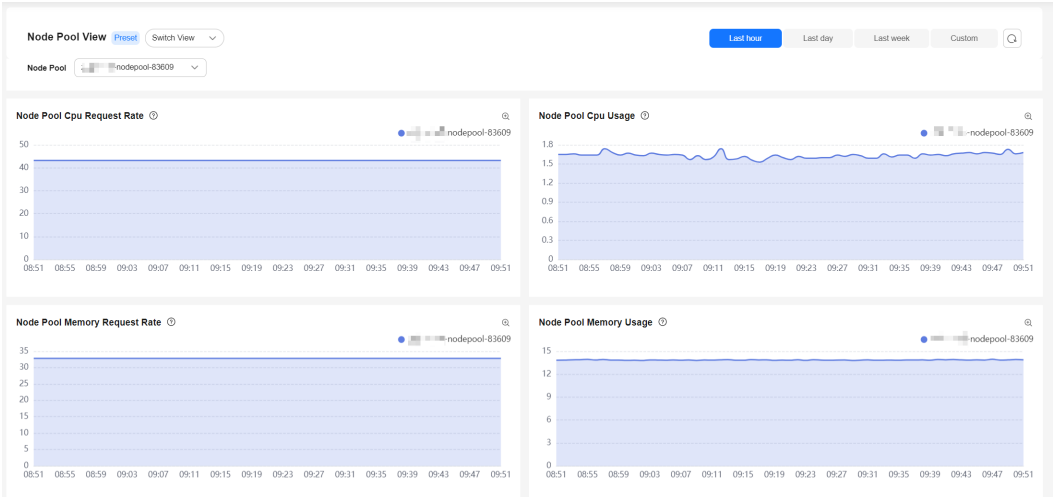


Table 10-24 Node pool metrics

Metric	Unit	Description
Node Pool CPU Allocation Rate	%	Total CPU requests of pods/Total CPUs of all nodes in the node pool
Node Pool CPU Usage	%	Total CPUs used by all nodes/Total CPU in the node pool
Node Pool Memory Allocation Rate	%	Total memory requests of pods/Total memory of all nodes in the node pool
Node Pool Memory Usage	%	Total memory used by all nodes/Total memory in the node pool
Node Count Trend	N/A	The number of nodes in the node pool.

Metric List

The following is the metric lis of the node pool view.

Table 10-25 Metric description

Metric	Unit	Description
kube_node_labels	Gauge	Node label. **label_cce_cloud_com_cce_nodepool** indicates the name of a node pool. If this label does not exist, **Default Pool** is used.
node_cpu_seconds_total	Counter	Seconds the CPUs spent in each mode.
node_memory_MemAvailable_bytes	Gauge	Available memory of a node.
node_memory_MemTotal_bytes	Gauge	Total memory of a node.
kube_pod_container_resource_requests	Gauge	Resource requests of a container.

10.4.9.8 GPU View

GPU resource metrics are used to measure GPU performance and usage, including the GPU usage, temperature, and GPU memory, so you can better monitor the GPU.

Metric Description

Figure 10-48 GPU metrics

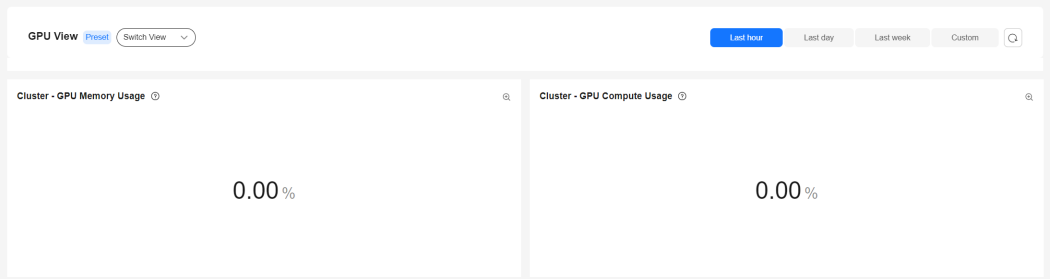


Table 10-26 GPU metrics

Metric	Unit	Description
Cluster - GPU Memory Usage	%	GPU memory usage of the cluster. Formula: Used GPU memory of the cluster/Total GPU memory of the cluster
Cluster - GPU Compute Usage	%	GPU compute usage of the cluster. Formula: Used GPU compute of the cluster/Total GPU compute of the cluster
Node - Used GPU Memory	byte	GPU memory used by the node.
Node - GPU Compute Usage	%	GPU compute usage of each node. Formula: Total GPU compute used by containers on the node/Total GPU compute of the node
Node - GPU Memory Usage	%	GPU memory usage of each node. Formula: Total GPU memory used by containers on the node/Total GPU memory of the node
GPU - Used GPU Memory	byte	Total GPU memory used by containers on the GPU
GPU - GPU Compute Usage	%	GPU compute usage of each graphics card. Formula: Total GPU compute used by containers on the graphics card/Total GPU compute of the graphics card
GPU - Temperature	°C	Temperature of each GPU.
GPU - Memory Clock	Hz	Memory clock of each GPU.

Metric	Unit	Description
GPU - PCIe Bandwidth	byte/s	PCIe bandwidth of each GPU.

Metric List

The following is the metric list of the GPU view.

Table 10-27 Metric description

Metric	Type	Description
cce_gpu_gpu_utilization	Gauge	GPU compute usage.
cce_gpu_memory_utilization	Gauge	GPU memory usage.
cce_gpu_memory_used	Gauge	Used GPU memory.
cce_gpu_memory_total	Gauge	Total GPU memory.
cce_gpu_memory_free	Gauge	Free GPU memory.
cce_gpu_memory_clock	Gauge	The speed at which the GPU memory operates.
cce_gpu_gpu_temperature	Gauge	GPU temperature.
cce_gpu_pcie_link_bandwidth	Gauge	GPU PCIe bandwidth.
cce_gpu_pcie_throughput_rx	Gauge	GPU PCIe RX bandwidth.

10.4.9.9 xGPU View

A GPU can be virtualized into xGPUs. The xGPU view displays the GPU memory and compute allocation rate of xGPUs from multiple perspectives, such as nodes, GPUs, and containers, so you can better monitor the GPU.

Metric Description

Figure 10-49 Resource metrics

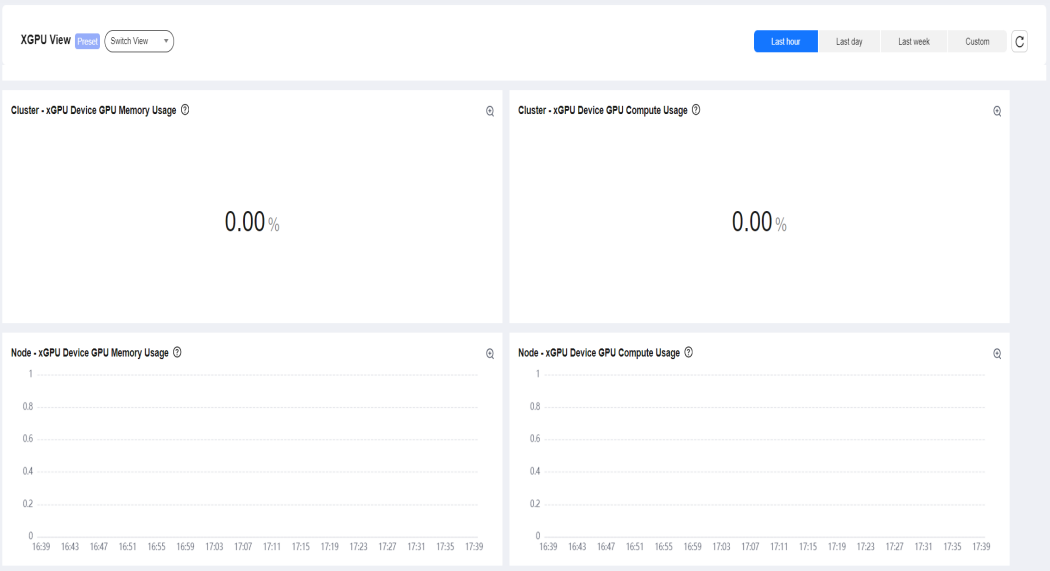


Table 10-28 Resource metrics

Metric	Unit	Description
Cluster - xGPU Device GPU Memory Usage	%	GPU memory usage of all xGPU devices in the cluster. Formula: Used GPU memory of all xGPU devices in the cluster/Total GPU memory of the cluster
Cluster - xGPU Device GPU Compute Usage	%	GPU compute usage of all xGPU devices in the cluster. Formula: Used GPU compute of all xGPU devices in the cluster/Total GPU compute of the cluster
Node - xGPU Device GPU Memory Usage	%	GPU memory usage of all xGPU devices on the node. Formula: Used GPU memory of all xGPU devices on the node/Total GPU memory of the node
Node - xGPU Device GPU Compute Usage	%	GPU compute usage of all xGPU devices on the node. Formula: Used GPU compute of all xGPU devices on the node/Total GPU compute of the node
Node - Number of xGPU Devices	N/A	The number of xGPU devices on the node.

Metric	Unit	Description
Node - Allocated GPU Memory of xGPU Devices	byte	Maximum GPU memory that can be used by all xGPU devices on the node.
GPU - xGPU Device GPU Memory Usage	%	Percentage of used GPU memory of all xGPU devices on the GPU. Formula: Used GPU memory of all xGPU devices on the GPU/Total GPU memory of the GPU
GPU - Allocated GPU Memory of xGPU Devices	byte	Maximum GPU memory that can be used by all xGPU devices on the GPU.
GPU - GPU Memory Allocation Rate of xGPU Devices	%	Percentage of the maximum GPU memory that can be used by all xGPU devices on the GPU. Formula: Maximum GPU memory that can be used by all xGPU devices on the GPU/Total GPU memory of the GPU
GPU - xGPU Device GPU Compute Usage	%	Percentage of the GPU compute used by all xGPU devices on the GPU. Formula: GPU compute used by all xGPU devices on the GPU/Total GPU compute of the GPU
GPU - Number of xGPU Devices	N/A	The number of xGPU devices created on the GPU.
GPU - Scheduling Policy	N/A	• **0**: xGPU memory is isolated and cores are shared. • **1**: Both xGPU memory and cores are isolated. • **2**: default mode, indicating that the current card is not used by any xGPU device for allocation.
GPU - Number of Unhealthy xGPU Devices	N/A	The number of unhealthy xGPU devices on the GPU.
Allocated Container GPU Memory	byte	Maximum GPU memory that can be used by a container.
Container GPU Compute Usage	%	GPU compute used by containers on the xGPU device. Formula: GPU compute used by containers on the xGPU device/Total GPU compute of the xGPU device

Metric	Unit	Description
Used Container GPU Memory	byte	GPU memory used by containers on the xGPU device.
Container GPU Memory Usage	%	GPU compute used by containers on the xGPU device. Formula: GPU memory used by containers on the xGPU device/Total GPU memory of the xGPU device

Metric List

The following is the metric list of the xGPU view.

Table 10-29 Metric description

Metric	Type	Description
xgpu_memory_total	Gauge	Total xGPU memory.
xgpu_memory_used	Gauge	Used xGPU memory.
xgpu_core_percentage_total	Gauge	Total xGPU compute.
xgpu_core_percentage_used	Gauge	xGPU compute usage.
gpu_schedule_policy	Gauge	There are three GPU modes. • **0**: xGPU memory is isolated and cores are shared. • **1**: Both xGPU memory and cores are isolated. • **2**: default mode, indicating that the current card is not used by any xGPU device for allocation.
xgpu_device_health	Gauge	Health of an xGPU device. Currently, the virtualization domain does not provide a specific interface to check the xGPU health. The xGPU health is based on the health of the GPU where the xGPU device is located. The value **0** indicates that the xGPU is healthy, and the value **1** indicates that the xGPU is unhealthy.

10.4.9.10 CoreDNS View

The CoreDNS view displays the request, response, and cache status for load domain name resolution.

Metric Description

You can view CoreDNS metrics in the following table.

Figure 10-50 CoreDNS metrics

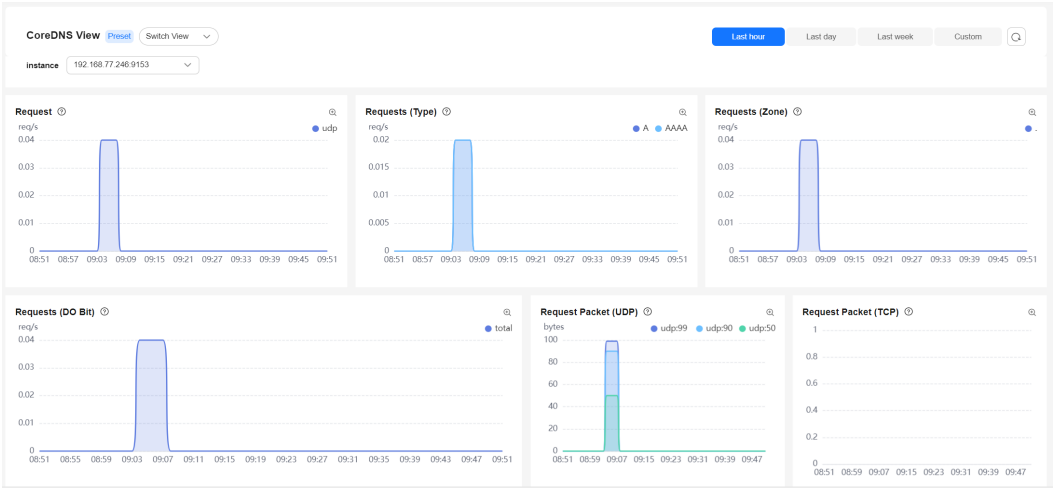


Table 10-30 CoreDNS metrics

Metric	Unit	Description
Requests	N/A	The number of requests CoreDNS receives per second.
Requests (Type)	N/A	The number of requests CoreDNS receives per second by each request type.
Requests (Zone)	N/A	The number of requests CoreDNS receives per second by each zone.
Requests (DO Bit)	N/A	The number of requests with the DO bit CoreDNS receives per second.
Request Packet (UDP)	byte	P99, P90, and P50 latencies of UDP packets CoreDNS receives.
Request Packet (TCP)	byte	P99, P90, and P50 latencies of TCP packets CoreDNS receives.
Responses (Status Code)	N/A	The number of responses CoreDNS sends per second by status code.
Response Latency	ms	P99, P90, and P50 latencies of CoreDNS.
Response Packet (UDP)	byte	P99, P90, and P50 latencies of UDP packets CoreDNS sends.

Metric	Unit	Description
Response Packet (TCP)	byte	P99, P90, and P50 latencies of TCP packets CoreDNS sends.
Cache Records	N/A	The number of DNS records cached by CoreDNS.
Cache Hit Ratio	N/A	The number of CoreDNS cache hits per second.

Metric List

The following is the metric list of the CoreDNS view.

Table 10-31 Metric description

Metric	Type	Description
coredns_dns_request_count_total	Counter	Cumulative count of DNS requests made per zone, protocol, and family.
coredns_dns_requests_total	Counter	Number of DNS requests made per zone, protocol, and family.
coredns_dns_request_type_count_total	Counter	Cumulative count of DNS requests per type.
coredns_dns_request_do_count_total	Counter	Cumulative count of requests with the DO bit set.
coredns_dns_do_requests_total	Counter	The number of requests with the DO bit set.
coredns_dns_request_size_bytes_bucket	Histogram	The number of CoreDNS request bytes.
coredns_dns_response_rcode_count_total	Counter	Cumulative count of response status codes.
coredns_dns_responses_total	Counter	The number of response status codes.
coredns_dns_request_duration_seconds_bucket	Histogram	CoreDNS request latency.
coredns_dns_response_size_bytes_bucket	Histogram	Size of the returned response in bytes.
coredns_cache_size	Gauge	Cache size.
coredns_cache_hits_total	Counter	The number of cache hits.

10.4.9.11 PVC View

The PVC view displays the PV/PVC status and usage.

NOTE

The following PVC types support usage monitoring:

- EVS PVCs (The value of **volumeMode** must be **Filesystem**.)
- Local PVCs (The Everest version installed in the cluster must be 2.4.41 or later.)
- SFS Turbo PVCs (Subdirectories generated dynamically by SFS Turbo also support usage monitoring. However, the PVC usage and capacity collected from the subdirectories are the same as those of the SFS Turbo instance).
- PVCs mounted to common containers (The usages and inodes usages can be collected for such PVCs, but not for the PVCs mounted to secure containers.)

Metric Description

You can view PVC metrics in the following table.

Table 10-32 PVC metrics

Metric	Unit	Description
PV Status	N/A	PV status, which can be **Available**, **Bound**, **Failed**, **Pending**, or **Released**.
PVC Status	N/A	PVC status, which can be **Bound**, **Lost**, or **Pending**.
Used PVC	GiB	Used and available PVC space.
PVC Usage	%	Percentage of a PVC that is used.
Used Inodes in PVC	N/A	Used and available space of inodes in a PVC.
Inodes Usage in PVC	%	Percentage of inodes that are used in a PVC.

Metric List

The following is the metric list of the PVC view.

Table 10-33 Metric description

Metric	Type	Description
kubelet_volume_stats_inodes_used	Gauge	The number of used inodes in a volume.
kubelet_volume_stats_inodes	Gauge	Maximum number of inodes in a volume.

Metric	Type	Description
kubelet_volume_stats_capacity_bytes	Gauge	Capacity in bytes of a volume.
kubelet_volume_stats_available_bytes	Gauge	The number of available bytes in a volume.
kubelet_volume_stats_used_bytes	Gauge	The number of used bytes in a volume.
kube_persistentvolume_status_phase	Gauge	PV status.
kube_persistentvolume-claim_status_phase	Gauge	PVC status.

10.4.9.12 Kubelet View

The kubelet is the agent that runs on each node in a cluster. The kubelet view allows you to monitor a cluster.

Metric Description

You can view kubelet metrics in the following table.

Table 10-34 Kubelet metrics

Metric	Unit	Description
Running Kubelets	N/A	The number of running kubelets in a cluster.
Running Pods	N/A	The number of running pods on the node where the kubelet resides.
Running Containers	N/A	The number of running containers on the node where the kubelet resides.
Actual Volumes	N/A	The actual number of volumes on the node where the kubelet resides.
Expected Volumes	N/A	The desired number of volumes on the node where the kubelet resides.
Configuration Errors	N/A	The number of incorrect Kubelet configurations on the node where the kubelet resides.
Operation Rate	N/A	The number of operations performed by the kubelet per second.
Operation Error Rate	N/A	The number of failed operations performed by the kubelet per second.

Metric	Unit	Description
Operation Latency	second	The latencies of operations performed by the kubelet.
Pod Startup Rate	N/A	The number of pod startups performed by the kubelet per second.
Pod Startup Latency (P99)	second	The P99 latencies of pod startups performed by the kubelet.
Storage Operation Rate	N/A	The number of storage-related operations performed by the kubelet per second.
Storage Operation Error Rate	N/A	The number of failed storage-related operations performed by the kubelet per second.
Storage Operation Latency (P99)	second	The P99 latencies of storage-related operations performed by the kubelet.
Cgroup Manager Operation Rate	N/A	The number of destroy or update operations performed by the kubelet per second.
Cgroup Manager Operation Latency (P99)	second	The P99 latencies of destroy or update operations performed by the kubelet.
PLEG Relist Rate	N/A	The number of relisting operations in PLEG per second.
PLEG Relist Interval (P99)	second	The intervals between 99% of relisting operations in PLEG.
PLEG Relist Latency (P99)	second	The P99 latencies of relisting operations in PLEG.
RPC Rate	N/A	The number of RPC requests with each status code.
Request Latency (P99)	second	The P99 latencies of requests with each method.
Used Memory	byte	The memory used by the kubelet.
Used CPU	byte	The CPU used by the kubelet.
Goroutines	N/A	The number of goroutines.

Metric List

The following is the metric list of the kubelet view.

Table 10-35 Metric description

Metric	Type	Description
storage_operation_errors_total	Counter	The number of errors in storage operations.
storage_operation_duration_seconds_count	Counter	The number of storage operations.
storage_operation_duration_seconds_bucket	Histogram	Duration for each storage operation.
kubelet_pod_start_duration_seconds_count	Counter	The number of pods that have been started.
kubelet_pod_start_duration_seconds_bucket	Histogram	Duration from the kubelet seeing a pod for the first time to the pod starting to run.
kubelet_runtime_operations_duration_seconds_bucket	Histogram	The time of every operation.
kubelet_runtime_operations_errors_total	Counter	The number of errors in operations at runtime level.
kubelet_node_config_error	Gauge	If a configuration-related error occurs on a node, the value of this metric is **true (1)**. If there is no configuration-related error, the value is **false (0)**.
volume_manager_total_volumes	Gauge	The number of volumes in Volume Manager.
kubelet_running_containers	Gauge	The number of running containers.
kubelet_running_pods	Gauge	The number of running pods.
kubelet_node_name	Gauge	Node name. The value is always **1**.
kubelet_runtime_operations_total	Counter	The number of total runtime operations of each type.
kubelet_cgroup_manager_duration_seconds_count	Counter	The number of destruction and update operations.
kubelet_cgroup_manager_duration_seconds_bucket	Histogram	Duration for destruction and update operations.
kubelet_pleg_relist_duration_seconds_count	Counter	The number of relisting operations in PLEG.
kubelet_pleg_relist_interval_seconds_bucket	Histogram	The intervals between relisting operations in PLEG.

Metric	Type	Description
kubelet_pleg_relist_duration_seconds_bucket	Histogram	Duration for relisting pods in PLEG.
rest_client_requests_total	Counter	The total number of HTTP requests, partitioned by status code and method.
rest_client_request_duration_seconds_bucket	Histogram	The number of HTTP requests, partitioned by status code and method.
process_resident_memory_bytes	Gauge	Resident memory size in bytes.
process_cpu_seconds_total	Counter	Total user and system CPU time spent in seconds.
go_goroutines	Gauge	The number of goroutines.

10.4.9.13 Prometheus Server View

Prometheus with local data storage enabled scrapes the metrics of all hosts and components. The metrics are then reported to and stored on AOM or a third-party monitoring platform. The Prometheus Server view displays some built-in metrics provided by Prometheus, which can be used to monitor and measure system performance and status.

Metric Description

You can view Prometheus Server metrics in the following table.

Figure 10-51 Prometheus Server metrics

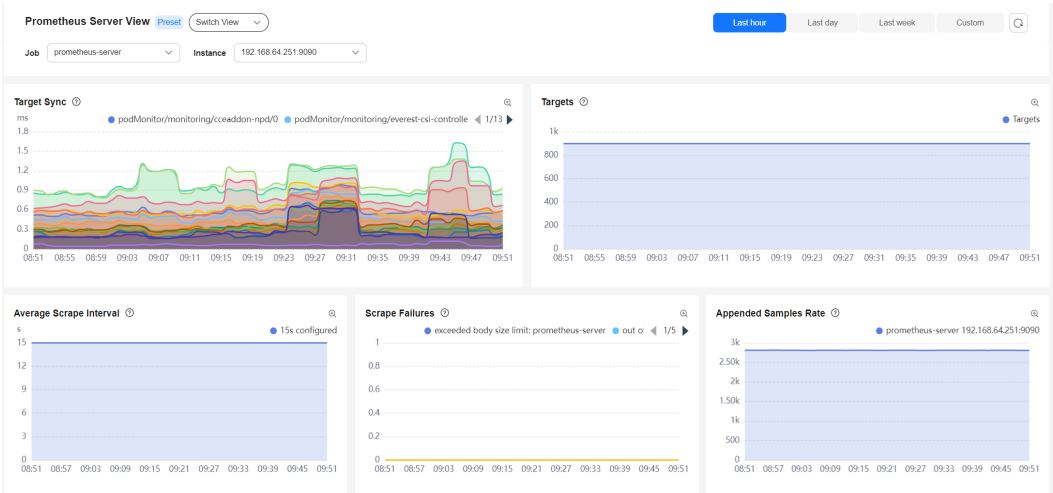


Table 10-36 Prometheus Server metrics

Metric	Unit	Description
Target Sync	second	Collection latency of the target.
Targets	N/A	Total number of metrics collected by the target.
Average Scrape Interval	second	Interval for collecting metrics.
Scrape Failures	N/A	The number of collection failures.
Appended Samples Rate	N/A	The rate at which samples are added to the head.
Head Series	N/A	The number of series in the head.
Head Chunks	N/A	The number of head blocks.
Query Rate	N/A	The number of Prometheus queries per second.
P90 Query Duration	second	Duration for querying 90% of operations on different shards.
Remote Sample Lag Ratio	second	Percentage of the highest timestamp of samples stored in WAL.
Remote Write Traffic	byte/s	Remote write rate.
Current Shards	N/A	The number of shards used for parallel sending to the remote storage.
Max Shards	N/A	Maximum number of shards that the queue is allowed to run.
Min Shards	N/A	Minimum number of shards that the queue is allowed to run.
Desired Shards	N/A	The number of shards that the queue wants to run based on the percentage of input samples to output samples.
Shard Capacity	N/A	Capacity of each shard of the queue used for parallel sending to the remote storage.
Pending Samples	N/A	Capacity of each shard of the queue used for parallel sending to the remote storage.
Current TSDB Segment	N/A	WAL segment index that TSDB is currently writing to.

Metric	Unit	Description
Current Segment of Remote Write	N/A	Current segment from which the WAL watcher is reading records.
Sample Discard Rate	N/A	The rate at which samples are discarded after being read from WAL before being sent via remote write.
Sample Failure Rate	N/A	Rate of samples that failed to be sent to remote storage due to unrecoverable errors.
Sample Retry Rate	N/A	Rate of samples that failed to be sent to remote storage due to recoverable errors and were resent.
Retry Rate of Enqueuing	N/A	Retry rate of enqueuing failed due to full shard queue.

Metric List

The following is the metric list of the Prometheus Server view.

Table 10-37 Metric description

Metric	Type	Description
prometheus_target_sync_length_seconds_sum	Summary	Collection latencies of different targets.
prometheus_sd_discovered_targets	Gauge	The number of metrics collected by different targets.
prometheus_target_interval_length_seconds_sum	Summary	Collection interval.
prometheus_target_scrapes_exceeded_body_size_limit_total	Counter	The number of collection failures.
prometheus_tsdb_head_samples_appended_total	Counter	Total number of samples added to the head.
prometheus_tsdb_head_series	Gauge	The number of series in the head block.
prometheus_tsdb_head_chunks	Gauge	The number of chunks in the head block.
prometheus_engine_query_duration_seconds_count	Counter	The number of queries.

Metric	Type	Description
prometheus_engine_query_duration_seconds	Counter	Time taken by queries to complete, in seconds.
prometheus_remote_storage_highest_timestamp_in_seconds	Gauge	Latest timestamp in the remote storage.
prometheus_remote_storage_queue_highest_sent_timestamp_seconds	Gauge	Latest timestamp in the Prometheus shard.
prometheus_remote_storage_bytes_total	Counter	Total number of bytes of data (non-metadata) sent by the queue after compression.
prometheus_remote_storage_shards	Gauge	The number of shards used for parallel sending to the remote storage.
prometheus_remote_storage_shards_max	Gauge	Maximum number of shards that the queue is allowed to run.
prometheus_remote_storage_shards_min	Gauge	Minimum number of shards that the queue is allowed to run.
prometheus_remote_storage_shards_desired	Gauge	The number of shards that the queue wants to run based on the percentage of input samples to output samples.
prometheus_remote_storage_shard_capacity	Gauge	Capacity of each shard of the queue used for parallel sending to the remote storage.
prometheus_remote_storage_pending_samples	Gauge	The number of pending samples in queue shards to be sent to remote storage.
prometheus_tsdb_wal_segment_current	Gauge	WAL segment index that TSDB is currently writing to.
prometheus_wal_watcher_current_segment	Gauge	Current segment from which the WAL is reading records.
prometheus_remote_storage_dropped_samples_total	Gauge	The rate at which samples are discarded after being read from WAL before being sent via remote write.

Metric	Type	Description
prometheus_remote_storage_failed_samples_total	Gauge	Rate of samples that failed to be sent to remote storage due to unrecoverable errors.
prometheus_remote_storage_retried_samples_total	Gauge	The number of samples that failed to be sent to remote storage due to recoverable errors and were resent.
prometheus_remote_storage_enqueue_retries_total	Gauge	The number of retries upon enqueuing failed due to full shard queue.

10.4.9.14 Prometheus Agent View

Prometheus Agent is a lightweight version of Prometheus Server. It scrapes the metrics for all hosts and components. The metrics data is then reported to and stored on AOM or a third-party monitoring platform. The Prometheus Agent view displays some built-in metrics provided by Prometheus, which can be used to monitor and measure system performance and status.

Metric Description

You can view Prometheus Agent metrics in the following table.

Figure 10-52 Prometheus Agent metrics

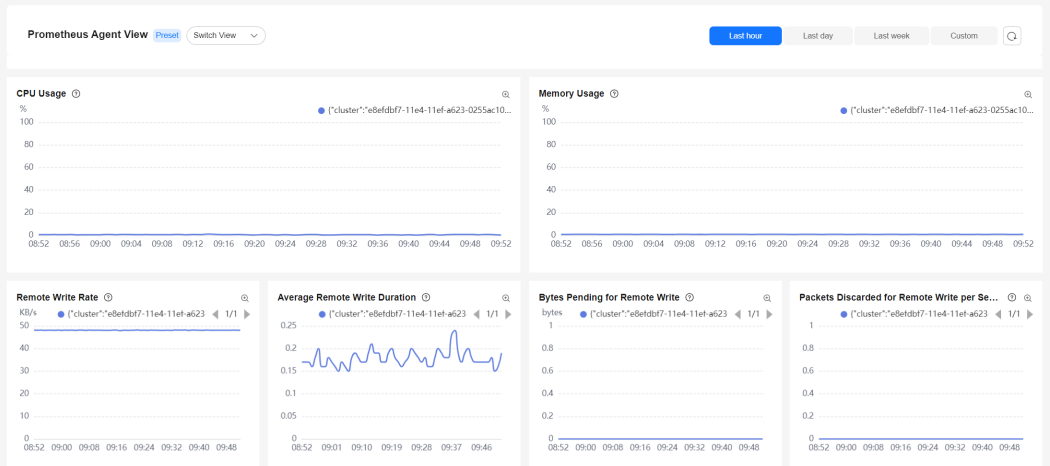


Table 10-38 Prometheus Agent metrics

Metric	Unit	Description
CPU Usage	%	Average CPU usage of pods.
Memory Usage	%	Average memory usage of pods.

Metric	Unit	Description
Remote Write Rate	byte/s	The number of bytes remotely written per second.
Average Remote Write Duration	second	Average time consumed by remote writes.
Bytes Pending for Remote Write	byte	The number of pending bytes during a remote write.
Packets Discarded for Remote Write per Second	N/A	The number of packets discarded per second during a remote write.
Failed Remote Write Requests per Second	N/A	The number of failed remote write requests per second.
Percentage of Failed Remote Write Requests	%	Percentage of failed remote write requests.
Remote Write Retries per Second	N/A	The number of retries of the remote write per second.
Scrapers	N/A	The number of scrapers.
Collections per Second	N/A	Collections per second.
Average Collection Duration	second	Average collection duration.
Failed Reads During Collection per Second	N/A	The number of read errors during scrapes per second.
Failed Writes During Collection per Second	N/A	The number of write errors during scrapes per second.
Collections with Size Exceeding the Threshold per Second	N/A	Collections with size exceeding the threshold per second.
Read Rate of Sending Queue	byte/s	The number of bytes read by a sending queue per second.
Write Rate of Sending Queue	byte/s	The number of bytes written to a sending queue per second.
Pending Size of Sending Queue	byte	The number of bytes suspended in a sending queue.
Blocks Read per Second	N/A	The number of blocks read by a sending queue per second.
Blocks Written per Second	N/A	The number of blocks written to a sending queue per second.

Metric	Unit	Description
Discarded Blocks per Second	N/A	The number of dropped blocks in a sending queue per second.

Metric List

The following is the metric list of the Prometheus Agent view.

Table 10-39 Metric description

Metric	Type	Description
container_cpu_usage_seconds_total	Gauge	Cumulative CPU time consumed in seconds.
container_memory_working_set_bytes	Gauge	Current working set in bytes.
vmagent_remotewrite_bytes_sent_total	Counter	Total number of bytes remotely written by Prometheus Agent.
vmagent_remotewrite_duration_seconds_sum	Summary	Time consumed by a Prometheus Agent remote write.
vmagent_remotewrite_pending_data_bytes	Gauge	The number of pending bytes during a Prometheus Agent remote write.
vmagent_remotewrite_packets_dropped_total	Counter	Total number of dropped packets during a Prometheus Agent remote write.
vmagent_remotewrite_requests_total	Counter	Total number of Prometheus Agent remote write requests.
vmagent_remotewrite_retries_count_total	Counter	Total number of remote write retries of Prometheus Agent.
vm_promscrape_active_scrapers	Gauge	The number of collected shards.
vm_promscrape_scrapes_total	Counter	The number of scrapes.
vm_promscrape_scrape_duration_seconds_sum	Summary	Time required for the vmagent to collect metrics.
vm_promscrape_conn_read_errors_total	Counter	The number of read errors during scrapes.
vm_promscrape_conn_write_errors_total	Counter	The number of write errors during scrapes.

Metric	Type	Description
vm_promscrape_max_scrape_size_exceeded_errors_total	Counter	The number of failed scrapes due to the exceeded response size.
vm_persistentqueue_bytes_read_total	Counter	The number of bytes read by a send queue.
vm_persistentqueue_bytes_written_total	Counter	The number of bytes written to a send queue.
vm_persistentqueue_bytes_pending	Gauge	The number of pending bytes in a send queue.
vm_persistentqueue_blocks_read_total	Counter	The number of blocks read by a send queue.
vm_persistentqueue_blocks_written_total	Counter	The number of blocks written to a send queue.
vm_persistentqueue_blocks_dropped_total	Counter	The number of dropped blocks in a send queue.

10.5 Logging

10.5.1 Overview

Kubernetes logs allow you to locate and rectify faults. This section describes how you can manage Kubernetes logs generated for CCE in multiple ways.

You can manage Kubernetes logs generated for CCE in the following ways:

- Use the Cloud Native Log Collection add-on to collect application logs and report them to LTS, which provides log statistics and analysis. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.
- (Not recommended) Connect CCE to AOM. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.
- Collect control plane component logs and Kubernetes audit logs from the CCE control plane and add them to the LTS log streams in your account. For details, see **Collecting Control Plane Component Logs** and **Collecting Kubernetes Audit Logs**.
- Collect Kubernetes events and add them to the LTS log stream in your account for persistent storage and statistical analysis. For details, see **Collecting Kubernetes Events**.
- Collects the logs of the Nginx Ingress Controller add-on to analyze traffic changes and obtain service traffic characteristics, providing data support for service decision making. For details, see **Collecting NGINX Ingress Controller Logs**.

Comparison Between ICAgent and Cloud Native Log Collection

Table 10-40 Comparison between ICAgent and Cloud Native Log Collection

Collection Tool	ICAgent		Cloud Native Log Collection	
Log Storage Location	LTS	AOM 1.0	LTS	AOM 2.0
Content to Be Collected	- Container stdout logs - Container file logs - Node logs - Kubernetes events	- Container stdout logs - Container file logs	- Container stdout logs - Container file logs - Node logs - Kubernetes events	Kubernetes events
Advantages and Disadvantages	- Log collection policies and workloads are configured separately. Modifying policies does not affect pod running. - You can specify a container whose logs are to be collected. - Docker and containerd are supported. If a node uses containerd, the ICAgent version must be 5.12.130 or later. - Container file log collection supports overlay2, not Device Mapper.	- Each workload needs to be configured separately. - Log collection policies are coupled with pods. Modifying policies will restart the pod.	- Log collection policies and workloads are configured separately. Modifying policies does not affect pod running. - You can specify a container whose logs are to be collected. - If the node storage driver is Device Mapper, container file logs must be collected from the path where the data disk is attached to the node.	All warning events and some normal events are reported by default. The reported events can be used to configure alarms.

Configuration Method	Create a collection policy on LTS. For details, see **Collecting Logs from CCE**.	Create a collection policy in the workload. For details, see **Collecting Container Logs Using ICAgent (Not Recommended)**.	Create a policy on the **Logging** page. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.	For details, see **Reporting Kubernetes Events to AOM**.
Monitored Directories	Up to five levels of directories, with up to 1,000 files		Up to three levels of directories by fuzzy match	-
Monitored Files	• Up to 20 logs from a volume mounting directory • Up to 1,000 stdout logs in JSON format		Up to 4,096 logs collected based on log policies on each node	-

10.5.2 Collecting Container Logs

10.5.2.1 Collecting Container Logs Using Cloud Native Log Collection

The Cloud Native Log Collection add-on (**Cloud Native Log Collection**) is developed based on Fluent Bit and OpenTelemetry for collecting logs and Kubernetes events. This add-on supports CRD-based log collection policies. It collects and forwards stdout logs, container file logs, node logs, and Kubernetes events in a cluster based on configured policies.

Constraints

- Up to 50 log rules can be created for each cluster.
- The Cloud Native Log Collection add-on cannot collect .gz, .tar, and .zip logs and cannot access symbolic links of logs.
- If the node **storage driver** is Device Mapper, container file logs must be collected from the path where the data disk is attached to the node.
- If the container runtime is containerd, each stdout log cannot be in multiple lines. (This does not apply to the Cloud Native Log Collection add-on of version 1.3.0 or later.)
- If a volume is attached to the data directory of a service container, this add-on cannot collect data from the parent directory. In this case, you need to configure a complete data directory.
- If the lifetime of a container is less than 1 minute, logs cannot be collected in a timely manner. As a result, logs may be lost.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota. For details, see **Price Calculator**.

Configuring Log Collection on the Console

Step 1 Enable log collection.

Enabling log collection during cluster creation

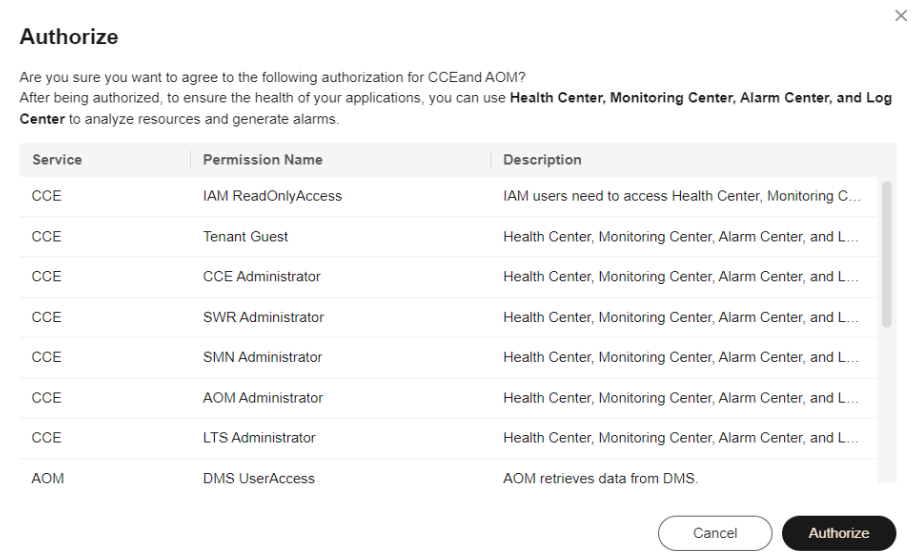
1. Log in to the CCE console.
2. Click **Buy Cluster** from the top menu.
3. On the **Select Add-on** page, select **Cloud Native Log Collection**.
4. Click **Next: Add-on Configuration** in the lower right corner and select the required logs.
   - Container logs: A log collection policy named **default-stdout** will be created, which will report stdout logs from all namespaces to LTS.
   - Kubernetes events: A log collection policy named **default-event** will be created, which will report Kubernetes events from all namespaces to LTS.
5. Click **Next: Confirm Configuration** in the lower right corner. On the displayed page, click **Submit**.

Enabling log collection for an existing cluster

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. (Optional) If you are not authorized, obtain required permissions first.

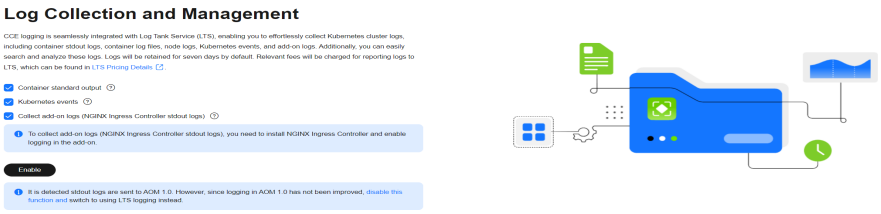
   In the displayed dialog box, click **Authorize**.

Figure 10-53 Authorize



3. Click **Enable** and wait for about 30 seconds until the log page is automatically displayed.

Figure 10-54 Enable



- Stdout logs: A log collection policy named **default-stdout** will be created, which will report stdout logs from all namespaces to LTS.
- Kubernetes events: A log collection policy named **default-event** will be created, which will report Kubernetes events from all namespaces to LTS.
- To collect add-on logs (stdout logs of NGINX Ingress Controller), you need to install NGINX Ingress Controller and enable logging for this add-on.

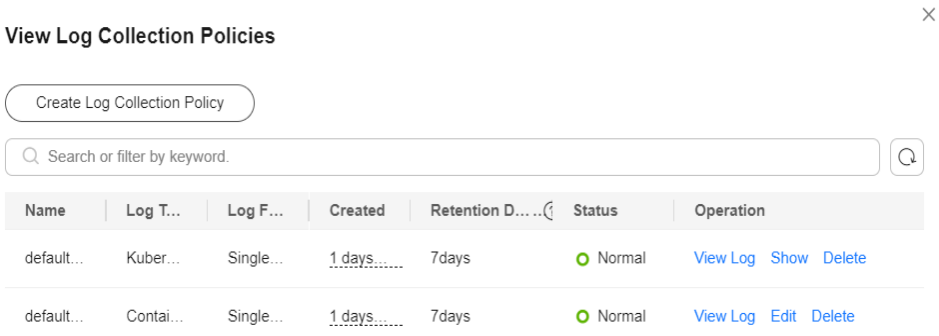
  After logging is enabled, a log policy named **default-nginx-ingress** will be created to collect the stdout logs of NGINX Ingress Controller and report the logs to LTS.

Step 2 View and configure log collection policies.

1. On the CCE console, click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.

Figure 10-55 Viewing log collection policies

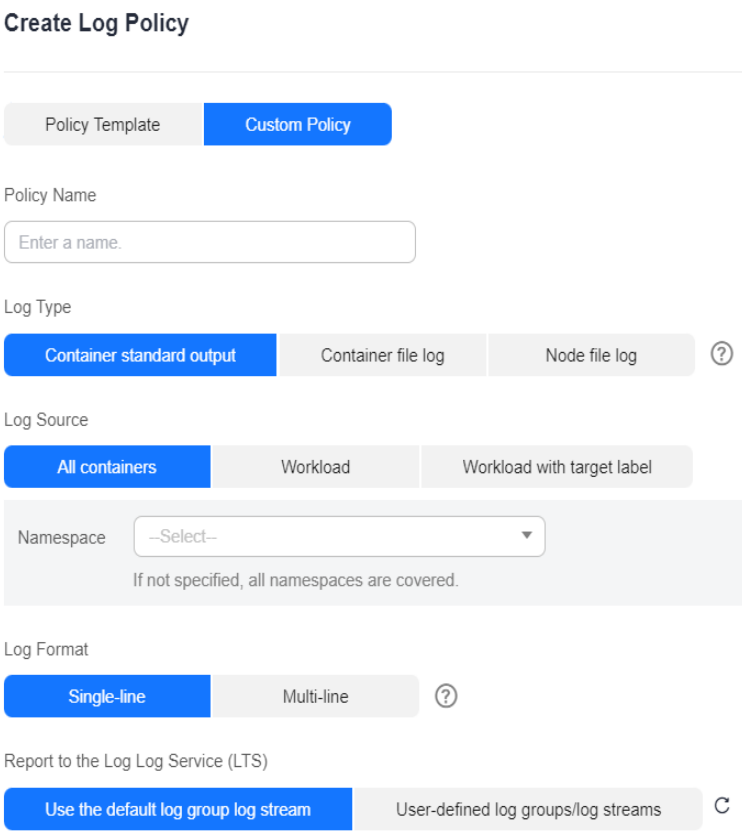


3. Click **Create Log Collection Policy**.

   Policy Template: If the default log collection policy provided by CCE is not enabled when log collection is enabled or the default log collection policy is deleted, you can use this option to create a default log collection policy.

   Custom Policy: You can use this option to create a custom log collection policy.

Figure 10-56 Custom policy



NOTE

- To avoid log disorder, you are advised to select different log streams for reporting logs in the log collection policies of various log types.
- The following are requirements for configuring the container and node file log paths:

  - Log directory: Enter an absolute path, for example, **/log**. The path must start with a slash (/) and contain a maximum of 512 characters. Only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), slashes (/), asterisks (*), and question marks (?) are allowed.
  - Log file name: It can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), asterisks (*), question marks (?), and periods (.). Logs in the format of .gz, .tar, and .zip are not supported.

  The directory and file names must be complete and support asterisks (*) and question marks (?) as wildcards. A maximum of three levels of directories can be matched using wildcards. The level-1 directory does not support wildcards. An asterisk (*) can match multiple characters. A question mark (?) can match only one character. For example:

  - If the directory is **/var/logs/*** and the file name is ***.log**, the match expression is **/var/logs/*/*.log**, indicating that any files with the extension .log in all level-1 directories in the **/var/logs** directory are matched. Note that this expression cannot match any files with the extension .log in the **/var/logs** directory and multi-level directories in the **/var/logs** directory.
  - If the directory is **/var/logs/app_*** and the file name is ***.log**, any log files with the extension .log in all directories that match app_* in the **/var/logs** directory will be reported.

Table 10-41 Custom policy parameters

Parameter	Description		
Log Type	**Container standard output**: used to collect container stdout logs. You can create a log collection policy by namespace, workload name, or instance label.	**Container file log**: used to collect text logs. You can specify a workload or instance label to create a log collection policy.	**Node file log**: used to collect logs from a node. Only one file path can be configured for a log collection policy.

Parameter	Description		
Log Source	– **All containers**: You can specify all containers in a namespace. If this parameter is not specified, logs of containers in all namespaces will be collected. – **Workload**: You can specify a workload and its containers. If this parameter is not specified, logs of all containers running the workload will be collected. – **Workload with target label**: You can specify a workload by label and its containers. If this parameter is not specified, logs of all containers running the workload will be collected.	– **Workload**: You can specify a workload and its containers. If this parameter is not specified, logs of all containers running the workload will be collected. – **Workload with target label**: You can specify a workload by label and its containers. If this parameter is not specified, logs of all containers running the workload will be collected. You also need to specify the log collection path. For details, see the **log path configuration requirements**.	**Collection Path**: used to configure the log collection path. For details, see the **log path configuration requirements**.

Parameter	Description

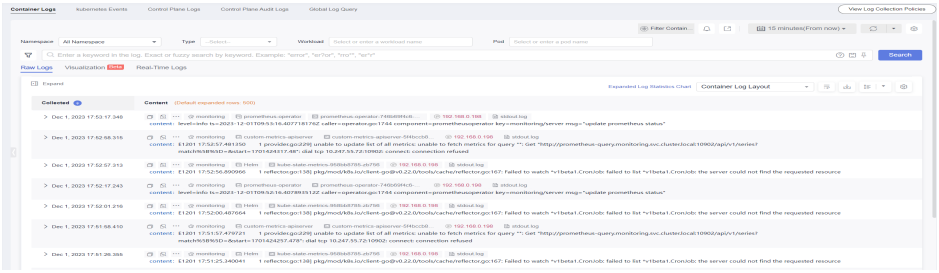
Log Format	– **Single-line**
	Each log contains only one line of text. The newline character \n denotes the start of a new log.
	– **Multi-line**
	Some programs (for example, Java program) print a log that occupies multiple lines. By default, logs are collected by line. If you want to display logs as a single message, you can enable multi-line logging and use the regular pattern. When you select **Multi-line**, configure **Log Matching Format**.
	For example, if logs need to be collected by line and each log starts with a date and occupies three lines, you can set **Log Matching Format** to the regular expression of the date, for example, **\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.***.
	The three lines starting with the date are regarded as a log.
	`2022-01-01 00:00:00 Exception in thread "main" java.lang.RuntimeException: Something has gone wrong, aborting!`
	`at com.myproject.module.MyProject.badMethod(MyProject.java:22)`
	`at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)`

Report to LTS	This parameter is used to configure the log group and log stream for log reporting. – **Default log groups/log streams**: The default log group (**k8s-log-*{Cluster ID}***) and default log stream (**stdout-*{Cluster ID}***) are automatically selected. – **Custom log groups/log streams**: You can select any log group and log stream. ▪ **Log Group**: A log group is the basic unit for LTS to manage logs. If you do not have a log group, CCE prompts you to create one. The default name is **k8s-log-*{Cluster ID}***, for example, **k8s-log-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**. ▪ **Log Stream**: A log stream is the basic unit for reading and writing logs. You can put different types of logs into different streams to ease management. When you install the add-on or create a log collection policy based on the policy template, the following log streams are automatically created: - **stdout-*{Cluster ID}*** for container logs, for example, **stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3** - **event-*{Cluster ID}*** for Kubernetes events, for example, **event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3** - **cceaddon-nginx-ingress-*{Cluster ID}*** for NGINX Ingress Controller logs, for example, **cceaddon-nginx-ingress-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.

Step 3 View the logs.

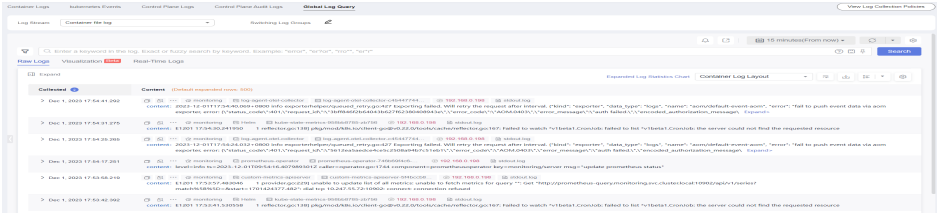
1. On the CCE console, click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. View different types of logs:
   - **Container Logs**: displays all logs in the default log stream **stdout-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*. You can search for logs by workload.

     Figure 10-57 Querying container logs

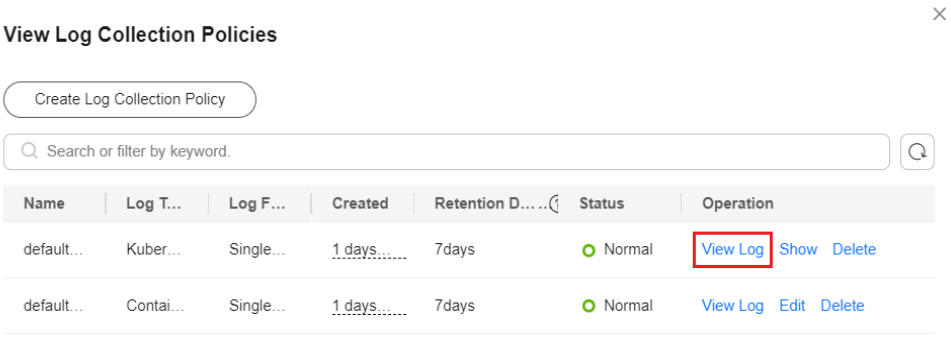
   - **Kubernetes Events**: displays all Kubernetes events in the default log stream **event-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   - **Control Plane Logs**: displays all logs of components on the control plane in the default log stream *{Component name}*-*{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   - **Control Plane Audit Logs**: displays all control plane audit logs in the default log stream **audit-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   - **Global Log Query**: You can view logs in the log streams of all log groups. You can specify a log stream to view the logs. By default, the default log group **k8s-log-***{Cluster ID}* is selected. You can click the edit icon on the right of **Switching Log Groups** to switch to another log group.

     Figure 10-58 Global log query

   - **Add-on Logs**: displays the add-on logs in the default log group **k8s-log-***{Cluster ID}*. You can view the cluster add-on logs.
3. Click **View Log Collection Policies** in the upper right corner. Locate the log collection policy and click **View Log** to go to the log list.

Figure 10-59 Viewing logs



----End

Configuring Log Collection Using YAML

NOTICE

The Cloud Native Log Collection add-on must be 1.6.1 or later.

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **log-config.yaml**. You can create a custom file name.

```
vi log-config.yaml
```

The following examples are for your reference. For details about the parameters, see **Table 10-42**.

- **Scenario 1: Collecting stdout logs of all workloads**

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  name: test-log-01  # Change the rule name as needed.
  namespace: kube-system  #Namespace of the collection rule. The value is fixed at kube-system.
spec:
  inputDetail  : # Input configuration
    type: container_stdout  # Input type. container_stdout indicates stdout logs.
    containerStdout:  # Stdout log configuration. This parameter is valid only when type is set to container_stdout.
      allContainers: true  # Whether to collect the logs of all containers.
      namespaces: []  # Namespace list, which is of array type. This parameter is valid only when allContainers is set to true. The stdout logs of containers in specified namespaces will be collected. An empty array indicates that the stdout logs of containers in all namespaces will be collected.
  outputDetail:  # Output configuration
    type: LTS  # Output type. The value is fixed at LTS.
    LTS:
      ltsGroupID: abf5f0ad-627e-41cc-8d3f-61c9e1f57f5a  # LTS log group ID. The specified ID must exist.
      ltsStreamID: f7ed71e9-6b9d-4ba3-86e4-b1b9d22ef4fb  # LTS log stream ID. The specified ID must exist.
```

- **Scenario 2: Collecting container file logs of a specified workload**

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  name: test-log-02  # Change the rule name as needed.
  namespace: kube-system  #Namespace of the collection rule. The value is fixed at kube-system.
spec:
```



```
  inputDetail  : # Input configuration
    type: container_file   # Input type. container_file indicates container file logs.
    containerFile:    # Container file log configuration. This parameter is valid only when type is set to container_file.
      workloads:        # Modify the workload information as needed.
      - namespace: monitoring  # Namespace that the workload belongs to
        kind: Deployment  # Workload type. The value can be Deployment, DaemonSet, StatefulSet, Job, or CronJob.
        name: prometheus-lightweight  # Workload name
        container: prometheus  # Container name
        files:
        - logPath: "/var/log"  # Log directory, which is an absolute path.
          filePattern: "*.log"  # Log file name, which supports wildcard characters.
    processors:    # Multi-line log definition. If multiple lines are not required, delete processors.
      type: multiline # Log type, which is optional. The value can be singleline or multiline. The default value is singleline.
      multilineRegulation: '\d+:\d+:\d+.*?'  # Multi-line regular expression, which is optional. This field is valid only when type is set to multiline.
  outputDetail:  # Output configuration
    type: LTS    # Output type. The value is fixed at LTS.
    LTS:
      ltsGroupID: abf5f0ad-627e-41cc-8d3f-61c9e1f57f5a      # LTS log group ID. The specified ID must exist.
      ltsStreamID: f7ed71e9-6b9d-4ba3-86e4-b1b9d22ef4fb      # LTS log stream ID. The specified ID must exist.
```

- **Scenario 3: Collecting container file logs of pods with specified labels**

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  name: test-log-03  # Change the rule name as needed.
  namespace: kube-system  #Namespace of the collection rule. The value is fixed at kube-system.
spec:
  inputDetail  : # Input configuration
    type: container_file   # Input type. container_file indicates container file logs.
    containerFile:    # Container file log configuration. This parameter is valid only when type is set to container_file.
      podLabels:      # Modify the value based on the CRD description.
      - includeLabels: # Label set. The logs of pods with the following labels will be collected. At least one label must be specified. Note that the pod label is not the label of the workload.
          foo: bar
        namespaces:  # Namespace list, which is of array type. An empty array indicates all namespaces.
        - monitoring
        - kube-system
        containers: []  # Container name list, which is of array type. An empty array indicates all containers.
        files:
        - logPath: "/var/log"  # Log directory, which is an absolute path.
          filePattern: "*.log"  # Log file name, which supports wildcard characters.
  outputDetail:  # Output configuration
    type: LTS    # Output type. The value is fixed at LTS.
    LTS:
      ltsGroupID: abf5f0ad-627e-41cc-8d3f-61c9e1f57f5a      # LTS log group ID. The specified ID must exist.
      ltsStreamID: f7ed71e9-6b9d-4ba3-86e4-b1b9d22ef4fb      # LTS log stream ID. The specified ID must exist.
```

- **Scenario 4: Collecting node logs**

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  name: test-log-04  # Change the rule name as needed.
  namespace: kube-system  #Namespace of the collection rule. The value is fixed at kube-system.
spec:
  inputDetail  : # Input configuration
    type: host_file   # Input type. host_file indicates node logs.
    hostFile:    # Node log configuration. This parameter is valid only when type is set to host_file.
      file:
        logPath: "/var/log"    # Log directory, which is an absolute path. Change it as needed.
        filePattern: "messages" # Log file name, which supports wildcard characters and can be
```



```
modified as needed.
  outputDetail:  # Output configuration
    type: LTS    # Output type. The value is fixed at LTS.
    LTS:
      ltsGroupID: abf5f0ad-627e-41cc-8d3f-61c9e1f57f5a      # LTS log group ID. The specified ID must
exist.
      ltsStreamID: f7ed71e9-6b9d-4ba3-86e4-b1b9d22ef4fb      # LTS log stream ID. The specified ID
must exist.
```

Table 10-42 Parameters

Parameter	Type	Description	Example
spec.inputDetail.type	String	Input type. The options are as follows: • **container_stdout**: indicates stdout logs. This field must be used together with the **containerStdout** field. • **container_file**: indicates container file logs. This field must be used together with the **containerFile** field. • **host_file**: indicates node logs. This field must be used together with the **hostFile** field.	-

Parameter	Type	Description	Example
spec.inputDetail.containerStdout	Object	Stdout log configuration. This parameter is valid only when **type** is set to **container_stdout**. It contains the following fields: • **allContainers**: Whether to collect the logs of all containers. If the value is **true**, the logs of all containers will be collected. In this case, you need to specify the **namespaces** field. If the value is **false**, the logs of specified workloads will be collected. In this case, you need to specify the **workloads** field. • **namespaces**: namespace list, which is of array type and is valid only when **allContainers** is set to **true**. The stdout logs of containers in specified namespaces will be collected. An empty array indicates that the stdout logs of containers in all namespaces will be collected. • **workloads**: workload list, which is of array type and is valid only when **allContainers** is set to **false**. – **namespace**: namespace that a workload belongs to. – **kind**: workload type. The value can be **Deployment**, **DaemonSet**, **StatefulSet**, **Job**, or **CronJob**. – **name**: workload name. – **containers**: container name list, which is of array type. An empty array indicates all containers. • **podLabels**: pod labels, which is of array type and is valid only when **allContainers** is set to **false** and **workloads** is left empty.	Example 1: Collecting the stdout logs of all containers in a namespace ... spec: inputDetail: **type: container_stdout** containerStdout: **allContainers: true** namespaces: - monitoring ... Example 2: Collecting the stdout logs of a workload ... spec: inputDetail: **type: container_stdout** containerStdout: **allContainers: false** workloads: - namespaces: monitoring kind: Deployment name: prometheus-lightweight container: prometheus ... Example 3: Collecting the stdout logs of a pod ... spec: inputDetail: **type: container_stdout** containerStdout: **allContainers: false** **workloads: []** podLabels: - includeLabels: foo: bar namespaces: - monitoring containers: [] ...

Parameter	Type	Description	Example
		– **includeLabels**: label set. The logs of pods with the following labels will be collected. At least one label must be specified. Note that the pod label is not the label of the workload. – **namespaces**: namespace list, which is of array type. An empty array indicates all namespaces. – **containers**: container name list, which is of array type. An empty array indicates all containers.	

Parameter	Type	Description	Example
spec.inputDetail.containerFile	Object	Container file log configuration. This parameter is valid only when **type** is set to **container_file**. It contains the following fields: • workloads: workload list, which is of array type. – **namespace**: namespace that a workload belongs to. – **kind**: workload type. The value can be **Deployment**, **DaemonSet**, **StatefulSet**, **Job**, or **CronJob**. – **name**: workload name. – **container**: container name. – **files**: file list, which is of array type and contains the **logPath** and **filePattern** fields. - **logPath**: log directory, which is an absolute path, for example, **/var/log**. - **filePattern**: log file name, which supports wildcard characters, for example, ***.log**. • **podLabels**: pod labels, which is of array type and is valid only when **workloads** is left empty. – **includeLabels**: label set. The logs of pods with the following labels will be collected. At least one label must be specified. Note that the pod label is not the label of the workload. – **namespaces**: namespace list, which is of array type. An empty array indicates all namespaces. – **containers**: container name list, which is of array type. An empty array indicates all containers. – **files**: file list, which is of array type and contains the	Example 1: Collecting container file logs of a workload ... spec: inputDetail: **type: container_file** containerFile: workloads: - namespaces: monitoring kind: Deployment name: prometheus-lightweight container: prometheus files: - logPath: "/var/log" filePattern: "*.log" ... Example 2: Collecting container file logs of a pod ... spec: inputDetail: **type: container_file** containerFile: **workloads: []** podLabels: - includeLabels: foo: bar namespaces: - monitoring containers: [] files: - logPath: "/var/log" filePattern: "*.log" ...

Parameter	Type	Description	Example
		logPath and **filePattern** fields. - **logPath**: log directory, which is an absolute path, for example, **/var/log**. - **filePattern**: log file name, which supports wildcard characters, for example, ***.log**.	
spec.inputDetail.hostFile	Object	Node log configuration. This parameter is valid only when **type** is set to **host_file**. • file: – - **logPath**: log directory, which is an absolute path, for example, **/var/log**. – - **filePattern**: log file name, which supports wildcard characters, for example, ***.log**.	... spec: inputDetail: **type: host_file** hostFile: files: logPath: "/var/log" filePattern: "*.log" ...
spec.inputDetail.processors	Object	• **type**: log type, which is optional. The value can be **singleline** or **multiline**. The default value is **singleline**. • **multilineRegulation**: multi-line regular expression. This parameter is optional and is valid only when **type** is set to **multiline**.	... processors: type: multiline multilineRegulation: '\d+:\d+:\d+.*?' ...
spec.outputDetail.type	String	Output type. The value is fixed at **LTS**.	-

Parameter	Type	Description	Example
spec.outputDetail.LTS	Object	The following fields are supported: • **ltsGroupID**: LTS log group ID. The specified ID must exist. • **ltsStreamID**: LTS log stream ID. The specified ID must exist. Either **ltsStreamID** or **ltsStreamName** must be configured. • **ltsStreamName**: LTS log stream name. If the specified log stream name does not exist, it will be automatically created. Either **ltsStreamID** or **ltsStreamName** must be configured. • **ltsStreamCreateParam**: log stream creation parameter. This parameter is optional and is valid only when **ltsStreamName** is specified and an LTS log stream is automatically created. – **enterpriseProjectID**: Enterprise project ID of the LTS log group. This field is optional. If this field is not specified, the ID of the enterprise project that the cluster belongs to will be used.	Example 1: Specifying an existing log group ID and log stream ID ... LTS: ltsGroupID: ***** ltsStreamID: ***** Example 2: Specifying an existing log group ID and an existing log stream name ... LTS: ltsGroupID: ***** ltsStreamName: test-stream-name-1 Example 3: Specifying an existing log group ID and a new log stream name to automatically create a log stream ... LTS: ltsGroupID: ***** ltsStreamName: test-stream-name-2 ltsStreamCreateParam: enterpriseProjectID: ""

Step 3 Create a LogConfig.

```
kubectl create -f log-config.yaml
```

If information similar to the following is displayed, the LogConfig has been created:

```
logconfig.logging.openvessel.io/test-log-xx created
```

Step 4 Check the created LogConfig.

```
kubectl get LogConfig -n kube-system
```

If information similar to the following is displayed, the log collection policy has been created.

```
NAME            AGE
test-log-xx     30s
```

----End

10.5.2.2 Collecting Container Logs Using ICAgent (Not Recommended)

CCE works with AOM to collect workload logs. When a node is created, ICAgent (a DaemonSet named **icagent** in the **kube-system** namespace of a cluster) of AOM is installed by default. ICAgent collects workload logs and reports them to AOM. You can view workload logs on the CCE or AOM console.

Constraints

ICAgent only collects text logs in .log, .trace, and .out formats.

Billing

AOM offers a free log collection quota of 500 MB for each account every month. You pay only for log volume that exceeds the quota. For details, see **Billing**. You can click **here** to view logs on the AOM console.

Using ICAgent to Collect Logs

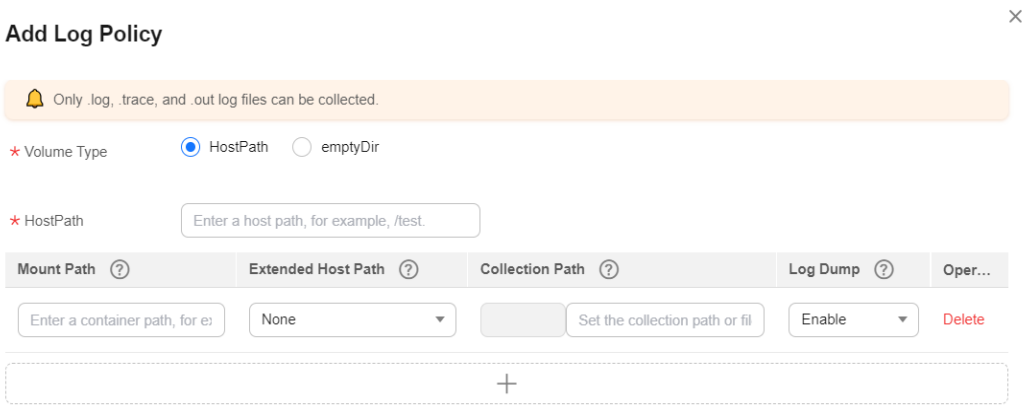
You can add a policy to collect logs using ICAgent for a workload.

Step 1 When **creating a workload**, set logging for the container.

Step 2 Click + to add a log policy.

The following uses Nginx as an example. Log policies vary depending on workloads.

Figure 10-60 Adding a log policy



Step 3 Set **Volume Type** to **hostPath** or **emptyDir**.

Table 10-43 Configuring log policies

Parameter	Description
Volume Type	• **hostPath**: A host path is mounted to the specified container path (mount path). In the node host path, you can view the container logs output into the mount path. • **emptyDir**: A temporary path of the node is mounted to the specified path (mount path). Log data that exists in the temporary path but is not reported by the collector to AOM will disappear after the pod is deleted.
hostPath	Enter a host path, for example, **/var/paas/sys/log/nginx**.
Mount Path	Container path (for example, **/tmp**) to which the storage resources will be mounted. **NOTICE** • Do not mount storage to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. • If the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host may be damaged. • AOM collects only the first 20 logs that have been modified recently. It collects logs from 2 levels of subdirectories by default. • AOM only collects **.log**, **.trace**, and **.out** text logs in mounting paths. • For details about how to set permissions for mount points in a container, see **Configure a Security Context for a Pod or Container**.
Extended Host Path	This parameter is mandatory only if **Volume Type** is set to **HostPath**. Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted. A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod. • **None**: No extended path is configured. • **PodUID**: ID of a pod. • **PodName**: name of a pod. • **PodUID/ContainerName**: ID of a pod or name of a container. • **PodName/ContainerName**: name of a pod or container.

Parameter	Description
Collection Path	A collection path narrows down the scope of collection to specified logs. • If no collection path is specified, log files in **.log**, **.trace**, and **.out** formats will be collected from the specified path. • **/Path/**/** indicates that all log files in **.log**, **.trace**, and **.out** formats will be recursively collected from the specified path and all subdirectories at 5 levels deep. • * in log file names indicates a fuzzy match. Example: The collection path **/tmp/**/test*.log** indicates that all **.log** files prefixed with **test** will be collected from **/tmp** and subdirectories at 5 levels deep. **CAUTION** Ensure that **ICAgent** is of version 5.12.22 or later.
Log Dump	Log dump refers to rotating log files on a local host. • **Enabled**: AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped. A new **.zip** file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 **.zip** files. When the number of **.zip** files exceeds 20, earlier **.zip** files will be deleted. • **Disabled**: AOM does not dump log files. **NOTE** • AOM rotates log files using copytruncate. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur. • Currently, mainstream log components such as Log4j and Logback support log file rotation. If you have already set rotation for log files, skip the configuration. Otherwise, conflicts may occur. • You are advised to configure log file rotation for your own services to flexibly control the size and number of rolled files.

Step 4 Click **OK**.

----End

YAML Example (ICAgent)

You can set the container log storage path by defining a YAML file.

As shown in the following figure, an emptyDir volume is mounted a temporary path to **/var/log/nginx**. In this way, the ICAgent collects logs in **/var/log/nginx**. The **policy** field is customized by CCE and allows the ICAgent to identify and collect logs.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
```



```
selector:
  matchLabels:
    app: testlog
template:
  replicas: 1
  metadata:
    labels:
      app: testlog
  spec:
    containers:
      - image: 'nginx:alpine'
        name: container-0
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
          limits:
            cpu: 250m
            memory: 512Mi
        volumeMounts:
          - name: vol-log
            mountPath: /var/log/nginx
            policy:
              logs:
                rotate: ''
    volumes:
      - emptyDir: {}
        name: vol-log
    imagePullSecrets:
      - name: default-secret
```

The following shows how to use a hostPath volume. Compared with emptyDir, the type of **volumes** is changed to **hostPath**, and the path on the host needs to be configured for this hostPath volume. In the following example, **/tmp/log** on the host is mounted to **/var/log/nginx**. In this way, the ICAgent can collects logs in **/var/log/nginx**, without deleting the logs from **/tmp/log**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: testlog
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: testlog
  template:
    metadata:
      labels:
        app: testlog
    spec:
      containers:
        - image: 'nginx:alpine'
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: vol-log
              mountPath: /var/log/nginx
              readOnly: false
              extendPathMode: PodUID
              policy:
                logs:
```



```
            rotate: Hourly
            annotations:
              pathPattern: '**'
              format: ''
  volumes:
    - hostPath:
        path: /tmp/log
      name: vol-log
  imagePullSecrets:
    - name: default-secret
```

Table 10-44 Parameter description

Parameter	Description	Description
extendPathMode	Extended host path	Extended host paths contain pod IDs or container names to distinguish different containers into which the host path is mounted. A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod. ● **None**: No extended path is configured. ● **PodUID**: ID of a pod. ● **PodName**: name of a pod. ● **PodUID/ContainerName**: ID of a pod or name of a container. ● **PodName/ContainerName**: name of a pod or container.
policy.logs.rotate	Log dump	Log dump refers to rotating log files on a local host. ● **Enabled**: AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped immediately. A new **.zip** file is generated in the directory where the log file locates. For a log file, AOM stores only the latest 20 **.zip** files. When the number of **.zip** files exceeds 20, earlier **.zip** files will be deleted. After the dump is complete, the log file in AOM will be cleared. ● **Disabled**: AOM does not dump log files. **NOTE** ● AOM rotates log files using copytruncate. Before enabling log dumping, ensure that log files are written in the append mode. Otherwise, file holes may occur. ● Currently, mainstream log components such as Log4j and Logback support log file rotation. If you have already set rotation for log files, skip the configuration. Otherwise, conflicts may occur. ● You are advised to configure log file rotation for your own services to flexibly control the size and number of rolled files.

Parameter	Description	Description
policy.logs.annotations.pathPattern	Collection path	A collection path narrows down the scope of collection to specified logs. • If no collection path is specified, log files in **.log**, **.trace**, and **.out** formats will be collected from the specified path. • **/Path/**/** indicates that all log files in **.log**, **.trace**, and **.out** formats will be recursively collected from the specified path and all subdirectories at 5 levels deep. • * in log file names indicates a fuzzy match. Example: The collection path **/tmp/**/test*.log** indicates that all **.log** files prefixed with **test** will be collected from **/tmp** and subdirectories at 5 levels deep. **CAUTION** Ensure that **ICAgent** is of version 5.12.22 or later.
policy.logs.annotations.format	Multi-line log matching	Some programs (for example, Java program) print a log that occupies multiple lines. By default, logs are collected by line. If you want to display logs as a single log message, you can enable multi-line logging and use the log time or regular pattern mode. When a line of log message matches the preset time format or regular expression, it is considered as the start of a log message and the next line starts with this line of log message is considered as the end identifier of the log message. The format is as follows: `{ "multi": { "mode": "time", "value": "YYYY-MM-DD hh:mm:ss" } }` **multi** indicates the multi-line mode. • **time**: log time. Enter a time wildcard. For example, if the time in the log is 2017-01-01 23:59:59, the wildcard is YYYY-MM-DD hh:mm:ss. • **regular**: regular pattern. Enter a regular expression.

Viewing Logs

After a log collection path is configured and the workload is created, the ICAgent collects log files from the configured path. The collection takes about 1 minute.

After the log collection is complete, go to the workload details page and click **Logs** in the upper right corner to view logs.

You can also view logs on the AOM console.

You can also run the **kubectl logs** command to view the container stdout.

```
# View logs of a specified pod.
kubectl logs <pod_name>
kubectl logs -f <pod_name> # Similar to tail -f

# View logs of a specified container in a specified pod.
kubectl logs <pod_name> -c <container_name>

kubectl logs pod_name -c container_name -n namespace (one-off query)
kubectl logs -f <pod_name> -n namespace (real-time query in tail -f mode)
```

10.5.3 Collecting Kubernetes Events

The Cloud Native Log Collection add-on works with LTS to collect and store Kubernetes events and works with AOM to generate alarms.

Reporting Kubernetes Events to LTS

To enable Kubernetes event collection in different scenarios, take the following steps.

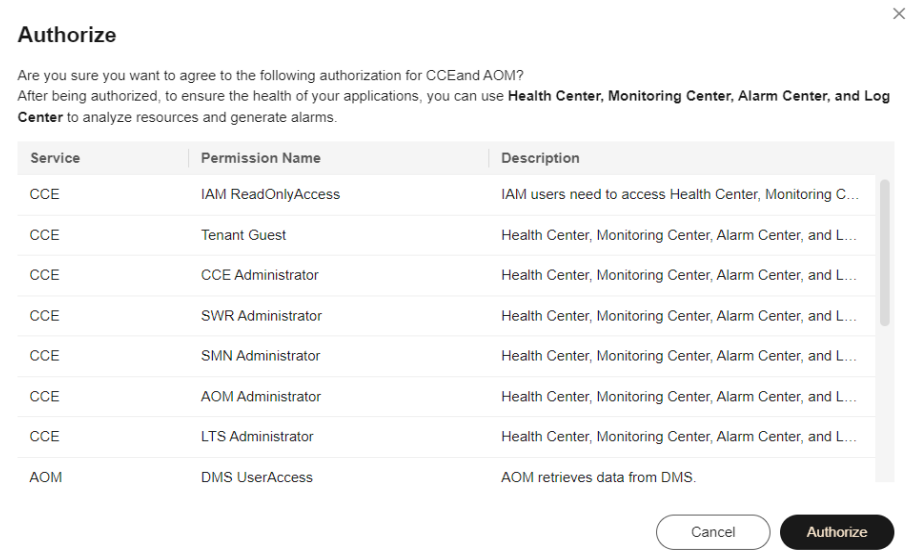
Logging Is Not Enabled for a Cluster

When enabling logging, you can select Kubernetes events to create a default log collection policy, so that all Kubernetes events are collected and reported to LTS.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. (Optional) If you are not authorized, obtain required permissions first.

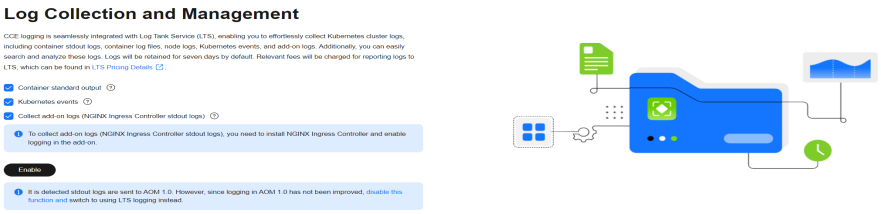
   In the displayed dialog box, click **Authorize**.

   Figure 10-61 Authorize



3. Click **Enable** and wait for about 30 seconds until the log page is automatically displayed.

Figure 10-62 Enable



- Stdout logs: A log collection policy named **default-stdout** will be created, which will report stdout logs from all namespaces to LTS.
- Kubernetes events: A log collection policy named **default-event** will be created, which will report Kubernetes events from all namespaces to LTS.
- To collect add-on logs (stdout logs of NGINX Ingress Controller), you need to install NGINX Ingress Controller and enable logging for this add-on.

  After logging is enabled, a log policy named **default-nginx-ingress** will be created to collect the stdout logs of NGINX Ingress Controller and report the logs to LTS.

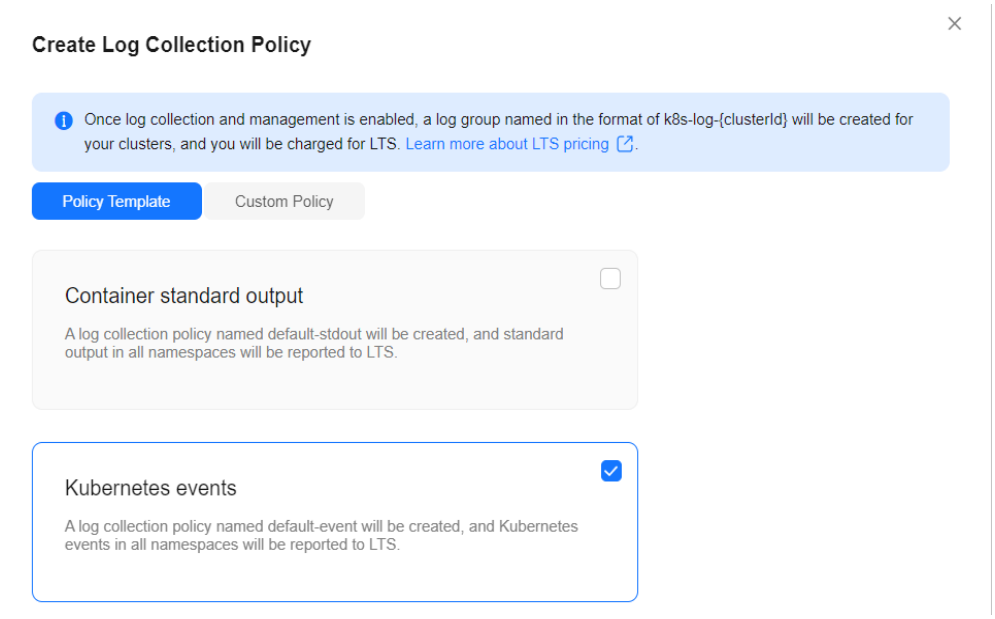
Logging Has Been Enabled for a Cluster

If logging has been enabled for a cluster but Kubernetes event collection has not been enabled, or the corresponding log collection policy has been deleted, you can manually create a log collection policy by taking the following steps:

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. In the upper right corner, click **View Log Collection Policies**.

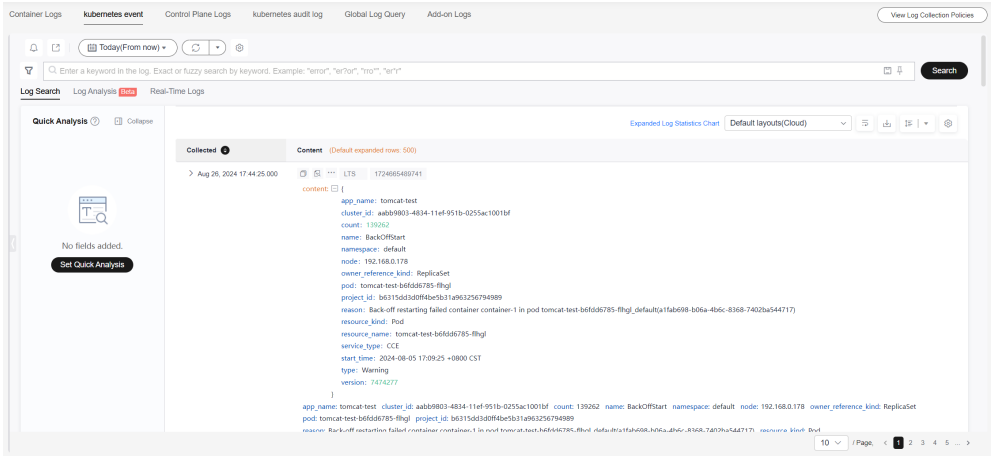
   All log collection policies are displayed.
3. Click **Create Log Collection Policy**. Then, select **Kubernetes events** and click **OK**.

Figure 10-63 Creating a log collection policy



4. (After the creation is complete, you can view logs on the **Logging** page.) Select the log stream configured in the log collection policy to view the events reported to LTS.

Figure 10-64 Viewing event details



Reporting Kubernetes Events to AOM

In 1.3.2 and later versions, the Cloud Native Log Collection add-on reports all warning events and some normal events to AOM by default. The reported events can be used to configure alarms. If the cluster version is 1.19.16, 1.21.11, 1.23.9, 1.25.4, or later, after the Cloud Native Log Collection add-on is installed, events will be reported to AOM by this add-on instead of the control plane component. After this add-on is uninstalled, events will not be reported to AOM.

Custom Event Reporting

If the reported events cannot meet requirements, you can modify the settings for the events.

Using the CCE Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings**.

Step 2 Click the **Monitoring** tab. In the **Log Configuration** area, modify the policy for reporting Kubernetes events to AOM.

- **Abnormal events**: This option is enabled by default. All abnormal events are reported to AOM. You can click **Configure Blocklist** to add events that do not need to be reported to the blocklist. You can obtain event names in **CCE Events**.
- **Normal events**: If this option is enabled, normal events will be reported to AOM. The system is pre-configured to report some normal events. If you need to customize the events to be reported, click **Configure Trustlist** to add the events to the trustlist. You can obtain event names in **CCE Events**.

Step 3 Click **Confirm configuration**.

----End

Using kubectl

Step 1 Run the following command on the cluster to modify the event collection settings:

kubectl edit logconfig -n kube-system default-event-aom

Step 2 Modify the event collection settings as required.

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    # Settings on CCE from which events are collected
    type: event    # Type of logs to be collected from CCE. Do not change the value.
    event:
      normalEvents:    # Used to configure normal events
        enable: true    # Whether to enable normal event collection
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all events will be collected.
        - ScaleDown
      warningEvents:    # Used to configure warning events
        enable: true    # Whether to enable warning event collection
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all events will be collected.
        - ScaleDown
  outputDetail:
    type: AOM    # Type of the system that receives the events. Do not change the value.
    AOM:
      events:
```



```
    - name: DeleteNodeWithNoServer    # Event name. This parameter is mandatory.
      resourceType: Namespace    # Type of the resource that operations are performed on.
      severity: Major    # Event severity after an event is reported to AOM, which can be Critical, Major, Minor, or Info. The default value is Major.
```

----End

10.5.4 Collecting NGINX Ingress Controller Logs

The Cloud Native Log Collection add-on can collect the logs of the NGINX Ingress Controller add-on. You can use the logs to analyze traffic changes and obtain service traffic characteristics for service decision making.

Constraints

- The **NGINX Ingress Controller** add-on of version 2.2.82 or later, 2.6.32 or later, or 3.0.8 or later must been installed in the cluster.
- The **Cloud Native Log Collection** add-on of version 1.6.0 or later must be installed in the cluster.

Billing

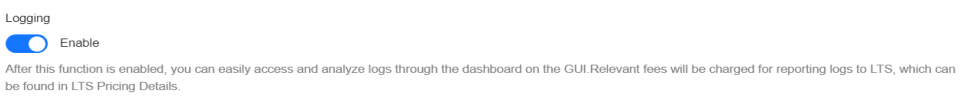
LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota. For details, see **Price Calculator**.

Step 1: Enabling Log Collection

The NGINX Ingress Controller add-on has been installed in the cluster, with log collection enabled.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.
2. Find the **NGINX Ingress Controller** add-on and enable **Logging**.
   - If the add-on has been installed, click **Manage**, locate the installed add-on instance, and click **Edit**. Then, enable **Logging**.

     If multiple add-on instances are installed in the cluster, you need to modify the configuration of each add-on instance to collect logs of all instances.
   - If the add-on is not installed, click **Install**. Then, enable **Logging**. Set other parameters as required. For details, see **NGINX Ingress Controller**.



Step 2: Collecting NGINX Ingress Controller Logs in Logging

To enable log collection for the NGINX Ingress Controller add-on in different scenarios, take the following steps.

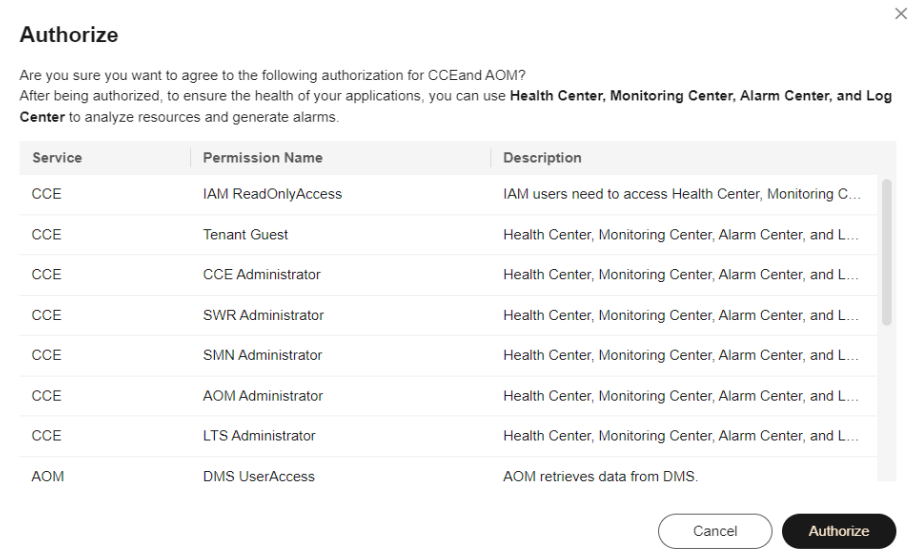
Logging Is Not Enabled for a Cluster

If logging is not enabled for a cluster, you can select **Collect add-on logs (NGINX Ingress Controller stdout logs)** when enabling logging to create the default log collection policy for the NGINX Ingress Controller add-on.

1. On the CCE console, click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. If you are not authorized, obtain required permissions first. Otherwise, skip this step.

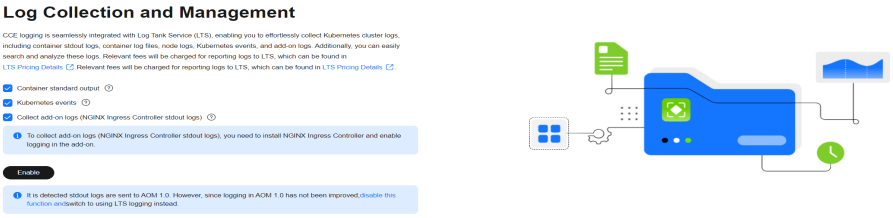
   In the displayed dialog box, click **Authorize**.

   Figure 10-65 Adding authorization



3. On the page that is displayed, select the type of logs to be collected and click **Enable**. Wait for about 30 seconds. The new page is automatically displayed.

   Figure 10-66 Enabling log collection



   - Stdout logs: You can enable this option as needed. After it is enabled, a log collection policy named **default-stdout** will be created, which will report stdout logs from all namespaces to LTS.
   - Kubernetes events: You can enable this option as needed. After it is enabled, a log collection policy named **default-event** will be created, which will report Kubernetes events from all namespaces to LTS.
   - NGINX Ingress Controller stdout logs: This option is mandatory. You need to install the NGINX Ingress Controller add-on and enable logging for it.

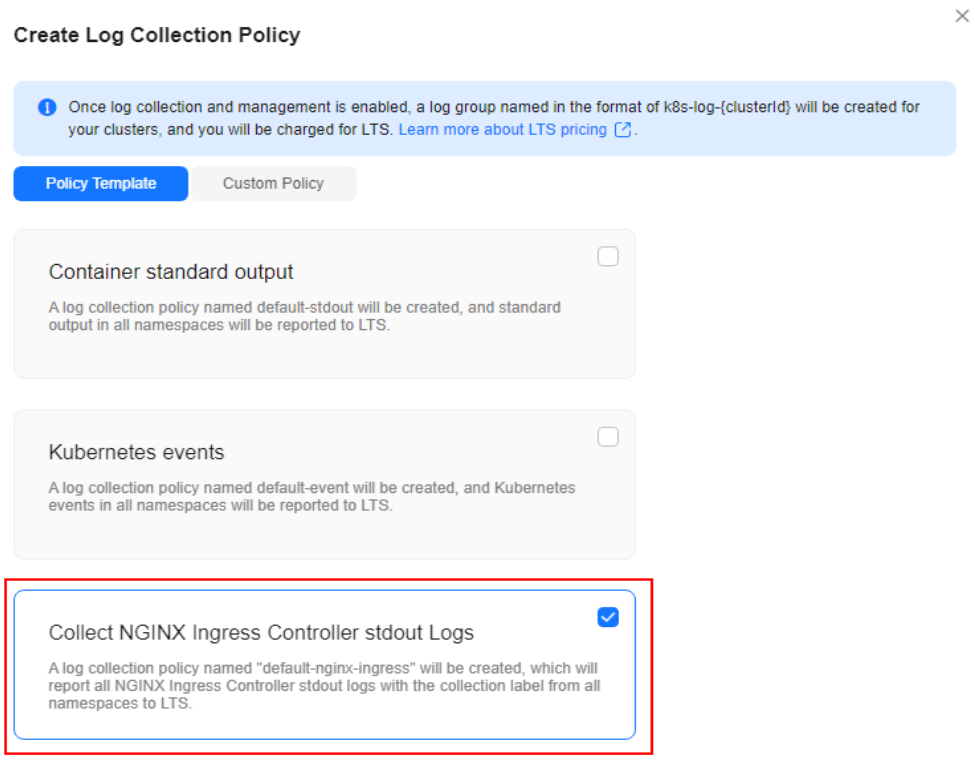
After logging is enabled, a log policy named **default-nginx-ingress** will be created to collect stdout logs of NGINX Ingress Controller and report the logs to LTS.

Logging Has Been Enabled for a Cluster

If logging has been enabled for a cluster but log collection is not enabled for the NGINX Ingress Controller add-on, you can manually create a log collection policy.

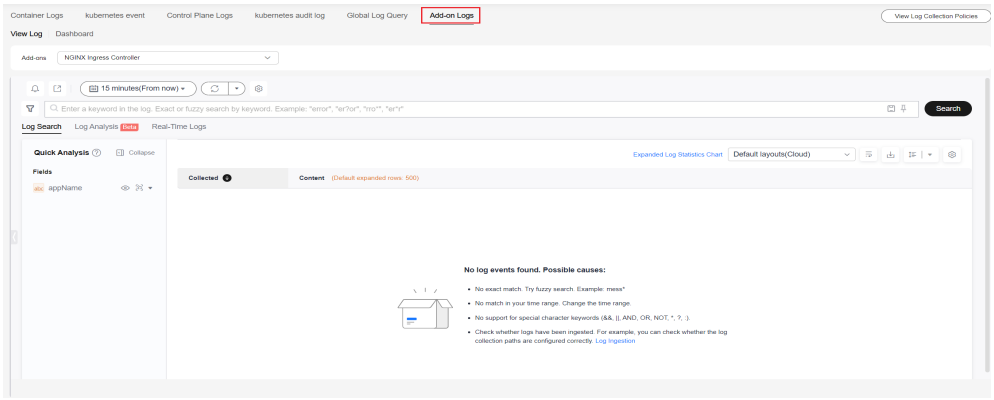
1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. In the upper right corner, click **View Log Collection Policies**. All log collection policies in the current cluster are displayed.
3. Click **Create Log Collection Policy**. Then, select **Collect NGINX Ingress Controller stdout Logs** and click **OK**.

   Figure 10-67 Creating a log collection policy



4. (The system automatically creates a log collection policy named **default-nginx-ingress**.) Go to the **Logging** page and click the **Add-on Logs** tab to view the logs reported by the add-on to LTS.

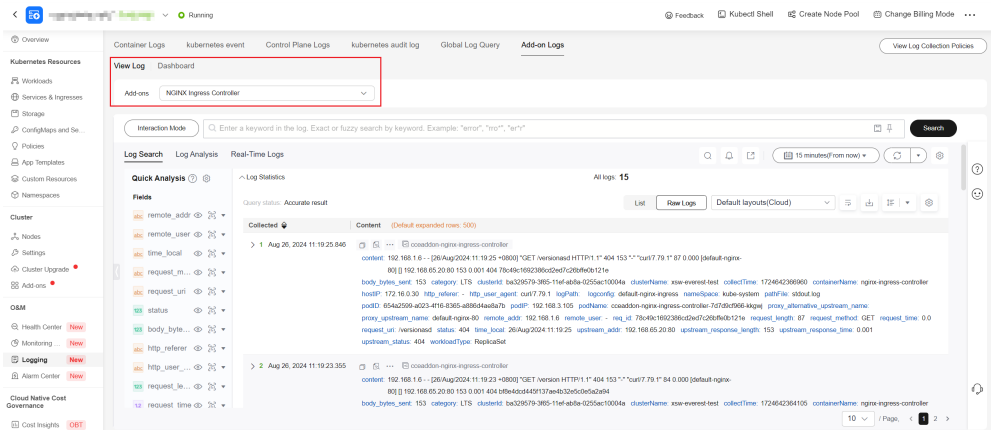
Figure 10-68 Viewing logs



Step 3: Viewing NGINX Ingress Controller Logs

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Add-ons Log** tab and select the **NGINX Ingress Controller** add-on. For details about related operations, see **LTS User Guide**.

Figure 10-69 Viewing NGINX Ingress Controller logs



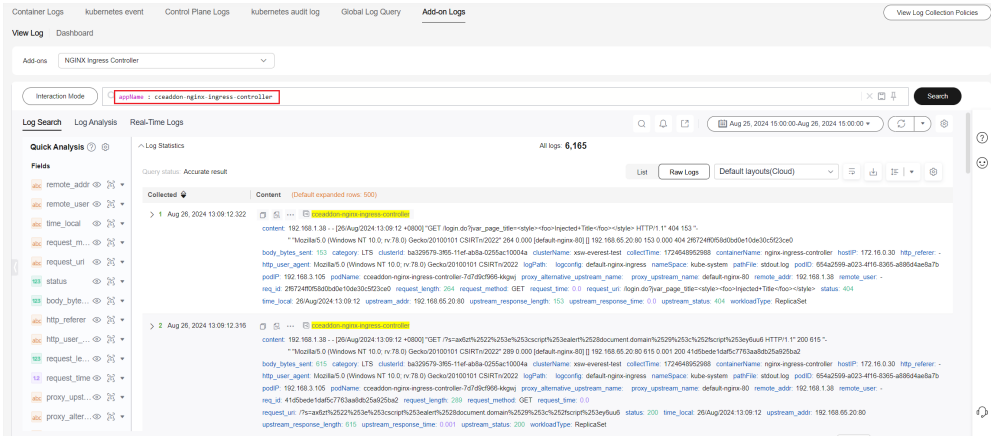
3. (If multiple add-on instances are installed in the cluster and log collection has been enabled for these instances, logs of all instances will be displayed.) Search for logs by **appName**.

   The options of **appName** are as follows:

   - If the NGINX Ingress Controller name is **nginx**, the value of **appName** is **cceaddon-nginx-ingress-controller**.
   - If the NGINX Ingress Controller name is user-defined, for example, *{className}*, the value of **appName** is **cceaddon-nginx-ingress-*{className}*-controller**.

   The following are example logs of **cceaddon-nginx-ingress-controller**:

   ```
   appName : cceaddon-nginx-ingress-controller
   ```

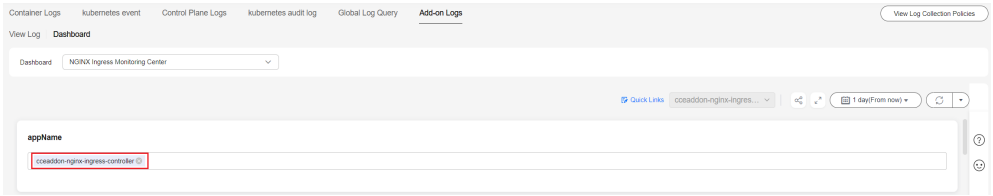
Step 4: Viewing Dashboards of NGINX Ingress

CCE can collect the logs of NGINX Ingress and work with LTS for multi-dimensional analysis. It also provides structured logging and three types of dashboards (monitoring center, access center, and second-by-second monitoring) to meet monitoring requirements in different scenarios.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Add-on Logs** tab and then the **Dashboard** tab. On the displayed tab, you can select different dashboard templates.
3. (If multiple add-on instances are installed in the cluster and log collection has been enabled for these instances, logs of all instances will be displayed.) Search for logs by **appName**.

   The options of **appName** are as follows:

   - If the NGINX Ingress Controller name is **nginx**, the value of **appName** is **cceaddon-nginx-ingress-controller**.
   - If the NGINX Ingress Controller name is user-defined, for example, *{className}*, the value of **appName** is **cceaddon-nginx-ingress-*{className}*-controller**.

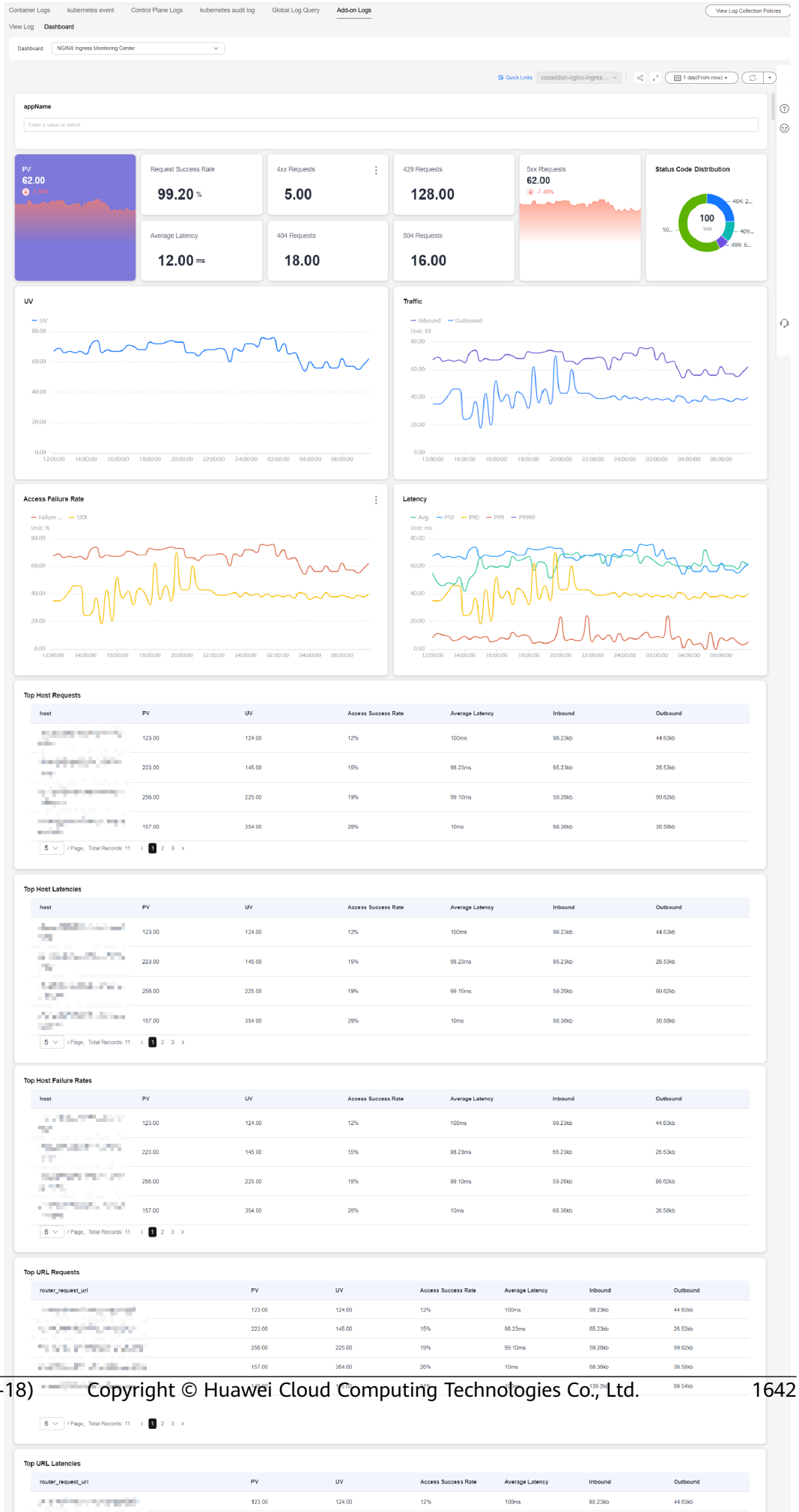


Monitoring Center Dashboard

This dashboard displays basic information about NGINX Ingress in different time dimensions (relative time from now, relative time from last, and specified time), including:

- Access data: PV, UV, request success rate, average latency, the number of 4XX requests, the number of 404 requests, the number of 429 requests, the number of 5XX requests, the number of 504 requests, status code distribution, traffic, access failure rate, and latency

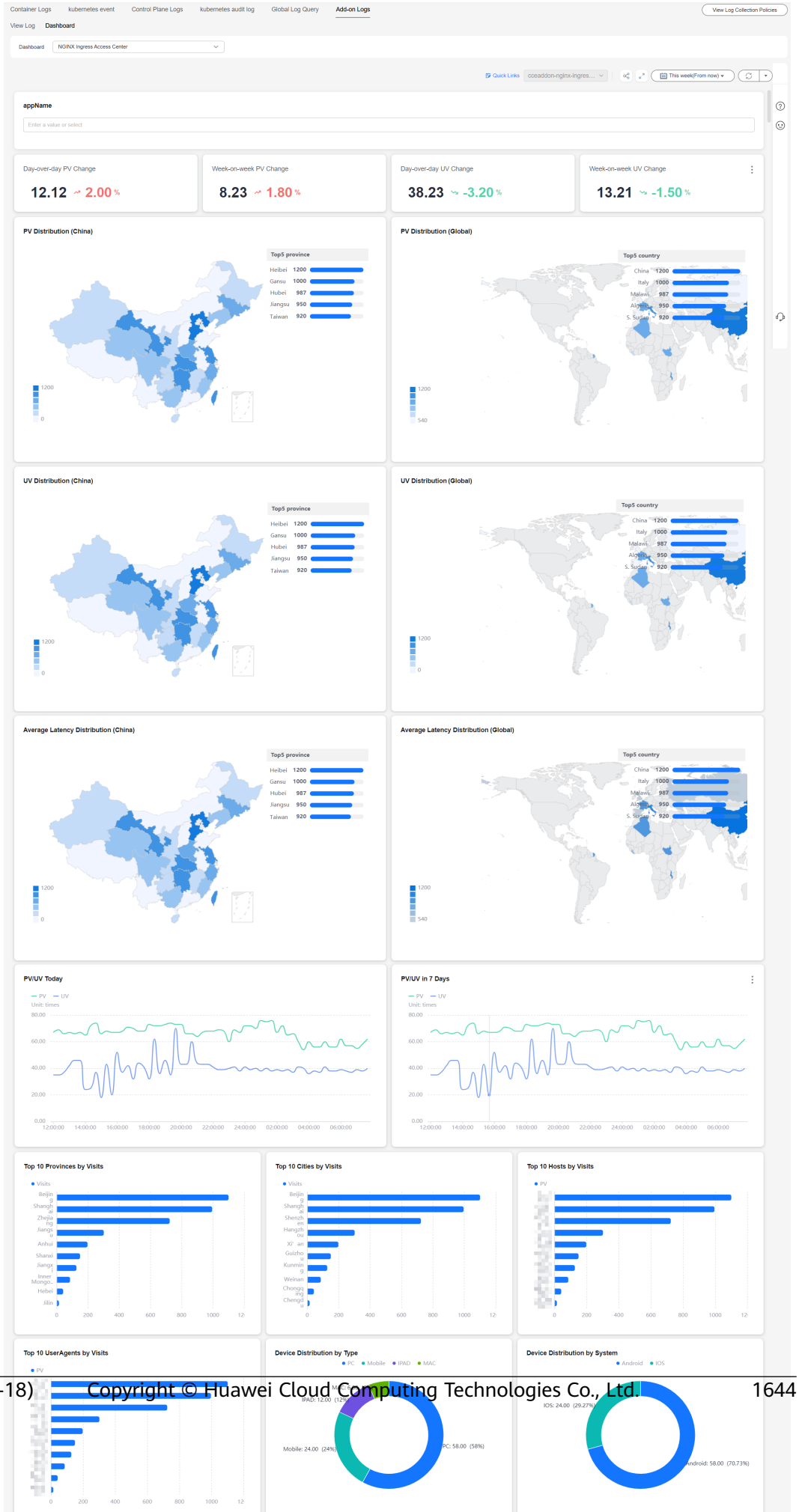
- Top statistics: hosts ranked by requests, hosts ranked by latencies, hosts ranked by failure rates, URLs ranked by requests, URLs ranked by latencies, URLs ranked by failure rates, backend severs ranked by requests, backend severs ranked by latencies, and backend severs ranked by failure rates


Container Logs
kubernetes event
Control Plane Logs
kubernetes audit log
Global Log Query
Add-on Logs
View Log Collection Policies
View Log
Dashboard
Dashboard
NGINX Ingress Monitoring Center
appName
PV
62.00
Request Success Rate
99.20 %
4xx Requests
5.00
429 Requests
128.00
5xx Requests
62.00
Status Code Distribution
Average Latency
12.00 ms
404 Requests
18.00
504 Requests
16.00
UV
Traffic
Access Failure Rate
Latency
Top Host Requests
Top Host Latencies
Top Host Failure Rates
Top URL Requests
Top URL Latencies

Access Center Dashboard

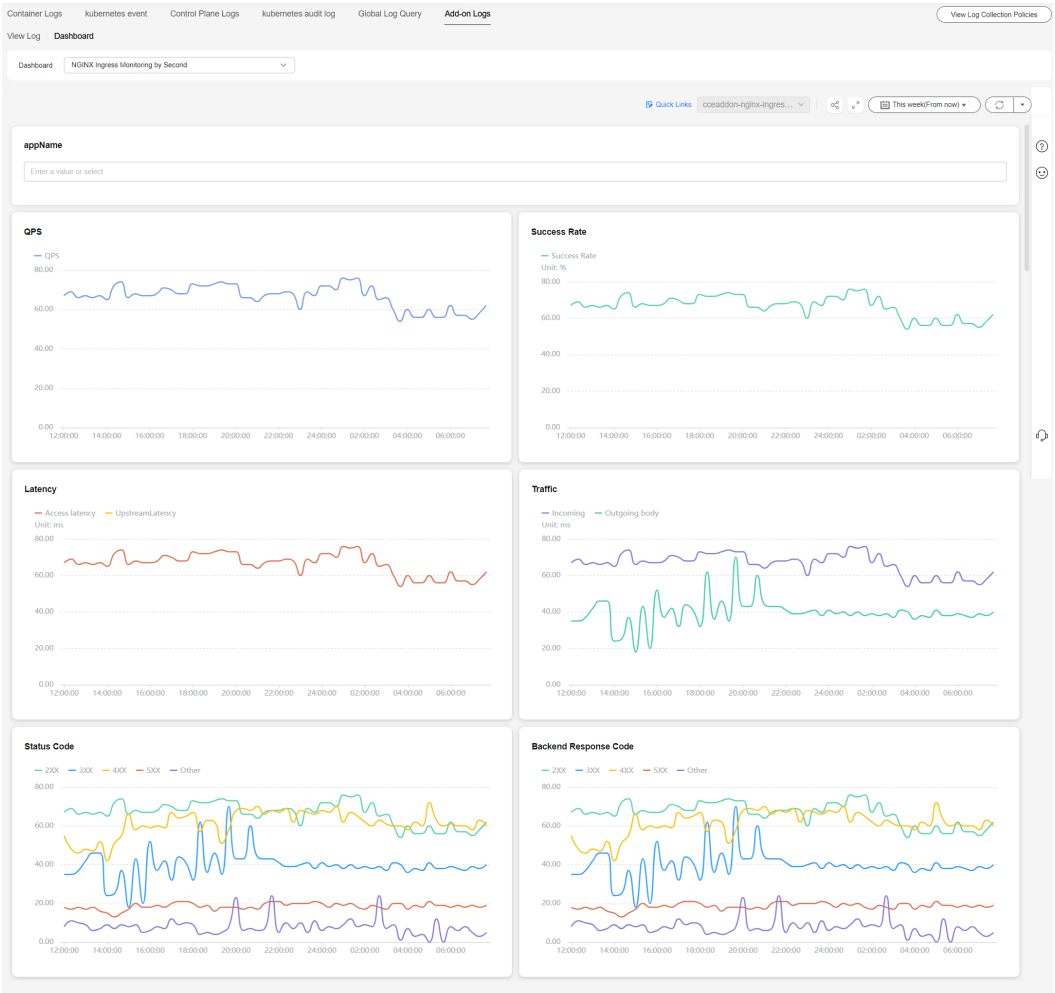
This dashboard displays access request statistics in different time dimensions (relative time from now, relative time from last, and specified time), including:

- Data comparison: day-over-day PV, week-over-week PV, day-over-day UV, week-over-week UV, PV/UV today, and PV/UV in past seven days
- Data distribution: PV distribution (China), PV distribution (global), UV distribution (China), UV distribution (global), average latency distribution (China), average latency distribution (global), device proportion (by terminal), and device proportion (by system)
- Top statistics: top 10 provinces with the most requests, top 10 cities with the most requests, top 10 hosts with the most requests, top 10 UserAgents with the most requests, top URLs with the most requests, and top IP addresses with the most requests



Second-by-Second Monitoring Dashboard

This dashboard displays NGINX Ingress traffic and application performance in real time, including QPS, success rates, latencies, traffic, status codes, and backend response codes.



10.5.5 Collecting Control Plane Component Logs

CCE supports logging for master nodes. On the **Logging** page, you can select one or more control plane components (kube-controller-manager, kube-apiserver, and kube-scheduler) whose logs need to be reported.

Constraints

- The cluster version must be v1.21.7-r0 or later, v1.23.5-r0 or later, or 1.25.
- There is required LTS resource quota. For details about the default LTS quota, see **Basic Resources**.

Control Plane Components

There are three control plane log types. Each log stream corresponds to a component of the Kubernetes control plane. To learn more about these components, see **Kubernetes Components**.

Table 10-45 Control plane components

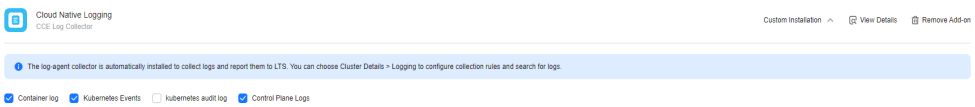
Log Type	Component	Log Stream	Description
Control plane component logs	kube-apiserver	kube-apiserver-{{clusterID}}	It exposes Kubernetes APIs. For more information, see **kube-apiserver**.
	kube-controller-manager	kube-controller-manager-{{clusterID}}	It manages controllers and embeds the core control loops shipped with Kubernetes. For more information, see **kube-controller-manager**.
	kube-scheduler	kube-scheduler-{{clusterID}}	It manages when and where to run Pods in your cluster. For more information, see **kube-scheduler**.

Enabling Control Plane Logging

Enabling control plane logging during cluster creation

1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**. Then, configure the parameters and click **Next: Select Add-on**.
3. On the displayed page, select **Cloud Native Log Collection** and click **Next: Add-on Configuration**.
4. On the displayed page, select **Control Plane Logs** for **Cloud Native Log Collection**.

   Figure 10-70 Enabling control plane logging during cluster creation

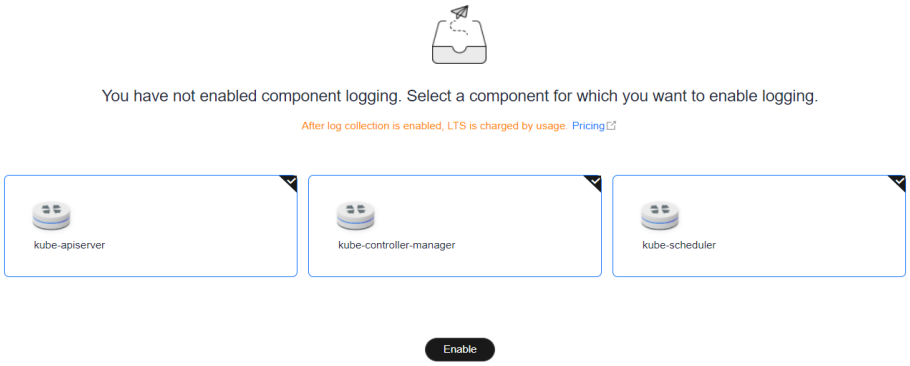
   

5. Click **Next: Confirm Configuration**.

Enabling control plane logging for an existing cluster

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab, select one or more control plane components whose logs need to be collected, and click **Enable**.

Figure 10-71 Selecting control plane components

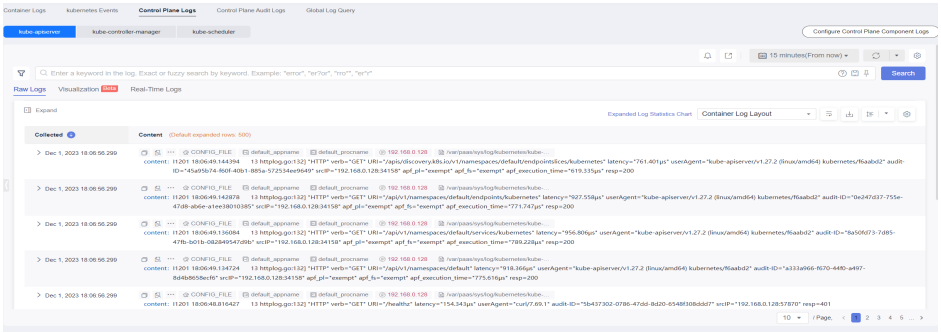


Viewing Control Plane Component Logs of the Target Cluster

Viewing control plane component logs of the target cluster on the CCE console

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab and select the topic of logs to be viewed. For details about available control plane log types, see **Control Plane Components**. For details about related operations, see **LTS User Guide**.

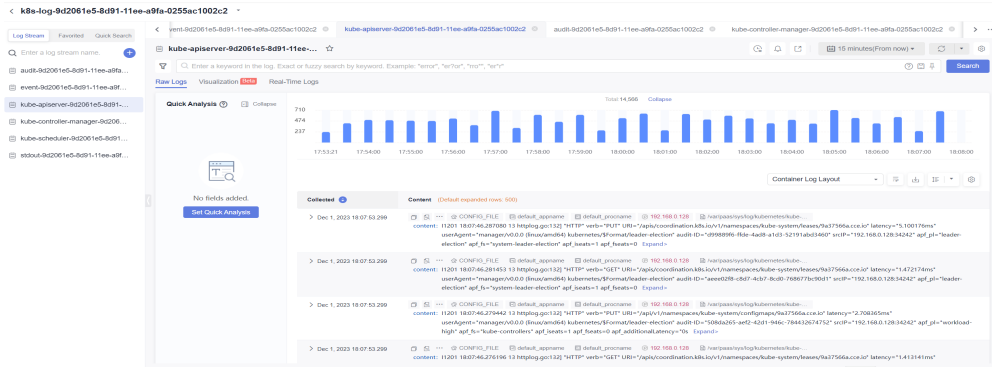
Figure 10-72 Viewing control plane component logs



Viewing control plane component logs of the target cluster on the LTS console

1. Log in to the LTS console and choose **Log Management**.
2. Search for the log group by cluster ID and click the log group name to view the log streams. For details, see **LTS User Guide**.

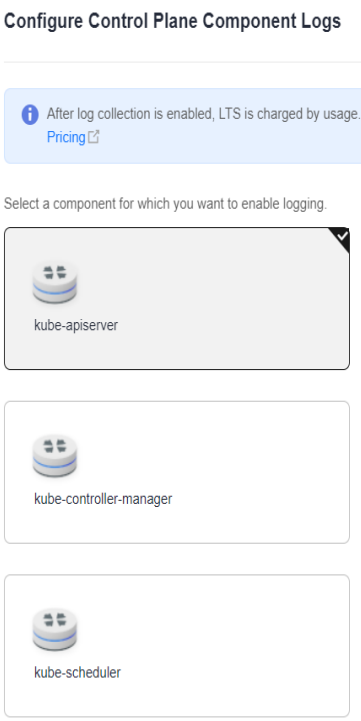
Figure 10-73 Viewing control plane component logs on the LTS console



Disabling Control Plane Logging

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Logs** tab. In the upper right corner, click **Configure Control Plane Component Logs**. Then, modify the log setting.

Figure 10-74 Disabling control plane logging



3. Determine whether to enable logging for each component. If yes, click **OK**.

NOTE

After you disable control plane logging, logs are no longer written to the original log stream, but the existing logs will not be deleted and expenses may be incurred for this.

10.5.6 Collecting Kubernetes Audit Logs

CCE supports logging for master nodes. On the **Control Plane Audit Logs** tab of **Logging**, you can determine whether to report Kubernetes audit logs to LTS.

Constraints

- The cluster version must be v1.21.7-r0 or later, v1.23.5-r0 or later, or 1.25.
- There is required LTS resource quota. For details about the default LTS quota, see **Basic Resources**.

Kubernetes Audit Logs

Table 10-46 Kubernetes audit logs

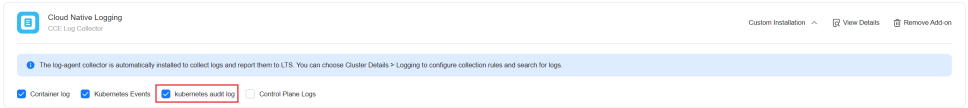
Log Type	Component	Log Stream	Description
Control plane audit logs	audit	audit-{{clusterID}}	An audit log is a chronological record of user operations on Kubernetes APIs and control plane activities for security.

Enabling Control Plane Audit Logging

Enabling control plane audit logging during cluster creation

1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**. Then, configure the parameters and click **Next: Select Add-on**.
3. On the displayed page, select **Cloud Native Log Collection** and click **Next: Add-on Configuration**.
4. On the displayed page, select **Kubernetes Audit Logs** for **Cloud Native Log Collection**.

   Figure 10-75 Enabling audit logging during cluster creation

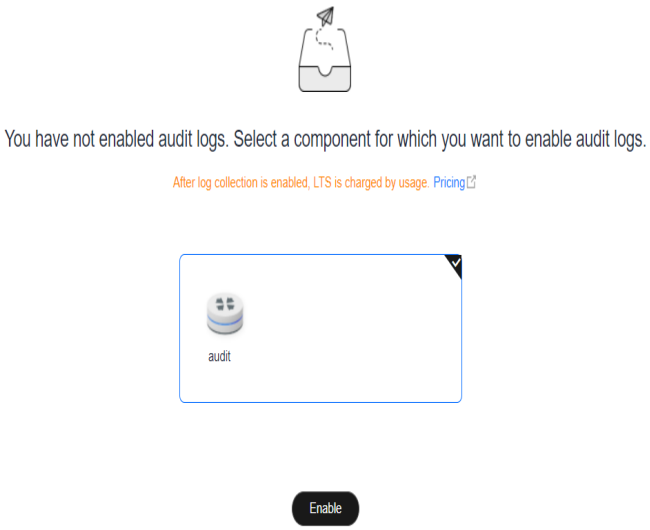
   

5. Click **Next: Confirm Configuration**.

Enabling control plane audit logging for an existing cluster

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab, select the audit component, and click **Enable**.

Figure 10-76 Enabling control plane audit logging for an existing cluster

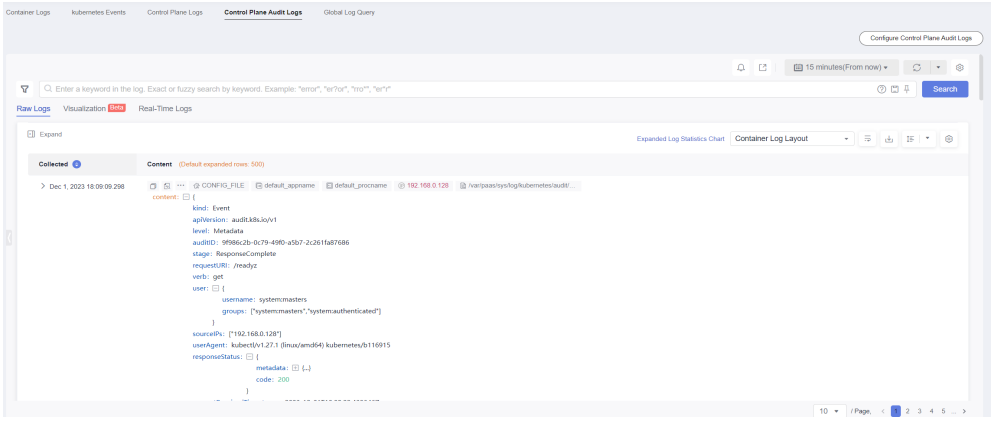


Viewing Control Plane Audit Logs

Viewing control plane audit logs on the CCE console

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab to view audit logs in the cluster. For details about related operations, see **LTS User Guide**.

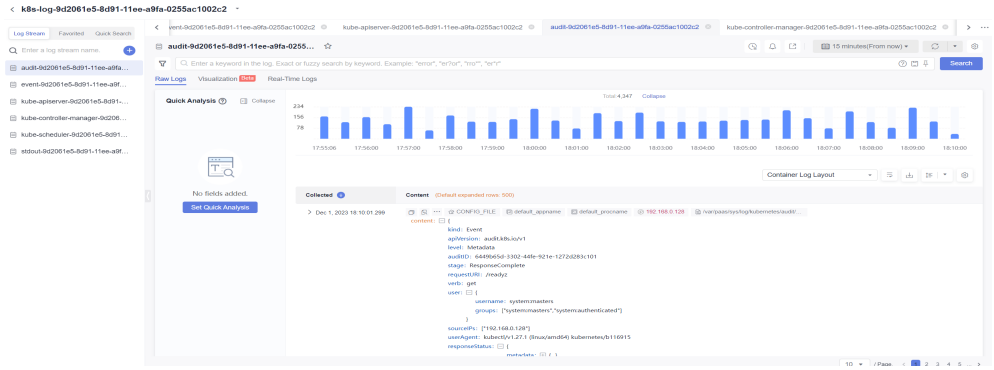
Figure 10-77 Viewing control plane audit logs on the CCE console



Viewing control plane audit logs on the TLS console

1. Log in to the LTS console and choose **Log Management**.
2. Search for the log group by cluster ID and click the log group name to view the log streams. For details, see **LTS User Guide**.

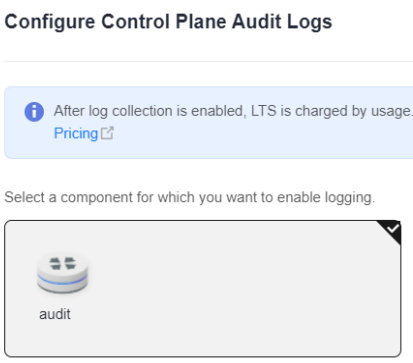
Figure 10-78 Viewing control plane audit logs on the TLS console



Disabling Control Plane Audit Logging

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab. In the upper right corner, click **Configure Control Plane Audit Logs**. Then, determine whether to enable control plane audit logging.

Figure 10-79 Disabling control plane audit logging



3. Deselect **audit** and click **OK**.

NOTE

After you disable control plane audit logging, logs are no longer written to the original log stream, but the existing logs will not be deleted and expenses may be incurred for this.

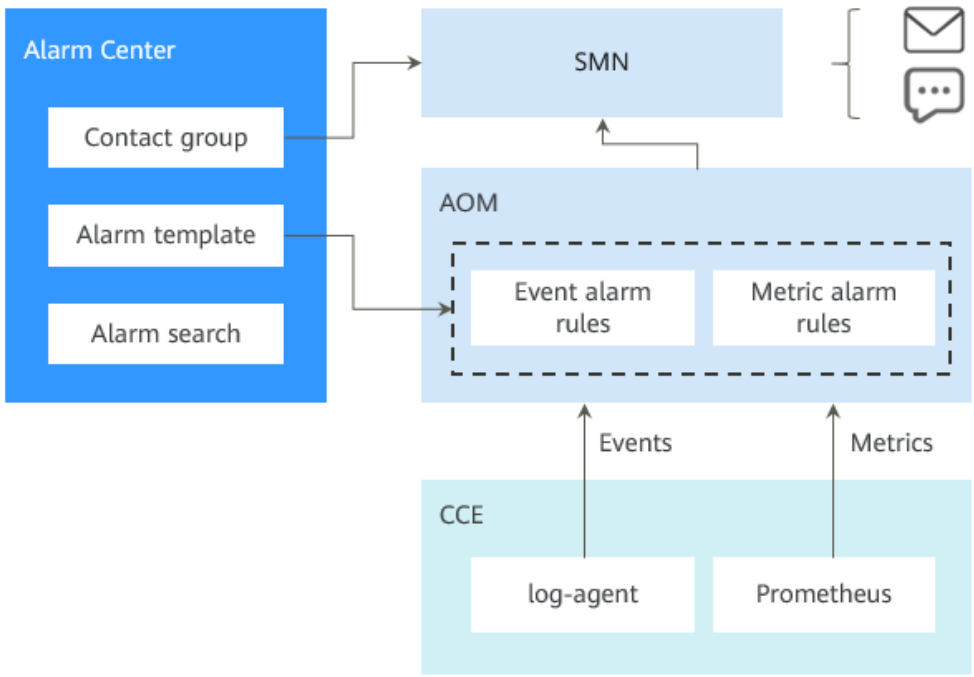
10.6 Alarm Center

10.6.1 Overview

Alarm reporting is an essential aspect of observability. In addition to traditional resource usage alarms (such as CPU and memory usage alarms), there are custom monitoring metric alarms (such as container restart alarms and application access failure alarms).

CCE works with AOM for metric and event alarm reporting and provides Alarm Center that allows you to quickly configure and view common alarms (such as resource usage alarms).

Figure 10-80 Alarm Center architecture



- Alarm Center

  Based on the alarm capabilities of AOM, Alarm Center provides quick alarm search and configuration for clusters. You can use Alarm Center to configure common alarm rules with just a few clicks.

- AOM

  AOM is a one-stop, multi-dimensional O&M management platform for monitoring and alarm reporting of cloud applications.

- Simple Message Notification

  Simple Message Notification (SMN) connects cloud applications. After events or alarms are triggered, SMN will send notifications. In cloud native scenarios, alarms triggered by AOM are sent by SMS message, email, or HTTP message configured on SMN.

10.6.2 Configuring Alarms in Alarm Center

By using AOM, Alarm Center can promptly detect cluster faults and generate alarms for service stability. Alarm Center provides built-in alarm rules, which can free you from manually configuring alarm rules on AOM. These rules are established based on the extensive cluster O&M experience of our Huawei Cloud container team and can cover container service exceptions, key metric alarms of basic cluster resources, and metric alarms of applications in a cluster to meet your routine O&M requirements.

Constraints

- The cluster version must be v1.17 or later.
- Only Huawei Cloud accounts, HUAWEI IDs, or IAM users with CCE administrator or FullAccess permissions can perform all operations using Alarm Center. IAM users with the CCE ReadOnlyAccess permission can only view all resources.

Enabling Alarm Center

Alarm Center can be enabled for CCE standard clusters and CCE Turbo clusters.

Step 1 Click the cluster name to access the cluster console. In the navigation pane, choose **Alarm Center**.

Step 2 On the **Alarm Rules** tab, click **Enable Alarm Center**. In the window that slides out from the right, select one or more contact groups to manage subscription endpoints and receive alarm messages by group. If no contact group is available, create one by referring to **Binding Contact Groups**.

Step 3 Click **OK**.

NOTE

Metric alarm rules can be created in Alarm Center only after the Cloud Native Cluster Monitoring add-on is installed and the AOM Prometheus instance is interconnected. For details about how to enable Monitoring Center, see **Enabling Monitoring Center**.

The alarm rules that use the problem_gauge metric in **Table 10-47** depend on the CCE Node Problem Detector add-on (**CCE Node Problem Detector**). To use related alarm rules, ensure that the CCE Node Problem Detector add-on has been installed and is running normally.

Event alarms in **Table 10-47** can be reported only when Kubernetes event collection is enabled in Logging. For details, see **Collecting Kubernetes Events**.

----End

Configuring Alarm Rules

After Alarm Center is enabled for CCE standard clusters and CCE Turbo clusters, you can configure and manage alarm rules.

Step 1 Log in to the CCE console.

Step 2 On the cluster list page, click the cluster name to access the cluster console.

Step 3 In the navigation pane, choose **Alarm Center**. Then, click the **Alarm Rules** tab and configure and manage alarm rules.

By default, Alarm Center generates alarm rules for containers. The rules are intended for alarms including event alarms and metric alarms for exceptions. Alarm rules are classified into several sets. You can associate an alarm rule set with multiple contact groups and enable or disable alarm items. An alarm rule set consists of multiple alarm rules. An alarm rule corresponds to the check items for a single exception. **Table 10-47** lists default alarm rules.

----End

Table 10-47 Default alarm rules

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
Load rule set	Abnormal pod	Check whether the pod is running normally.	Metric	Cloud Native Cluster Monitoring	sum(min_over_time(kube_pod_status_phase{phase=~"Pending|Unknown|Failed"}[10m]) and count_over_time(kube_pod_status_phase{phase=~"Pending|Unknown|Failed"}[10m]) > 18)by (namespace,pod, phase, cluster_name, cluster) > 0
	Frequent pod restarts	Check whether the pod frequently restarts.	Metric	Cloud Native Cluster Monitoring	increase(kube_pod_container_status_restarts_total[5m]) > 3
	Unexpected number of Deployment replicas	Check whether the number of Deployment replicas is the same as the expected value.	Metric	Cloud Native Cluster Monitoring	(kube_deployment_spec_replicas != kube_deployment_status_replicas_available) and (changes(kube_deployment_status_replicas_updated[5m]) == 0)
	Unexpected number of StatefulSet replicas	Check whether the number of StatefulSet replicas is the same as the expected value.	Metric	Cloud Native Cluster Monitoring	(kube_statefulset_status_replicas_ready != kube_statefulset_status_replicas) and (changes(kube_statefulset_status_replicas_updated[5m]) == 0)

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Container CPU usage higher than 80%	Check whether the container CPU usage is higher than 80%.	Metric	Cloud Native Cluster Monitoring	100 * (sum(rate(container_cpu_usage_seconds_total{image!="", container!="POD"}[1m])) by (cluster_name,pod,node,namespace,container, cluster) / sum(kube_pod_container_resource_limits{resource="cpu"}) by (cluster_name,pod,node,namespace,container, cluster)) > 80
	Container memory usage higher than 80%	Check whether the container memory usage is higher than 80%.	Metric	Cloud Native Cluster Monitoring	(sum(container_memory_working_set_bytes{image!="", container!="POD"}) BY (cluster_name, node,container, pod , namespace, cluster) / sum(container_spec_memory_limit_bytes > 0) BY (cluster_name, node, container, pod , namespace, cluster) * 100) > 80
	Abnormal container	Check whether the container is running normally.	Metric	Cloud Native Cluster Monitoring	sum by (namespace, pod, container, cluster_name, cluster) (kube_pod_container_status_waiting_reason) > 0
	Load balancer update failed	Check whether a load balancer is updated.	Event	Cloud Native Log Collection	N/A

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Pod OOM	Check whether OOM occurs on the pod.	Event	CCE Node Problem Detector (1.18.41 or later) Cloud Native Log Collection (1.3.2 or later)	PodOOMKilling
Node resource rule set	High usage of Kubernetes PV	Check whether the PV usage on a node is too high.	Metric	Cloud Native Cluster Monitoring	(kubelet_volume_stats_available_bytes{job="kubelet"} / kubelet_volume_stats_capacity_bytes{job="kubelet"}) < 0.03 and kubelet_volume_stats_used_bytes{job="kubelet"} > 0
	Abnormal Kubernetes PVC	Check whether the PVC is normal.	Metric	Cloud Native Cluster Monitoring	kube_persistentvolumeclaim_status_phase{phase=~"Failed|Pending|Lost"} > 0
	Abnormal Kubernetes PV	Check whether the PV is normal.	Metric	Cloud Native Cluster Monitoring	kube_persistentvolume_status_phase{phase=~"Failed|Pending"} > 0
	Node CPU usage higher than 80%	Check whether the node CPU usage is higher than 80%.	Metric	Cloud Native Cluster Monitoring	100 - (avg by(node, cluster_name, cluster) (rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100) > 80

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Available node memory less than 10%	Check whether the available node memory is less than 10%.	Metric	Cloud Native Cluster Monitoring	node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10
	Available node disk space less than 10%	Check whether the available node disk space is less than 10%.	Metric	Cloud Native Cluster Monitoring	avg((node_filesystem_avail_bytes * 100) / node_filesystem_size_bytes) by (device, node, cluster_name, cluster) < 10
	Insufficient node disk space	Check whether the node disk space is sufficient.	Event	Cloud Native Log Collection	N/A
	emptyDir storage pool error	Check whether the node's EV storage pool is functional.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="EmptyDirVolumeGroupStatusError"} >= 1
	Insufficient node memory	Check whether the overall node memory is sufficient.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="MemoryProblem"} >= 1

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	PV storage pool error	Check whether the node's PV storage pool is functional.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="LocalPvVolumeGroupStatusError"} >= 1
	Abnormal node mount point	Check whether the node's mount point is available.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="MountPointProblem"} >= 1
	Insufficient node file handles	Check whether the FD file handles are sufficient.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="FDProblem"} >= 1
	Node disk I/O suspension	Check whether I/O suspension occurs on the node disk.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="DiskHung"} >= 1
	Node disk read-only	Check whether the node disk is read-only.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="DiskReadonly"} >= 1

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Abnormal node disk	Check the usage of the node's system disk and CCE data disks (including Docker and kubelet logical disks).	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="DiskProblem"} >= 1
	Slow node disk I/O	Check whether slow I/O occurs on the node disk.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="DiskSlow"} >= 1
	Insufficient node PIDs	Check whether the PIDs are sufficient.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="PIDProblem"} >= 1
	Node conntrack table full	Check whether the node's conntrack table space is sufficient.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="ConntrackFullProblem"} >= 1

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
Node status rule set	ResolvConf error	Check whether the ResolvConf configuration file is available.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="ResolvConfFileProblem"} >= 1
	Abnormal node CNI component	Check whether the CNI component of the node is running properly.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="CNIProblem"} >= 1
	Abnormal node CRI component	Check the running of the key component CRI (Docker or containerd).	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="CRIProblem"} >= 1
	Node kube-proxy error	Check whether kube-proxy is running properly.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="KUBEPROXYProblem"} >= 1
	Abnormal node kubelet	Check whether kubelet is running normally.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="KUBELETProblem"} >= 1

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Scheduled event on the node	Check whether there is a scheduled event on the node.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="ScheduledEvent"} >= 1
	Unstable node status	Check whether the node status alternates between normal and abnormal.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	sum(changes(kube_node_status_condition{status="true",condition="Ready"}[15m])) by (cluster_name, node, cluster) > 2
	Frequent node containerd restarts	Check whether containerd frequently restarts.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="FrequentContainerdRestart"} >= 1
	Node task suspended	Check whether a task is suspended on the node.	Event	Cloud Native Log Collection	TaskHung
	Incorrect node storage pool configuration	Check whether the node's EV and PV storage pools are correctly configured.	Event	Cloud Native Log Collection	InvalidStoragePool

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Abnormal node	Check whether the node is running normally.	Event	Cloud Native Log Collection	NodeNotReady
	Abnormal node process D	Check whether there is a D state process on the node.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="ProcessD"} >= 1
	Abnormal node process Z	Check whether there is a Z state process on the node.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="ProcessZ"} >= 1
	Frequent node CRI restarts	Check whether CRI frequently restarts.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="FrequentCRIRestart"} >= 1
	Frequent node Docker restarts	Check whether Docker frequently restarts.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="FrequentDockerRestart"} >= 1

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Frequent node kubelet restarts	Check whether kubelet frequently restarts.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="FrequentKubeletRestart"} >= 1
	Node NTP service error	Check whether the node clock synchronization service ntpd or chronyd is running properly.	Metric	Cloud Native Cluster Monitoring CCE Node Problem Detector	problem_gauge{type="NTPProblem"} >= 1
	Processes forcibly stopped due to node OOM	Check whether an OOM event occurred on the node.	Event	CCE Node Problem Detector	OOMKilling
Node scaling rule set	Node pool resources sold out	Check whether the node pool resources are sufficient.	Event	Cloud Native Log Collection	NodePoolSoldOut
	Scale-out timed out	Check whether adding nodes to the node pool timed out.	Event	Cloud Native Log Collection	ScaleUpTimedOut

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Node pool scale-out failed	Check whether an error occurred during a node pool scale-out.	Event	Cloud Native Log Collection	FailedToScaleUpGroup
	Node pool scale-in failed	Check whether an error occurred during a node pool scale-in.	Event	Cloud Native Log Collection	ScaleDownFailed
Cluster status rule set	Unavailable cluster	Check whether the cluster is available.	Event	Cloud Native Log Collection	N/A

Binding Contact Groups

NOTE

An alarm rule can be bound to a maximum of five contact groups.

A contact group, backed on **Simple Message Notification**, enables message publishers and subscribers to contact each other. A contact group contains one or more endpoints. You can attach an alarm rule to a contact group to manage endpoints that have subscribed to alarm messages.

Step 1 Log in to the CCE console.

Step 2 On the cluster list page, click the cluster name to access the cluster console.

Step 3 In the navigation pane, choose **Alarm Center**. Then, click the **Default Contact Groups** tab.

Step 4 Click **Bind Contact Group**. You can select a contact group created in SMN or create a contact group. The parameters for creating a contact group are described as follows:

- **Contact Group Name**: Enter the name of the contact group, which cannot be changed after the contact group is created. The name can contain 1 to 255 characters and must start with a letter or digit. Only letters, digits, hyphens (-), and underscores (_) are allowed.
- **Alarm Message Display Name**: Enter the title of the message received by the specified subscription endpoint. For example, if you set **Terminal Type** to **Email** and specify a display name, the name you specified will be displayed as

the alarm message sender. If no alarm message display name is specified, the sender will be **username@example.com**. The alarm message display name can be changed after a contact group is created.

- **Add Subscription Terminal**: Add one or more endpoints to receive alarm messages. The endpoint type can be **SMS** or **Email**. If you select **SMS**, enter a valid mobile number. If you select **Email**, enter a valid email address.

Step 5 Click **OK**.

You will be redirected to the contact group list. The subscription endpoint is in the **Unconfirmed** state. Send a subscription request to the endpoint to verify the validity of the endpoint.

Step 6 Click **Request Confirmation** in the **Operation** column to send a subscription request to the endpoint. After the endpoint receives and confirms the request, the subscription endpoint status changes to **Confirmed**.

----End

Viewing Alarms

You can view the latest historical alarms on the **Alarms** tab.

Step 1 Log in to the CCE console.

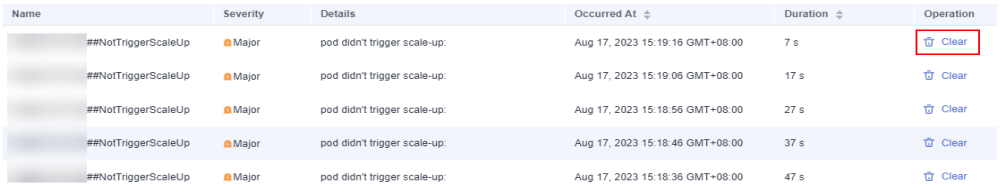
Step 2 On the cluster list page, click the cluster name to access the cluster console.

Step 3 In the navigation pane, choose **Alarm Center**. Then, click the **Alarms** tab.

By default, all alarms to be cleared are displayed in the list. You can query alarms by alarm keyword, alarm severity, or alarm time. In addition, you can view the distribution of alarms that meet the specified criteria in different periods.

If an alarm to be cleared is not triggered within 10 minutes, the alarm is considered cleared by default and converted to a historical alarm. If you confirm that an alarm has been handled in advance, you can also click **Clear** in the **Operation** column. You can view this cleared alarm in the historical alarm list.

Figure 10-81 Querying alarms



----End

10.6.3 Configuring Custom Alarms on CCE

If the default alarm rules cannot meet your requirements, you can create alarm rules on CCE. Based on the alarm rules, you can check whether resources in clusters are normal in a timely manner.

Adding Metric Alarms

NOTE

- To create Prometheus metric threshold-crossing alarm rules and metric alarm rules, you need to enable Monitoring Center. For details, see **Enabling Monitoring Center**.
- Some metric templates are created based on the problems reported by CCE Node Problem Detector (**CCE Node Problem Detector**). For details about these metrics, see **Table 10-47**. To use related alarm rules, ensure that CCE Node Problem Detector has been installed and is running normally.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Alarm Center**. Then, choose **Alarm Rules** > **Custom Alarm Rules**, and click **Create Alarm Rule**.

Step 3 Configure the alarm rule parameters.

- **Rule Type**: Select **Metric alarm**.
- **Alarm Template**: If you select **No template**, you need to configure the parameters in **Rule Details**. You can also set this parameter to **Use template** to quickly define a PromQL-based alarm rule or modify an existing template.
- **Rule Details**: Configure the parameters listed in the following table.

Parameter	Description	Example Value
Rule Name	Enter the name of the alarm rule.	CoreDNS memory usage higher than 80%
(Optional) Description	Describe the alarm rule.	Check whether the memory usage of CoreDNS is higher than 80%.
Alarm Rule (PromQL)	Enter a Prometheus query statement. For details about how to compile Prometheus query statements, see **Query Examples**.	The following is an example statement for generating an alarm when the maximum memory usage of CoreDNS is higher than 80%: `(sum(container_memory_working_set_bytes{image!="", container!="POD",namespace="kube-system",container="coredns"}) BY (cluster_name, node,container, pod , namespace, cluster) / sum(container_spec_memory_limit_bytes{namespace="kube-system", container="coredns"} > 0) BY (cluster_name, node, container, pod , namespace, cluster) * 100) > 80`
Severity	Select **Critical**, **Major**, **Minor**, or **Warning**.	Critical
Duration	Select an alarm duration from the drop-down list. The default value is **1 minute**.	1 minute

Parameter	Description	Example Value
Alarm Content	Define the content in the alarm notification. Variables in Prometheus can be obtained in the form of ${variable}.	Example: Cluster: ${cluster_name}, Namespace: ${namespace}, Pod: ${pod}, Container: ${container} memory usage is higher than 80%. The current value is ${value} %.
Contact Group	Select an existing contact group. You can also click **Create Contact Group** to create one. For details about the parameters, see **Binding Contact Groups**.	CCEGroup

In the preceding example, an alarm rule named **CoreDNS memory usage higher than 80%** is set for CoreDNS in the **kube-system** namespace, and its severity is **Critical**. When the maximum memory usage is higher than 80% for 1 minute, a notification is sent to all alarm contacts in the **CCEGroup** contact group by SMS message or email. The notification contains the cluster name, namespace, pod name, container name, and current memory usage.

- (Optional) Advanced Settings
  - **Alarm Tag**: An attribute for identifying and grouping alarms to reduce noise. In the message template, the tag value is referenced as *$event.metadate.* A maximum of 10 alarm tags can be added.
  - **Alarm Annotation**: An attribute that is not used for alarm identification. In the message template, the annotation value is referenced as *$event.annotations.* A maximum of 10 alarm annotations can be added.

Step 4 Click **OK**. Then, go to the **Custom Alarm Rules** page to check whether the rule is successfully created.

----End

Adding Event Alarms

NOTE

- To create event-triggered alarm rules, you need to enable Logging and Kubernetes event collection. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.
- Some metric templates are created based on the problems reported by CCE Node Problem Detector (**CCE Node Problem Detector**). For details about these metrics, see **Table 10-47**. To use related alarm rules, ensure that CCE Node Problem Detector has been installed and is running normally.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Alarm Center**. Then, choose **Alarm Rules** > **Custom Alarm Rules**, and click **Create Alarm Rule**.

Step 3 Configure the alarm rule parameters.

- Rule Type: Select **Event alarm**. Common events include Kubernetes events and cloud service events.
- **Rule Details**: Configure the parameters listed in the following table.

Parameter	Description	Example Value
Rule Name	Enter the name of the alarm rule.	ReplicaSet quantity change
(Optional) Description	Describe the alarm rule.	The number of ReplicaSets changes more than three times within 5 minutes.
Event Name	Enter the event name based on the actual Kubernetes event or cloud service event. For details about event names, see **CCE Events**.	ScalingReplicaSet
Triggering Mode	– **Immediate trigger**: An alarm is generated as long as the event occurs. – **Accumulative trigger**: An alarm is generated only after the event is triggered for a preset number of times within the triggering period.	Select **Accumulative trigger**, and set **Monitoring Interval** to **5 minutes** and **Occurrences** to **> 3**.
Severity	Select **Critical**, **Major**, **Minor**, or **Warning**.	Minor
Contact Group	Select an existing contact group. You can also click **Create Contact Group** to create one. For details about the parameters, see **Binding Contact Groups**.	CCEGroup

In the preceding example, an alarm named **ReplicaSet quantity change** is set for the **ScalingReplicaSet** event, and its severity is **Minor**. When the number of ReplicaSet changes more than three times within 5 minutes, a notification is sent to all alarm contacts in the **CCEGroup** by SMS or email.

Step 4 Click **OK**. Then, go to the **Custom Alarm Rules** page to check whether the rule is successfully created.

----End

10.6.4 Configuring Custom Alarms on AOM

CCE interworks with AOM to report alarms and events. By setting alarm rules on AOM, you can check whether resources in clusters are normal in a timely manner.

Process

1. **Creating a Topic on SMN**
2. **Creating an Action Rule**
3. Adding an Alarm Rule
   a. Event alarms: Generate alarms based on the events reported by clusters to AOM. For details about the events and configurations, see **Adding an Event Alarm**.
   b. Metric alarms: Generate alarms based on the thresholds of monitoring metrics, such as resource utilization of servers and components. For details about the metric thresholds and configurations, see **Adding a Metric Alarm**.

Creating a Topic on SMN

Simple Message Notification (SMN) pushes messages to subscribers through emails, SMS messages, and HTTP/HTTPS requests.

A topic is used to publish messages and subscribe to notifications. It serves as a message transmission channel between publishers and subscribers.

You need to create a topic and add a subscription to it. For details, see **Creating a Topic** and **Adding a Subscription to a Topic**.

NOTE

After subscribing to a topic, confirm the subscription in the email or SMS message for the notification to take effect.

Creating an Action Rule

AOM allows you to customize alarm action rules. You can create an alarm action rule to associate an SMN topic with a message template. You can also customize notification content based on a message template.

For details, see **Creating an Alarm Action Rule**. When creating an action rule, select the topic that is created and subscribed to in **Creating a Topic on SMN**.

Adding an Event Alarm

The following uses **NodeNotReady** as an example to describe how to add an event alarm. You can add other alarms by referring to **Table 10-48**.

Table 10-48 Event-based alarms

Event Name	Source	Description	Solution
NodeNotReady	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node.
Rebooted	CCE	An alarm is triggered immediately when a node is restarted.	Log in to the cluster to check the status of the node for which the alarm is generated, check whether the node can be started properly, and locate the cause of the restart.
KUBELETIsDown	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node. Then, restart kubelet.
DOCKERIsDown	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node. Then, restart Docker.
KUBEPROXYIsDown	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node.
KernelOops	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node.
ConntrackFull	CCE	An alarm is triggered immediately when a node is abnormal.	Log in to the cluster and check the status of the node for which the alarm is generated. Set the node as unschedulable and schedule the service pods to another node.
NodePoolSoldOut	CCE	An alarm is triggered immediately when node pool resources are sold out.	Set auto node pool switchover or change the node pool specifications.

Event Name	Source	Description	Solution
NodeCreateFailed	CCE	An alarm is triggered immediately upon a node creation failure.	Rectify the failure and create the node again.
ScaleUpTimedOut	CCE	An alarm is triggered immediately upon node scale-out timeout.	Rectify the failure and try scale-out again.
ScaleDownFailed	CCE	An alarm is triggered immediately upon node scale-in timeout.	Rectify the failure and try scale-in again.
BackOffPullImage	CCE	Image pull retry failed.	Log in to the cluster, locate the failure cause, and deploy the service workload again.

Step 1 Log in to the AOM 2.0 console.

Step 2 In the navigation pane, choose **Alarm Management** > **Alarm Rules**. Then, click **Create Alarm Rule**.

Step 3 Enter basic information as prompted and configure other parameters as follows:

For details about parameters, see **Creating an Event Alarm Rule**.

- **Rule Type**: Select **Event alarm rule**.
- **Event Type**: Select **System**.
- **Event Source**: Select **CCE**.
- **Monitored Object**: Filter monitored objects by notification type, event name, alarm severity, custom attribute, namespace, and cluster name.

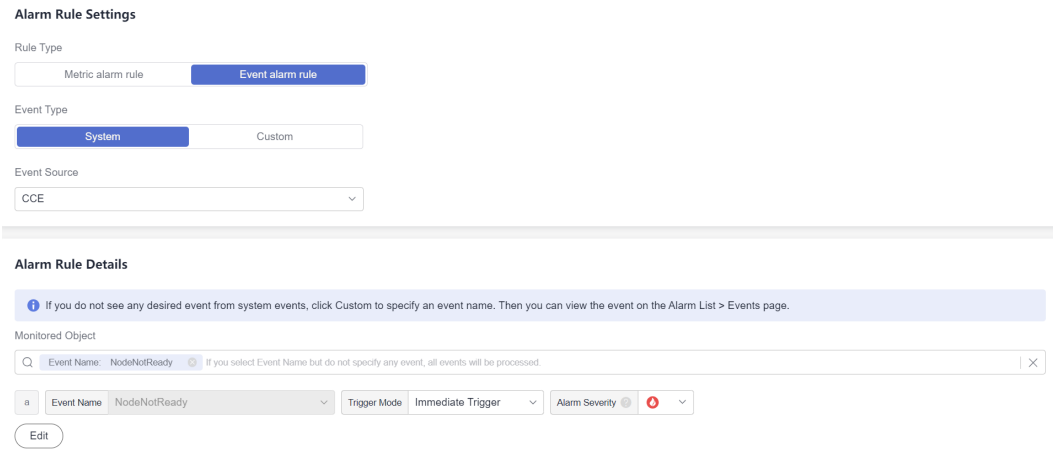
  In this example, filter monitored objects by event name, select **NodeNotReady**, and set **Trigger Mode** to **Immediate Trigger**.
- **Alarm Mode**: Select **Direct alarm reporting**.
- **Action Rule**: Select the action rule created in **Creating an Action Rule**.

Configure other parameters as required.

In this example, the alarm settings are as follows:

If a node in the cluster becomes abnormal, CCE reports the **NodeNotReady** event to AOM. AOM immediately notifies you through SMN based on the action rule.

Figure 10-82 Adding an event alarm



Step 4 Click **Confirm**.

A successfully created alarm rule will be displayed in the rule list.

----End

Adding a Metric Alarm

The following uses a PromQL statement as an example to describe how to add a metric alarm.

Step 1 Log in to the AOM 2.0 console.

Step 2 In the navigation pane, choose **Alarm Management** > **Alarm Rules**. Then, click **Create Alarm Rule**.

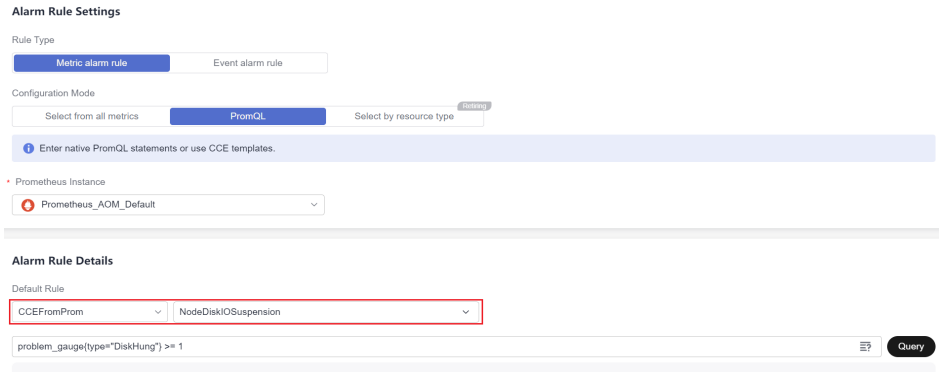
Step 3 Configure parameters as follows:

For details about parameters, see **Creating a Metric Alarm Rule**.

- **Rule Type**: Select **Metric alarm rule**.
- **Configuration Mode**: Select **PromQL**. You can enter native PromQL statements or use CCE templates.
- **Prometheus Instance**: Select the AOM instance whose metrics are reported by Cloud Native Cluster Monitoring in the cluster.
- **Default Rule**:
  - **Custom**: Enter a PromQL statement to configure the alarm rule. For example:
    kube_persistentvolume_status_phase{phase=~"Failed|Pending",cluster="*${cluster_id}*"} > 0

    ${cluster_id} indicates the cluster name. If a PV in the cluster is in the **Failed** or **Pending** state, an alarm will be generated.
  - **CCEFromProm**: Select an alarm template provided by CCE.

Figure 10-83 Adding a metric alarm



- **Alarm Mode**: Select **Direct alarm reporting**.
- **Action Rule**: Select the action rule created in **Creating an Action Rule**.

Configure other parameters as required.

Step 4 Click **Confirm**.

A successfully created alarm rule will be displayed in the rule list.

----End

10.6.5 CCE Events

CCE can report a range of events in a running cluster to AOM. You can add event alarms as required to monitor the health of cluster data plane and control plane components. This helps you quickly identify and resolve problems, ensuring cluster stability and reliability.

- **Data Plane Events**: user operation events, including workload, network, node, storage, and auto scaling events.
  - **Workload events**
  - **Network events**
  - **Node events**
  - **Storage events**
  - **Auto scaling events**
- **Control Plane Events**: master node events, which are usually caused by faults or upgrades of control plane components.

Data Plane Events

Table 10-49 Workload events

Category	Event Name	Severity	Description
Pod	PodOOMKilling	Major	Check whether the pod exits due to OOM. This event is reported by CCE Node Problem Detector (1.18.41 or later) and Cloud Native Log Collection (1.3.2 or later).
Pod	FailedStart	Major	Check whether the pod is started.
Pod	FailedPullImage	Major	Check whether the pod has pulled an image.
Pod	BackOffStart	Major	Check whether the pod fails to be restarted.
Pod	FailedScheduling	Major	Check whether the pod is scheduled.
Pod	BackOffPullImage	Major	Check whether the pod has pulled an image after a retry.
Pod	FailedCreate	Major	Check whether the pod is created.
Pod	Unhealthy	Minor	Check whether the pod health check is successful.
Pod	FailedDelete	Minor	Check whether the workload is deleted.
Pod	ErrImageNeverPull	Minor	Check whether the workload has pulled an image.
Pod	FailedScaleOut	Minor	Check whether workload copies are scaled out.
Pod	FailedStandBy	Minor	Check whether the pod enters the standby state.
Pod	FailedReconfig	Minor	Check whether the pod configuration is updated.
Pod	FailedActive	Minor	Check whether the pod is activated.
Pod	FailedRollback	Minor	Check whether the pod is rolled back.
Pod	FailedUpdate	Minor	Check whether the pod is updated.
Pod	FailedScaleIn	Minor	Check whether a pod scale-in failed.
Pod	FailedRestart	Minor	Check whether the pod is restarted.

Category	Event Name	Severity	Description
Deployment	SelectorOverlap	Minor	Check whether label selectors in the cluster conflict.
Deployment	ReplicaSetCreateError	Minor	Check whether a workload ReplicaSet can be created.
Deployment	DeploymentRollbackRevisionNotFound	Minor	Check whether the Deployment rollback version is available.
DaemonSet	SelectingAll	Minor	Check whether the workload label selector is correctly configured.
Job	TooManyActivePods	Minor	Check whether there are still active pods after the number of pods in a job reaches the preset value.
Job	TooManySucceededPods	Minor	Check whether there are extra running pods after the number of pods in a job reaches the preset value.
CronJob	FailedGet	Minor	Check whether CronJobs can be obtained.
CronJob	FailedList	Minor	Check whether pods can be obtained.
CronJob	UnexpectedJob	Minor	Check whether there are any unknown CronJobs.

Table 10-50 Network events

Category	Event Name	Severity	Description
Service	CreatingLoadBalancerFailed	Minor	Check whether the load balancer is created.
Service	DeletingLoadBalancerFailed	Minor	Check whether the load balancer is deleted.
Service	UpdateLoadBalancerFailed	Minor	Check whether the load balancer is updated.

Table 10-51 Node events

Category	Event Name	Severity	Description
Node	Rebooted	Major	Check whether the node is restarted.
Node	NodeNotSchedulable	Major	Check whether the node is schedulable.
Node	NodeNotReady	Major	Check whether the node is running normally.
Node	NodeCreateFailed	Major	Check whether the node is created.
Node	KUBELETIsDown	Minor	Check whether kubelet is running normally on the node.
Node	NodeHasInsufficientMemory	Minor	Check whether the available memory of the node is sufficient.
Node	UnregisterNetDevice	Minor	Check whether the node is associated with any unregistered network device.
Node	NetworkCardNotFound	Minor	Check the node ENI status.
Node	KUBEPROXYIsDown	Minor	Check whether kube-proxy is running normally on the node.
Node	NodeOutOfDisk	Minor	Check whether the node disk space is sufficient.
Node	TaskHung	Minor	Check whether there are any suspended tasks on the node.
Node	CIDRNotAvailable	Minor	Check whether the node CIDR block is available.
Node	ConntrackFull	Minor	Check whether the connection tracking table on the node is full.
Node	NodeHasDiskPressure	Minor	Check whether the node disk space is sufficient.
Node	NodeInstallFailed	Minor	Check whether nodes are managed in the cluster.
Node	KernelOops	Minor	Check whether the OS kernel of the node is faulty.

Category	Event Name	Severity	Description
Node	OOMKilling	Minor	• The memory used by pods on the node exceeds the limit. As a result, the process is terminated. • The memory used by pods on the node does not exceed the limit, but the available memory of the node is insufficient. As a result, OOM occurs.
Node	DOCKERIsDown	Minor	Check whether the container engine of the node is running normally.
Node	CIDRAssignmentFailed	Minor	Check whether a CIDR block is allocated for the node.
Node	DockerHung	Minor	Check whether the Docker process on the node is suspended.
Node	FilesystemIsReadOnly	Minor	Check whether the file system of the node is read-only.
Node	NTPIsDown	Minor	Check whether NTP is running normally on the node.
Node	NodeUninstallFailed	Minor	Check whether the node is uninstalled.
Node	AUFSUmountHung	Minor	Check whether detaching the node disk is suspended.
Node	CNIIsDown	Minor	Check whether the CNI add-on on the node is faulty.
Namespace	DeleteNodeWithNoServer	Minor	Check whether discarded nodes are cleared.

Table 10-52 Storage events

Category	Event Name	Severity	Description
PV	DetachVolumeFailed	Minor	Check whether the block storage is detached.
PV	VolumeUnknownReclaimPolicy	Minor	Check whether a volume reclamation policy is specified.
PV	SetUpAtVolumeFailed	Minor	Check whether the data volume is mounted.

Category	Event Name	Severity	Description
PV	VolumeFailedRecycle	Minor	Check whether the data volume is reclaimed.
PV	WaitForAttachVolumeFailed	Minor	Check whether block storage is attached to the node.
PV	VolumeFailedDelete	Minor	Check whether the data volume is deleted.
PV	MountDeviceFailed	Minor	Check whether the data volume is mounted.
PV	TearDownAtVolumeFailed	Minor	Check whether the data volume is detached.
PV	UnmountDeviceFailed	Minor	Check whether the drive letter of the data volume is unmounted.
PV	AttachVolumeFailed	Minor	Check whether block storage is detached from the node.
PVC	VolumeResizeFailed	Minor	Check whether the capacity of the data volume is expanded.
PVC	ClaimLost	Minor	Check whether the PVC volume is normal.
PVC	ProvisioningFailed	Minor	Check whether the data volume is created.
PVC	ProvisioningCleanupFailed	Minor	Check whether the data volume is cleared.
PVC	ClaimMisbound	Minor	Check whether the PVC is bound to an incorrect volume.

Table 10-53 Auto scaling events

Category	Event Name	Severity	Description
Autoscaler	ScaleUpTimedOut	Major	Check whether adding nodes to the node pool timed out.
Autoscaler	NodePoolAvailable	Major	Check whether the node pool resources are sufficient.
Autoscaler	ScaleDown	Major	Nodes are being deleted from the cluster.
Autoscaler	NotTriggerScale-Up	Major	Check whether a node scale-out is triggered.

Category	Event Name	Severity	Description
Autoscaler	DeleteUnregistered	Major	Check whether unregistered nodes are deleted.
Autoscaler	ScaleDownEmpty	Major	Check whether idle nodes are scaled in.
Autoscaler	ScaleDownFailed	Major	Check whether nodes are scaled in.
Autoscaler	FailedToScaleUpGroup	Major	Check whether an error occurred during a node pool scale-out.
Autoscaler	ScaledUpGroup	Major	Check whether the node pool is scaled out.
Autoscaler	ScaleUpFailed	Major	Check whether the node is scaled out.
Autoscaler	FixNodeGroupSizeDone	Major	Check whether the number of nodes in the node pool is restored.
Autoscaler	NodeGroupInBackOff	Major	Check whether there are any rollback retries during node pool scaling.
Autoscaler	FixNodeGroupSizeError	Major	Check whether the number of nodes in the node pool is restored.
Autoscaler	NodePoolSoldOut	Major	Check whether the node pool resources are sufficient.
Autoscaler	TriggeredScaleUp	Major	Check whether a node scale-out is triggered.
Autoscaler	StartScaledUpGroup	Major	Check whether a node pool scaled-out is started.
Autoscaler	DeleteUnregisteredFailed	Major	Check whether unregistered nodes are deleted.
HPA	InvalidTargetRange	Major	• Invalid **extendedhpa.metrics** is configured in **annotations** of HPA. • The metric type in **spec** of HPA is incorrect.
HPA	FailedGetScale	Major	HPA failed to obtain the resource object to be scaled.

Category	Event Name	Severity	Description
HPA	FailedComputeMetricsReplicas	Major	An error occurs when the number of copies to be adjusted for resources is calculated. For example, metric-server is unavailable, resource metric collection fails, or the CPU usage is incorrectly set. You can run the following command to view details: `kubectl describe horizontalpodautoscaler <hpa-name>`
HPA	FailedGetObject-Metric	Major	Failed to obtain the metrics of the specified object (such as PVC and ConfigMaps).
HPA	FailedGetPodsMetric	Major	Failed to obtain the pod resource metric (resource usage of a pod).
HPA	FailedGetResour-ceMetric	Major	Failed to obtain the cluster resource metric (resource usage of a cluster).
HPA	FailedGetContai-nerResourceMetric	Major	Failed to obtain the resource metrics of a container.
HPA	FailedGetExternalMetric	Major	Failed to obtain external metrics.
HPA	FailedRescale	Major	Failed to update the desired number of copies of the resource object to be scaled.
HPA	SuccessfulRescale	Minor	The desired number of copies of the resource object to be scaled is updated.
CronHPA	ScaleFailed	Major	CronHPA failed to update the desired number of copies of the resource object to be scaled.
CronHPA	FailedGetHorizontalPodAutoscaler	Major	CronHPA failed to query the associated HPA object. (Generally, kube-apiserver cannot respond.)
CronHPA	FailedGetHpaScale	Major	CronHPA failed to obtain the resource object to be scaled.
CronHPA	UpdateHPAFailed	Major	CronHPA failed to update the associated HPA object.
CronHPA	UpdateHPASuccess	Minor	CronHPA successfully updates the associated HPA object.

Category	Event Name	Severity	Description
CronHPA	SkipUpdateHPA	Minor	CronHPA skips updating the associated HPA object.
CronHPA	SkipUpdateTarget	Minor	CronHPA skips updating the number of copies of the resource object to be scaled.
CronHPA	UpdateTargetSuccess	Minor	CronHPA successfully updates the number of copies of the resource object to be scaled.
CustomedHPA	FailedSetPolicy-Settings	Major	Failed to parse the cooldown period of CustomedHPA.
CustomedHPA	FailedSubmitRule	Major	CustomedHPA failed to process schedule rules or metric rules.
CustomedHPA	FailedComputeReplicas	Major	CustomedHPA failed to trigger resource scaling based on the compute metrics.
CustomedHPA	FailedScale	Major	CustomedHPA failed to update the desired number of copies of the resource object to be scaled. (Generally, kube-apiserver cannot respond).
CustomedHPA	MetricScaleSuccess	Minor	CustomedHPA triggers resource scaling based on the metric rule.
CustomedHPA	CronScaleSuccess	Minor	CustomedHPA triggers resource scaling based on the periodic rule.

Control Plane Events

Table 10-54 Control plane events

Event ID	Severity	Description
Internal error	Major	Check whether an internal error occurs in the cluster.
External dependency error	Major	Check whether an error occurs in cluster external dependencies.
Failed to initialize process thread	Major	Check whether a cluster initialization thread is executed.
Failed to update database	Major	Check whether the database for the cluster is updated.

Event ID	Severity	Description
Failed to create node by nodepool	Major	Check whether nodes are created in the node pool.
Failed to delete node by nodepool	Major	Check whether nodes are deleted from the node pool.
Failed to create yearly/ monthly subscription node	Major	Check whether the yearly/monthly node is created in the cluster.
Failed to cancel the authorization of accessing the image of the master.	Major	When creating a cluster, check whether the authorization for the resource tenant to access the master node image is canceled.
Failed to create the virtual IP for the master	Major	When creating a cluster, check whether the virtual IP address is allocated.
Failed to delete the node VM	Major	Check whether the node (VM) is deleted from the cluster.
Failed to delete the security group of node	Major	Check whether the security group of the node is deleted from the cluster.
Failed to delete the security group of master	Major	Check whether the security group of the master node is deleted from the cluster.
Failed to delete the security group of port	Major	Check whether the ENI security group of the master node is deleted from the cluster.
Failed to delete the security group of eni or subeni	Major	Check whether ENI or sub-ENI security group is deleted from the cluster.
Failed to detach the port of master	Major	Check whether the ENI of the master node is detached from the cluster.
Failed to delete the port of master	Major	Check whether the ENI of the master node is deleted from the cluster.
Failed to delete the master VM	Major	Check whether the master node (VM) is deleted from the cluster.
Failed to delete the key pair of master	Major	Check whether the key pair of the master node is deleted from the cluster.
Failed to delete the subnet of master	Major	Check whether the subnet of the master node is deleted from the cluster.
Failed to delete the VPC of master	Major	Check whether the VPC of the master node is deleted from the cluster.
Failed to delete certificate of cluster	Major	Check whether the certificate is deleted from the cluster.

Event ID	Severity	Description
Failed to delete the server group of master	Major	Check whether the master node (ECS) is deleted from the cluster.
Failed to delete the virtual IP for the master	Major	Check whether the virtual IP address is deleted from the cluster.
Failed to get floating IP of the master	Major	Check whether the floating IP address of the master node is obtained.
Failed to get cluster flavor	Major	Check whether the cluster flavor is obtained.
Failed to get cluster endpoint	Major	Check whether the cluster endpoint is obtained.
Failed to get kubernetes connection	Major	Check whether the Kubernetes cluster connections are obtained.
Failed to update secret	Major	Check whether the cluster Secret is updated.
Operation timed out	Major	Check whether the user operation timed out.
Connecting to Kubernetes cluster timed out	Major	Check whether accessing the Kubernetes cluster timed out.
Failed to check component status or components are abnormal	Major	Check whether the statuses of cluster components can be obtained or whether the components malfunction.
The node is not found in kubernetes cluster	Major	Check whether the node can be found in the Kubernetes cluster.
The status of node is not ready in kubernetes cluster	Major	Check whether the node is running normally in the Kubernetes cluster.
Can't find corresponding vm of this node in ECS	Major	Check whether the node can be found on the ECS console.
Failed to upgrade the master	Major	Check whether the master node has been upgraded.
Failed to upgrade the node	Major	Check whether the node has been upgraded.
Failed to change flavor of the master	Major	Check whether the master node flavor has been changed.
Change flavor of the master timeout	Major	Check whether changing the master node flavor timed out.
Failed to pass verification while creating yearly/ monthly subscription node	Major	Check whether creating a yearly/ monthly node has been verified.

Event ID	Severity	Description
Failed to install the node	Major	Check whether the node is installed in the cluster.
Failed to clean routes of cluster container network in VPC	Major	Check whether the routes of cluster container VPCs are cleaned.
Cluster status is Unavailable	Major	Check whether the cluster is available.
Cluster status is Error	Major	Check whether the cluster is faulty.
Cluster status is not updated for a long time	Major	Check whether the cluster retains in a state for a long time.
Failed to update master status after upgrading cluster timeout	Major	Check whether the status of the master node is updated after the cluster upgrade timed out.
Failed to update running jobs after upgrading cluster timeout	Major	Check whether running tasks are updated after the cluster upgrade timed out.
Failed to update cluster status	Major	Check whether the cluster status is updated.
Failed to update node status	Major	Check whether the node status is updated.
Failed to remove the static node from database	Major	Check whether nodes are removed from the database after managing nodes timed out.
Failed to update node status to abnormal after node processing timeout	Major	Check whether the node status is updated to abnormal after processing the node timed out.
Failed to update the cluster endpoint	Major	Check whether the cluster endpoint is updated.
Failed to delete the unavailable connection of the Kubernetes cluster	Major	Check whether unavailable Kubernetes connections are deleted.
Failed to sync the cluster cert	Major	Check whether the cluster certificate is synchronized.

10.7 Log Auditing

10.7.1 CCE Operations Supported by Cloud Trace Service

Cloud Trace Service (CTS) records operations on cloud service resources, allowing users to query, audit, and backtrack the resource operation requests initiated from the management console or open APIs as well as responses to the requests.

Table 10-55 CCE Operations Supported by CTS

Operation	Resource Type	Event Name
Creating an agency	Cluster	createUserAgencies
Creating a cluster	Cluster	createCluster
Creating a yearly/monthly-billed cluster	Cluster	createCluster/createPeriodicCluster
Updating the description of a cluster	Cluster	updateCluster
Upgrading a cluster	Cluster	clusterUpgrade
Deleting a cluster	Cluster	claimCluster/deleteCluster
Downloading a cluster certificate	Cluster	getClusterCertByUID
Binding and unbinding an EIP	Cluster	operateMasterEIP
Waking up a cluster and resetting node management (V2)	Cluster	operateCluster
Hibernating a cluster (V3)	Cluster	hibernateCluster
Waking up a cluster (V3)	Cluster	awakeCluster
Changing the specifications of a pay-per-use cluster	Cluster	resizeCluster
Changing the specifications of a yearly/monthly-billed cluster	Cluster	resizePeriodCluster
Modifying configurations of a cluster	Cluster	updateConfiguration
Creating a node pool	Node pool	createNodePool

Operation	Resource Type	Event Name
Updating a node pool	Node pool	updateNodePool
Deleting a node pool	Node pool	claimNodePool
Migrating a node pool	Node pool	migrateNodepool
Modifying node pool configurations	Node pool	updateConfiguration
Creating a node	Node	createNode
Creating a yearly/ monthly-billed node	Node	createPeriodNode
Deleting all the nodes from a specified cluster	Node	deleteAllHosts
Deleting a single node	Node	deleteOneHost/claimOneHost
Updating the description of a node	Node	updateNode
Creating an add-on instance	Add-on instance	createAddonInstance
Deleting an add-on instance	Add-on instance	deleteAddonInstance
Uploading a chart	Chart	uploadChart
Updating a chart	Chart	updateChart
Deleting a chart	Chart	deleteChart
Creating a release	Release	createRelease
Upgrading a release	Release	updateRelease
Deleting a release	Release	deleteRelease
Creating a ConfigMap	Kubernetes resource	createConfigmaps
Creating a DaemonSet	Kubernetes resource	createDaemonsets
Creating a Deployment	Kubernetes resource	createDeployments
Creating an event	Kubernetes resource	createEvents

Operation	Resource Type	Event Name
Creating an Ingress	Kubernetes resource	createIngresses
Creating a job	Kubernetes resource	createJobs
Creating a namespace	Kubernetes resource	createNamespaces
Creating a node	Kubernetes resource	createNodes
Creating a PersistentVolume-Claim	Kubernetes resource	createPersistentvolumeclaims
Creating a pod	Kubernetes resource	createPods
Creating a replica set	Kubernetes resource	createReplicasets
Creating a resource quota	Kubernetes resource	createResourcequotas
Creating a secret	Kubernetes resource	createSecrets
Creating a service	Kubernetes resource	createServices
Creating a StatefulSet	Kubernetes resource	createStatefulsets
Creating a volume	Kubernetes resource	createVolumes
Deleting a ConfigMap	Kubernetes resource	deleteConfigmaps
Deleting a DaemonSet	Kubernetes resource	deleteDaemonsets
Deleting a Deployment	Kubernetes resource	deleteDeployments
Deleting an event	Kubernetes resource	deleteEvents
Deleting an Ingress	Kubernetes resource	deleteIngresses
Deleting a job	Kubernetes resource	deleteJobs

Operation	Resource Type	Event Name
Deleting a namespace	Kubernetes resource	deleteNamespaces
Deleting a node	Kubernetes resource	deleteNodes
Deleting a Pod	Kubernetes resource	deletePods
Deleting a replica set	Kubernetes resource	deleteReplicasets
Deleting a resource quota	Kubernetes resource	deleteResourcequotas
Deleting a secret	Kubernetes resource	deleteSecrets
Deleting a service	Kubernetes resource	deleteServices
Deleting a StatefulSet	Kubernetes resource	deleteStatefulsets
Deleting volumes	Kubernetes resource	deleteVolumes
Replacing a specified ConfigMap	Kubernetes resource	updateConfigmaps
Replacing a specified DaemonSet	Kubernetes resource	updateDaemonsets
Replacing a specified Deployment	Kubernetes resource	updateDeployments
Replacing a specified event	Kubernetes resource	updateEvents
Replacing a specified ingress	Kubernetes resource	updateIngresses
Replacing a specified job	Kubernetes resource	updateJobs
Replacing a specified namespace	Kubernetes resource	updateNamespaces
Replacing a specified node	Kubernetes resource	updateNodes
Replacing a specified PersistentVolume-Claim	Kubernetes resource	updatePersistentvolumeclaims

Operation	Resource Type	Event Name
Replacing a specified pod	Kubernetes resource	updatePods
Replacing a specified replica set	Kubernetes resource	updateReplicasets
Replacing a specified resource quota	Kubernetes resource	updateResourcequotas
Replacing a specified secret	Kubernetes resource	updateSecrets
Replacing a specified service	Kubernetes resource	updateServices
Replacing a specified StatefulSet	Kubernetes resource	updateStatefulsets
Replacing the specified status	Kubernetes resource	updateStatus
Uploading a chart	Kubernetes resource	uploadChart
Updating a component template	Kubernetes resource	updateChart
Deleting a chart	Kubernetes resource	deleteChart
Creating a template application	Kubernetes resource	createRelease
Updating a template application	Kubernetes resource	updateRelease
Deleting a template application	Kubernetes resource	deleteRelease

10.7.2 Viewing CTS Traces in the Trace List

Scenarios

After you enable CTS and the management tracker is created, CTS starts recording operations on cloud resources. After a data tracker is created, the system starts recording operations on data in Object Storage Service (OBS) buckets. Cloud Trace Service (CTS) stores operation records (traces) generated in the last seven days.

This section describes how to query or export operation records of the last seven days on the CTS console.

- **Viewing Real-Time Traces in the Trace List of the New Edition**

- **Viewing Real-Time Traces in the Trace List of the Old Edition**

Constraints

- Traces of a single account can be viewed on the CTS console. Multi-account traces can be viewed only on the **Trace List** page of each account, or in the OBS bucket or the **CTS/system** log stream configured for the management tracker with the organization function enabled.
- You can only query operation records of the last seven days on the CTS console. To store operation records for longer than seven days, you must configure transfer to OBS or Log Tank Service (LTS) so that you can view them in OBS buckets or LTS log groups.
- After performing operations on the cloud, you can query management traces on the CTS console one minute later and query data traces five minutes later.
- These operation records are retained for seven days on the CTS console and are automatically deleted upon expiration. Manual deletion is not supported.

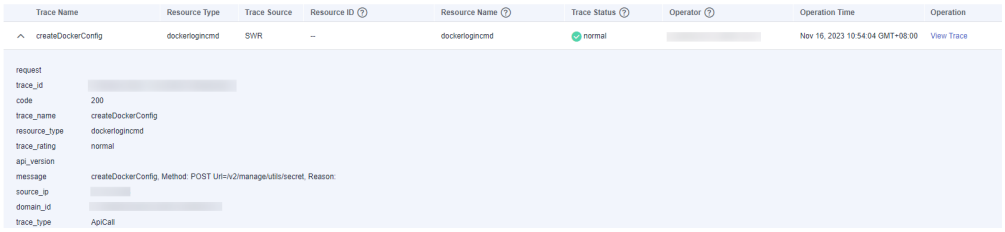
Viewing Real-Time Traces in the Trace List of the New Edition

1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & Governance** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. On the **Trace List** page, use advanced search to query traces. You can combine one or more filters.
   - **Trace Name**: Enter a trace name.
   - **Trace ID**: Enter a trace ID.
   - **Resource Name**: Enter a resource name. If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty.
   - **Resource ID**: Enter a resource ID. Leave this field empty if the resource has no resource ID or if resource creation failed.
   - **Trace Source**: Select a cloud service name from the drop-down list.
   - **Resource Type**: Select a resource type from the drop-down list.
   - **Operator**: Select one or more operators from the drop-down list.
   - **Trace Status**: Select **normal**, **warning**, or **incident**.
     - **normal**: The operation succeeded.
     - **warning**: The operation failed.
     - **incident**: The operation caused a fault that is more serious than the operation failure, for example, causing other faults.
   - **Enterprise Project ID**: Enter an enterprise project ID.
   - **Access Key**: Enter a temporary or permanent access key ID.
   - Time range: Select **Last 1 hour**, **Last 1 day**, or **Last 1 week**, or specify a custom time range within the last seven days.

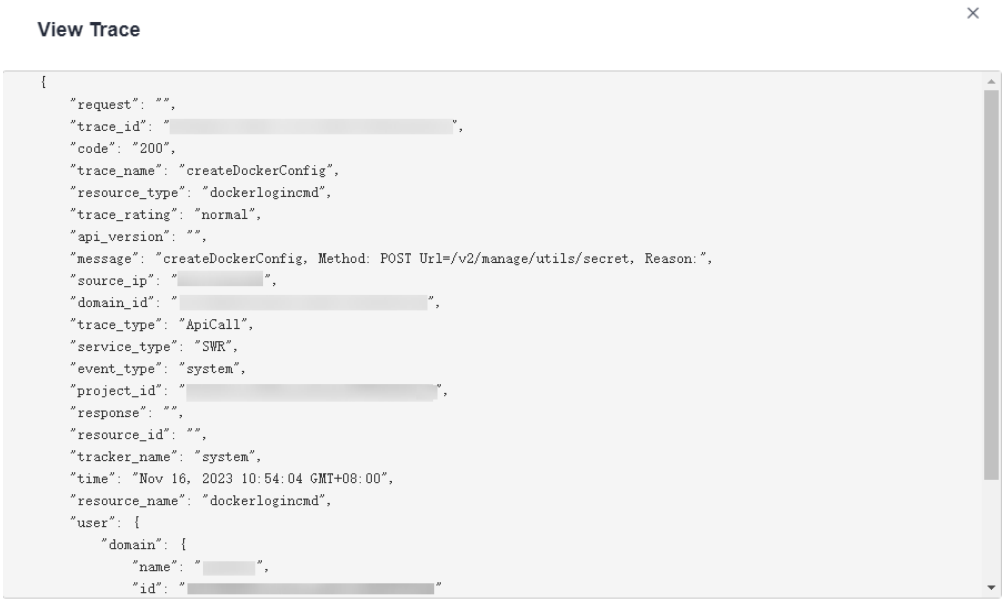
5. On the **Trace List** page, you can also export and refresh the trace list, and customize columns to display.
   - Enter any keyword in the search box and press **Enter** to filter desired traces.
   - Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5,000 records.
   - Click to view the latest information about traces.
   - Click to customize the information to be displayed in the trace list. If **Auto wrapping** is enabled ( ), excess text will move down to the next line; otherwise, the text will be truncated. By default, this function is disabled.
6. For details about key fields in the trace structure, see **Trace Structure** and **Example Traces**.
7. (Optional) On the **Trace List** page of the new edition, click **Go to Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

Viewing Real-Time Traces in the Trace List of the Old Edition

1. Log in to the management console.
2. Click in the upper left corner and choose **Management & Governance** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. Each time you log in to the CTS console, the new edition is displayed by default. Click **Go to Old Edition** in the upper right corner to switch to the trace list of the old edition.
5. Set filters to search for your desired traces. The following filters are available.
   - **Trace Type**, **Trace Source**, **Resource Type**, and **Search By**: Select a filter from the drop-down list.
     - If you select **Resource ID** for **Search By**, specify a resource ID.
     - If you select **Trace name** for **Search By**, specify a trace name.
     - If you select **Resource name** for **Search By**, specify a resource name.
   - **Operator**: Select a user.
   - **Trace Status**: Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.
   - Time range: Select **Last 1 hour**, **Last 1 day**, or **Last 1 week**, or specify a custom time range within the last seven days.
6. Click **Query**.
7. On the **Trace List** page, you can also export and refresh the trace list.
   - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5,000 records.
   - Click to view the latest information about traces.
8. Click on the left of a trace to expand its details.



9. Click **View Trace** in the **Operation** column. The trace details are displayed.



10. For details about key fields in the trace structure, see **Trace Structure** and **Example Traces** in the *CTS User Guide*.
11. (Optional) On the **Trace List** page of the old edition, click **New Edition** in the upper right corner to switch to the **Trace List** page of the new edition.

10.8 O&M FAQ

10.8.1 Billing FAQ

Indexes

- **How Are Observability Services Billed?**
- **Why Am I Still Being Billed after Logging is Disabled?**

How Are Observability Services Billed?

Free Scenarios

Monitoring Center is free. All metrics used by Monitoring Center are reported and stored in AOM. In AOM, basic metrics are free and stored for 15 days (cannot be modified currently). For details, see **Basic Metrics**.

Logging is free. All logs generated for a cluster are reported to and stored in LTS. LTS offers a free quota of 500 MB per month for log read/write, indexing, and retention.

Alarm Center is free. Cluster alarm notifications are sent by SMN. SMN offers a free quota of 100 SMS messages per month and 1,000 emails per month. For details about how to calculate the number of SMS messages, see **SMS Length Calculation**.

Billed Scenarios

Monitoring Center: Custom metrics configured for a cluster are billed based on the AOM billing rules. For details, see **AOM Billing Items**.

Logging: When the free quota is used up, you will be billed for excess usage. For details, see **LTS Billing**.

Alarm Center: When the free quota is used up, you will be billed for excess usage. For details, see **SMN Billing**.

Why Am I Still Being Billed after Logging is Disabled?

You will be billed for Logging through LTS. If you disable Logging, you will not be billed for log read/write and indexing. However, log retention will still incur expenditures. For example, if you change the log retention period to one day, you will not be billed for storage until that day has elapsed. For details, see **How Logging Is Billed?**

To change the log retention period, log in to the LTS console, locate the log group (named after the cluster ID) of the cluster, and click **Modify** in the **Operation** column to change the log retention period.

Figure 10-84 Modifying a log group

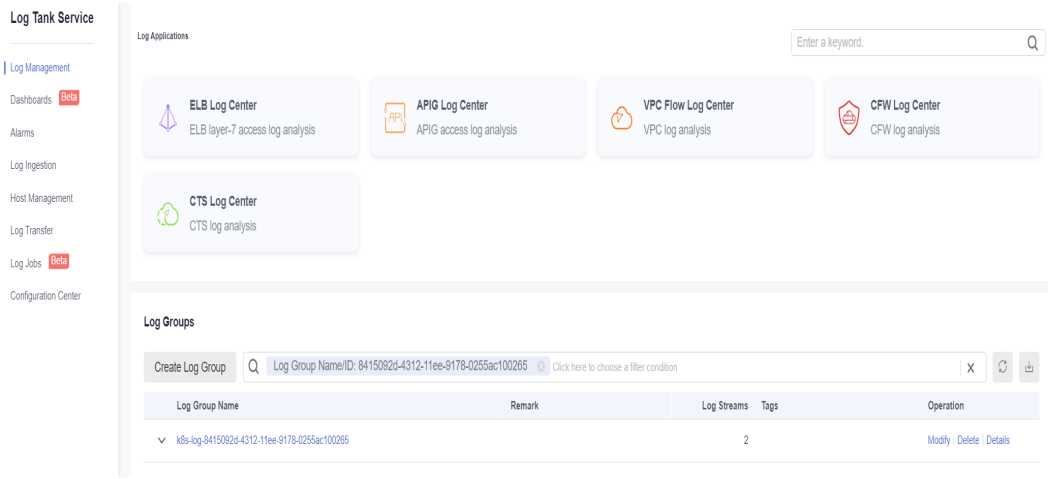
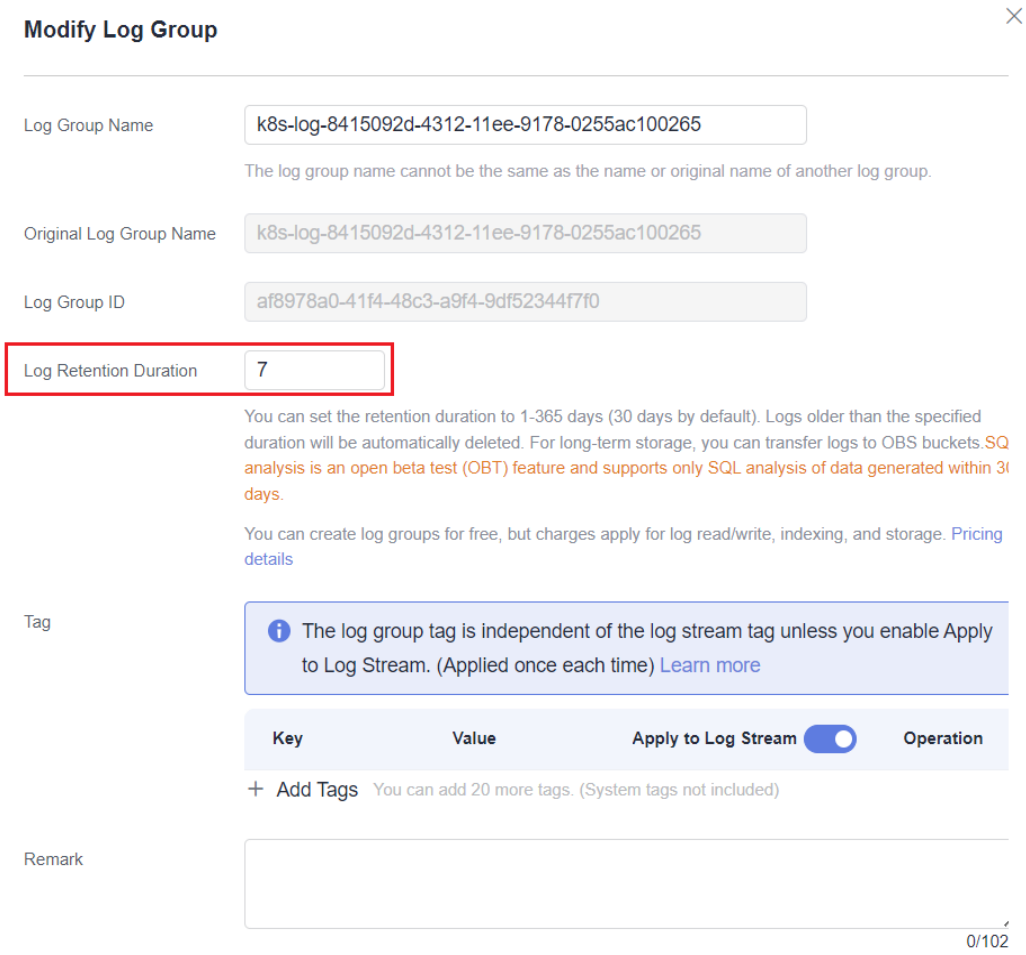


Figure 10-85 Changing the log retention period



10.8.2 Monitoring Center FAQ

Indexes

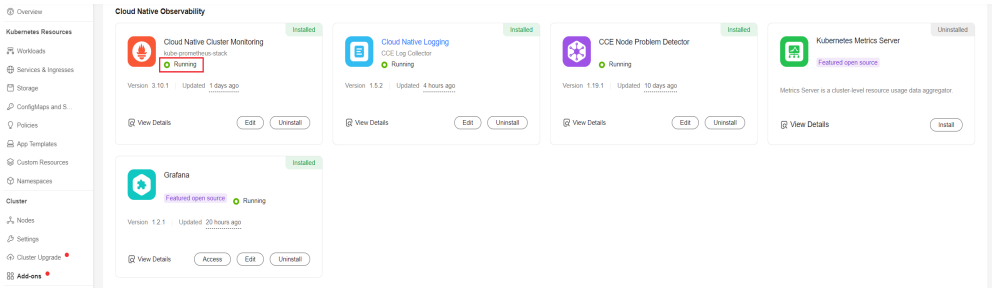
- Why Is There No Data on Monitoring Center?
- How Do I Disable Monitoring Center?
- Why Are Custom Metrics Not Displayed on Monitoring Center?
- Why Is the Resource Information Not Displayed in the Node List for a Short Time (1 to 2 Minutes) After the prometheus-server Instance Is Restarted When the Cloud Native Cluster Monitoring Add-on Is Deployed with Local Data Storage Enabled?
- Why Is Some Data Doubled After the kube-state-metrics Instance Is Restarted When the Cloud Native Cluster Monitoring Add-on Is Deployed with Local Data Storage Enabled?
- Why Does the Cloud Native Cluster Monitoring Add-on with Local Data Storage Enabled Fail to Report Metrics?
- Why Does the Workload/Node CPU Usage of Monitoring Center Exceed 100%?
- Why Is 403 Displayed in Collection Endpoint Access? How Do I Handle It?

Why Is There No Data on Monitoring Center?

- Possible cause 1: The Cloud Native Cluster Monitoring add-on is abnormal.

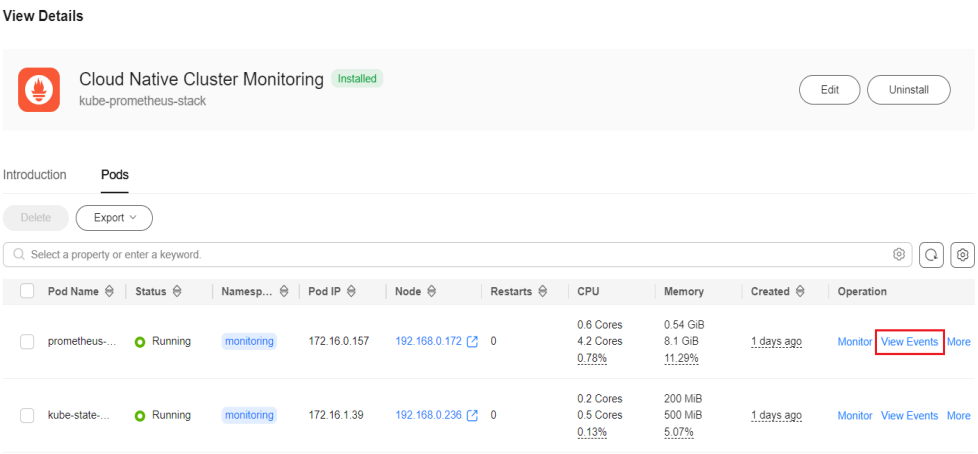
  Access the **Add-ons** page on the cluster console and check whether the Cloud Native Cluster Monitoring add-on is in the **Running** state.

  Figure 10-86 Checking the add-on status

  If the add-on is not running normally, locate the fault based on the events.

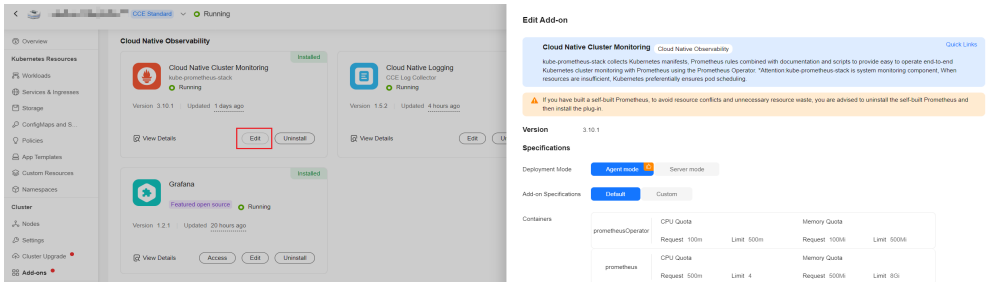
  Figure 10-87 Viewing add-on events

- Possible cause 2: The AOM instance interconnected with the Cloud Native Cluster Monitoring add-on is deleted.

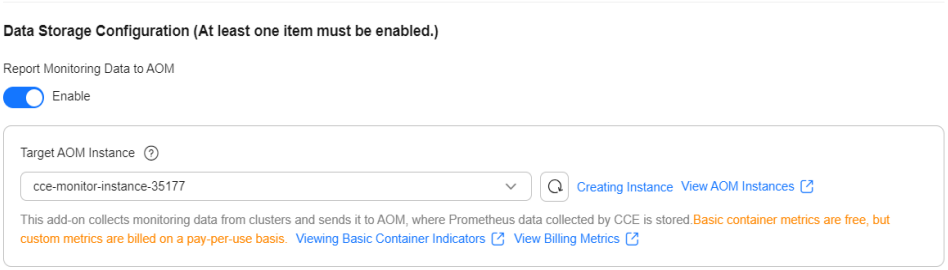
  Access the **Add-ons** page on the cluster console and check the configuration of the Cloud Native Cluster Monitoring add-on.

  Figure 10-88 Editing add-on configuration

  Ensure that **AOM Instance** is not left empty.

Figure 10-89 Viewing the AOM instance



How Do I Disable Monitoring Center?

To disable cluster monitoring, uninstall the Cloud Native Cluster Monitoring add-on on the **Add-ons** page or disable the option for interconnecting with AOM.

Why Are Custom Metrics Not Displayed on Monitoring Center?

Monitoring Center currently does not display custom metrics. To view custom metrics, you can create a dashboard for custom metrics on the dashboard of AOM.

Why Is the Resource Information Not Displayed in the Node List for a Short Time (1 to 2 Minutes) After the prometheus-server Instance Is Restarted When the Cloud Native Cluster Monitoring Add-on Is Deployed with Local Data Storage Enabled?

After a prometheus-server instance is restarted, the UID tag values of its metrics change. During the rolling restart of the prometheus-server instance, metrics overlap because data is stored locally. This means the Cloud Native Cluster Monitoring add-on reports metrics from both the old and new prometheus-server instances to AOM, resulting in inaccurate resource information in the node list. When the metrics overlap, the resource information in the node list is not displayed. Unless otherwise specified, you are advised to connect the Cloud Native Cluster Monitoring add-on to AOM with local data storage disabled.

Why Is Some Data Doubled After the kube-state-metrics Instance Is Restarted When the Cloud Native Cluster Monitoring Add-on Is Deployed with Local Data Storage Enabled?

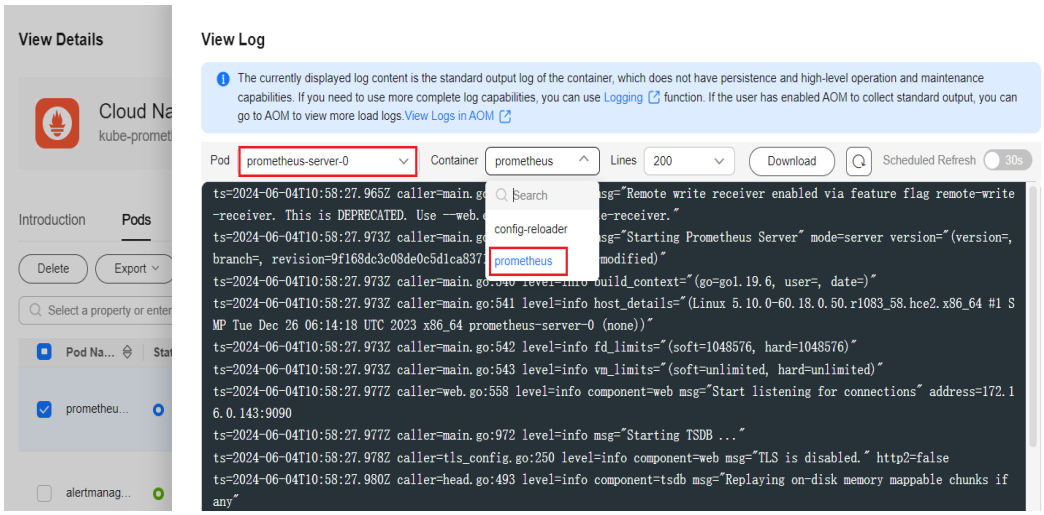
When the kube-state-metrics instance is scheduled to a new node, the instance tag values of the metrics collected by the kube-state-metrics instance change. During the rolling restart of the kube-state-metrics instance, metrics overlap because data is stored locally. This means the Cloud Native Cluster Monitoring add-on reports metrics from both the old and new kube-state-metrics instances. In addition, the instance label values are inconsistent, so all metrics are considered valid. As a result, the number of nodes, the number of workloads, the number of pods, the number of namespaces, and the number of control plane components displayed on the **Clusters** tab (**Monitoring Center** > **Clusters**) are all doubled. Unless otherwise specified, you are advised to connect the Cloud Native Cluster Monitoring add-on to AOM with local data storage disabled.

Why Does the Cloud Native Cluster Monitoring Add-on with Local Data Storage Enabled Fail to Report Metrics?

The add-on pod with local data storage enabled has run out of storage space on the PV. As a result, metrics cannot be written.

Go to the **Add-ons** page, select the **prometheus-server-*x*** instance, and view its logs. If the log contains information similar to "no space left on device", the space of the disk mounted to this add-on pod is insufficient.

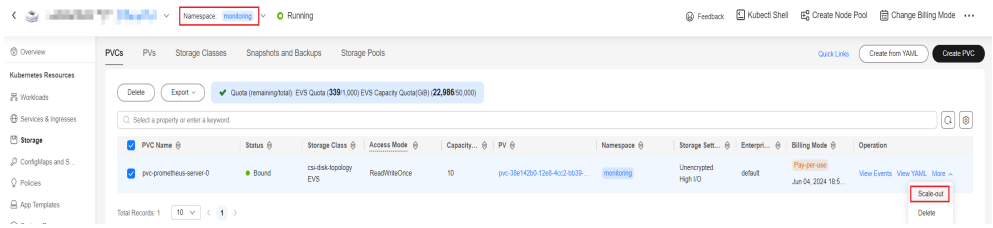
Figure 10-90 Viewing the add-on pod logs



Solutions

- Solution 1: You are advised to connect the add-on with local data storage disabled to the AOM instance. If AOM is used to manage metrics, storage management is not required.
- Solution 2: In the navigation pane, choose **Storage**. On the displayed page, switch to the **monitoring** namespace, select the **pvc-prometheus-server-0** disk, and click **More** > **Scale-out** in the **Operation** column. After the scale-out is complete, go to the **StatefulSets** tab and restart the **prometheus-server-0** instance.

Figure 10-91 Expanding the PVC capacity



NOTE

Insufficient disk space will prevent Prometheus metrics from being written. As a result, data cannot be collected. This means that any monitoring data generated during the scale-out and subsequent restart will be lost.

Why Does the Workload/Node CPU Usage of Monitoring Center Exceed 100%?

The workload CPU usage is calculated using **container_cpu_usage_seconds_total**. The system periodically updates the used CPU and the time point at which the used CPU is collected. By default, Prometheus collects metrics at the collection time point instead of the time point specified by **container_cpu_usage_seconds_total**. As a result, the time point at which the used CPU is collected is inaccurate, and there is a short latency.

Assume that the system updates the used CPU every 6 seconds, and the collection period is 15 seconds, Prometheus collected data at 18:30:14 for the first time and at 18:30:29 for the second time. However, the time point specified by **container_cpu_usage_seconds_total** is 18:30:10 for the first and 18:30:28 for the second time.

Used CPU	Time Point
100,000	18:30:10
150,000	18:30:16
200,000	18:30:22
250,000	18:30:28
300,000	18:30:34

- Actual used CPU per second: (150000-100000)/(18:30:16-18:30:10) = 8333.33
- Used CPU per second collected by Prometheus: (250000-100000)/(18:30:29-18:30:14) = 10000

NOTE

The preceding data values are manually amplified and are only examples. The actual difference is small.

Solution

You can configure **honorTimestamps** to use the time point specified by **container_cpu_usage_seconds_total** to avoid this problem. Weigh the pros and cons before deciding whether to configure **honorTimestamps**.

Configure honorTimestamps	Pros	Cons
No (default behavior of Prometheus)	• The metric compression ratio is higher, the total number of stored metrics decreases, and the query performance is excellent. • The collection intervals of different metrics are the same, which is driven by the Prometheus collection interval. Generally, breakpoints do not occur. • There is a low probability that deviations occur when multiple metrics are combined for PromQL calculation.	Metrics such as the CPU usage may be slightly distorted.
Yes	The time points at which metrics are collected are consistent with the actual time points. In scenarios such as pressure tests, the calculated results are more authentic.	• The total number of stored metrics increases, the metric compression ratio decreases, and the query performance deteriorates. • Not all metrics carry correct timestamps. As a result, a large deviation may occur when PromQL calculation is performed on multiple metrics. • In extreme cases, breakpoints may occur on metrics.

To configure **honorTimestamps**, take the following steps:

NOTE

Cloud Native Cluster Monitoring 3.11.0 or later has been installed in the cluster, and the preset collection function has been enabled.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **ConfigMaps and Secrets**, switch to the **monitoring** namespace, and locate the **persistent-user-config** configuration item.

Step 3 Click **Edit YAML**, search for **kubelet-cadvisor**, and add **honorTimestamps: true**.

```
...
  - customBlacklist: []
    customWhitelist: []
```



```
    destNamespace: kube-system
    name: kubelet-cadvisor
    namespace: monitoring
    scrapeAllMetrics: false
    honorTimestamps: true
    scrapeInterval: ""
    status: "on"
    type: ServiceMonitor
...
```

Step 4 Click **OK** to save the configuration. The configuration takes effect in about 1 minute.

----End

Why Is 403 Displayed in Collection Endpoint Access? How Do I Handle It?

Root Cause

Authentication has been configured for collection tasks in the ServiceMonitor or PodMonitor format corresponding to your collection endpoint. For security purposes, the endpoint to be authenticated cannot be accessed by default.

Solution: You can configure to allow access to endpoints with authentication.

NOTICE

If you allow access to endpoints with authentication, the endpoints to be authenticated can be directly accessed by accessing the prometheus-lightweight service in the cluster. For this reason, do not expose the prometheus-lightweight service port outside the cluster.

1. Log in to the CCE console and click the cluster name to access the cluster console.
2. In the navigation pane, choose **ConfigMaps and Secrets**. Then, set **Namespace** to **All namespaces** and locate the **persistent-user-config** configuration item.
3. Click **Update** to edit **lightweight-user-config.yaml** and add **- --target-response-auto-auth=true** under **operatorConfigOverride**.

```
customSettings:
  operatorEnvOverride: []
  operatorConfigOverride:
  - --target-response-auto-auth=true
  promAdapterConfigOverride: []
```

4. Click **OK** to save the configuration. The configuration takes effect in about 1 minute.

10.8.3 Logging FAQ

Indexes

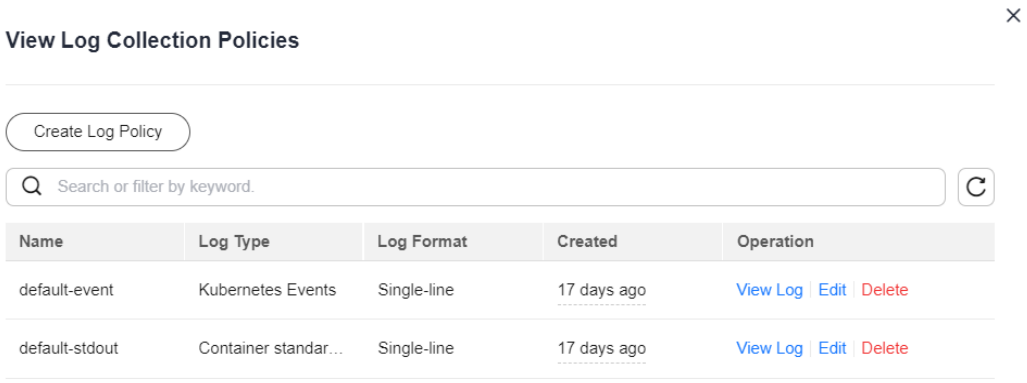
- Container File Logs Cannot Be Collected When Docker Is Used as the Container Engine
- Logs Cannot Be Reported and "log's quota has full" Is Displayed in Stdout Logs of otel
- Container File Logs Cannot Be Collected Due to the Wildcard in the Collection Directory
- fluent-bit Pod Keeps Restarting
- Logs Cannot Be Collected When the Node OS Is Ubuntu 18.04
- Job Logs Cannot Be Collected
- Cloud Native Log Collection is Running Normally, but Some Log Collection Policies Do Not Take Effect
- OOM Occurs on log-agent-otel-collector
- Some Pod Information Is Missing During Log Collection Due to Excessive Node Load
- How Do I Change the Log Storage Period on Logging?
- What Can I Do If the Log Group (Stream) in the Log Collection Policy Does Not Exist?
- Logs Cannot Be Collected After Pods Are Scheduled to CCI

How Do I Disable Logging?

Disabling container log and Kubernetes event collection

Method 1: Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**. In the upper right corner, click **View Log Collection Policies**. Then, locate and delete the corresponding log collection policy. The **default-event** policy reports Kubernetes events by default, and the **default-stdout** policy reports stdout logs by default.

Figure 10-92 Deleting a log collection policy

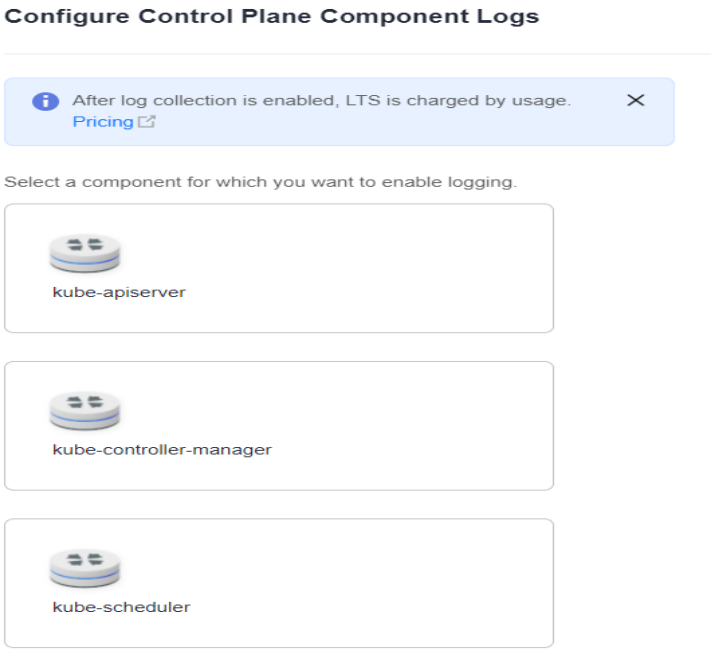


Method 2: Access the **Add-ons** page and uninstall the Cloud Native Log Collection add-on. **Note: Once you uninstall this add-on, it will no longer report Kubernetes events to AOM.**

Disabling log collection for control plane components

Choose **Logging** > **Control Plane Logs**, click **Configure Control Plane Component Logs**, and deselect one or more components whose logs do not need to be collected.

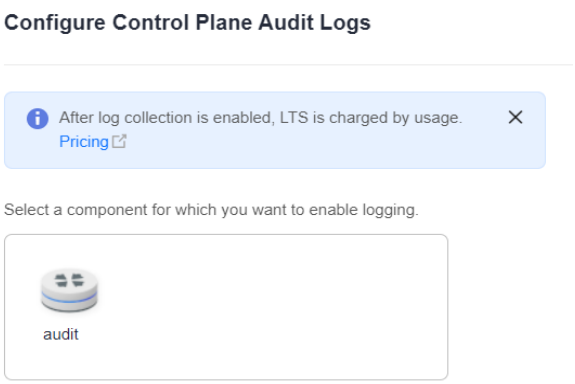
Figure 10-93 Configuring control plane component logs



Disabling control plane audit log collection

Choose **Logging** > **Control Plane Audit Logs**, click **Configure Control Plane Audit Logs**, and deselect the component whose logs do not need to be collected.

Figure 10-94 Configuring control plane audit logs



All Components Except log-operator Are Not Ready

Symptom: All components except log-operator are not ready, and the volume failed to be attached to the node.

Solution: Check the logs of log-operator. During add-on installation, the configuration files required by other components are generated by log-operator. If the configuration files are invalid, all components cannot be started.

The log information is as follows:

```
MountVolume.SetUp failed for volume "otel-collector-config-vol":configmap "log-agent-otel-collector-config" not found
```

There Is An Error in Stdout Logs of log-operator

Symptom:

```
2023/05/05 12:17:20.799 [E] call 3 times failed, reason: create group failed, projectID: xxx, groupName: k8s-log-xxx, err: create groups status code: 400, response: {"error_code":"LTS.0104","error_msg":"Failed to create log group, the number of log groups exceeds the quota"}, url: https://lts.cn-north-4.myhuaweicloud.com/v2/xxx/groups, process will retry after 45s
```

Solution: On the LTS console, delete unnecessary log groups. For details about the log group quota, see **Log Groups**.

Container File Logs Cannot Be Collected When Docker Is Used as the Container Engine

Symptom:

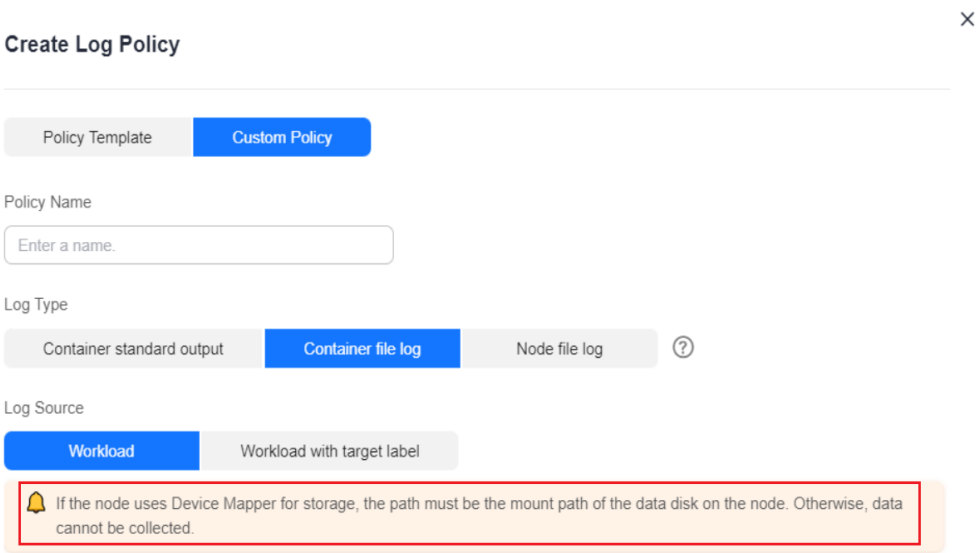
A container file path is configured but is not mounted to the container, and Docker is used as the container engine. As a result, logs cannot be collected.

Solution:

Check whether Device Mapper is used for the node where the workload resides. Device Mapper does not support text log collection. (This restriction is displayed when you create a log collection policy.) To check this, perform the following operations:

1. Go to the node where the workload resides.
2. Run the **docker info | grep "Storage Driver"** command.
3. If the value of **Storage Driver** is **Device Mapper**, text logs cannot be collected.

Figure 10-95 Creating a log policy



Logs Cannot Be Reported and "log's quota has full" Is Displayed in Stdout Logs of otel

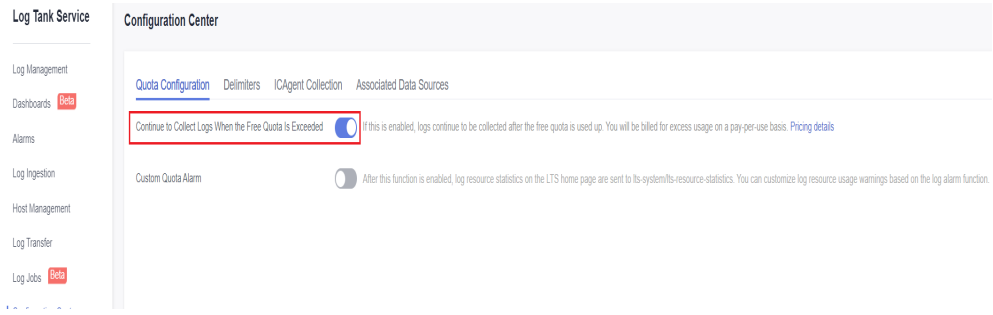
Figure 10-96 Error information of otel

```
2023-08-16T09:03:20.067+0800    error    exporterhelper/queued_retry.go:361    Exporting failed. Try enabling retry_
on_failure config option to retry on retryable errors    {"kind": "exporter", "data_type": "logs", "name": "lts/defaul
t-event", "error": "fail to push event data via lts exporter: read body {\"errorCode\":\"SVCSTG.ALS.200.210\",\"error
Message\":\"projectid                                    s quota has full!!\",\"result\":null} error"}
go.opentelemetry.io/collector/exporter/exporterhelper.(*retrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:361
go.opentelemetry.io/collector/exporter/exporterhelper.(*logsExporterWithObservability).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:142
go.opentelemetry.io/collector/exporter/exporterhelper.(*queuedRetrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:295
go.opentelemetry.io/collector/exporter/exporterhelper.NewLogsExporterWithContext.func2
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:122
go.opentelemetry.io/collector/consumer.ConsumeLogsFunc.ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/consumer/logs.go:36
go.opentelemetry.io/collector/service/internal/fanoutconsumer.(*logsConsumer).ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/service/internal/fanoutconsumer/logs.go:77
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).handleEvent
        cieotelcol/receiver/k8seventsreceiver/receiver.go:138
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).startWatch.func1
        cieotelcol/receiver/k8seventsreceiver/receiver.go:116
k8s.io/client-go/tools/cache.ResourceEventHandlerFuncs.OnAdd
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:232
k8s.io/client-go/tools/cache.processDeltas
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:441
k8s.io/client-go/tools/cache.newInformer.func1
```

Solution:

LTS provides a free log quota. If the quota is used up, you will be charged for the excess log usage. If an error message is displayed, the free quota has been used up. To continue collecting logs, log in to the LTS console, choose **Configuration Center** in the navigation pane, and enable **Continue to Collect Logs When the Free Quota Is Exceeded**.

Figure 10-97 Quota configuration



Container File Logs Cannot Be Collected Due to the Wildcard in the Collection Directory

Troubleshooting: Check the volume mounting status in the workload configuration. If a volume is attached to the data directory of a service container, this add-on cannot collect data from the parent directory. In this case, you need to set the collection directory to a complete data directory. For example, if the data

volume is attached to the **/var/log/service** directory, logs cannot be collected from the **/var/log** or **/var/log/*** directory. In this case, you need to set the collection directory to **/var/log/service**.

Solution: If the log generation directory is **/application/logs/*{Application name}*/*.log**, attach the data volume to the **/application/logs** directory and set the collection directory in the log collection policy to **/application/logs/*/*.log**.

fluent-bit Pod Keeps Restarting

Troubleshooting: Run the **kubectl describe pod** command. The output shows that the pod was restarted due to OOM. There are a large number of evicted pods on the node where the fluent-bit resides. As a result, resources are occupied, causing OOM.

Solution: Delete the evicted pods from the node.

Logs Cannot Be Collected When the Node OS Is Ubuntu 18.04

Troubleshooting: Restart the fluent-bit pod on the current node and check whether logs are properly collected. If the logs cannot be collected, check whether the log file to be collected already exists in the image during image packaging. In the container log collection scenario, the logs of the existing files during image packaging are invalid and cannot be collected. This issue is known in the community. For details, see **Issues**.

Solution: If you want to collect log files that already exist in the image during image packaging, you are advised to set **Startup Command** to **Post-Start** on the **Lifestyle** page when creating a workload. Before the pod of the workload is started, delete the original log files so that the log files can be regenerated.

Job Logs Cannot Be Collected

Troubleshooting: Check the job lifetime. If the job lifetime is less than 1 minute, the pod will be destroyed before logs are collected. In this case, logs cannot be collected.

Solution: Prolong the job lifetime.

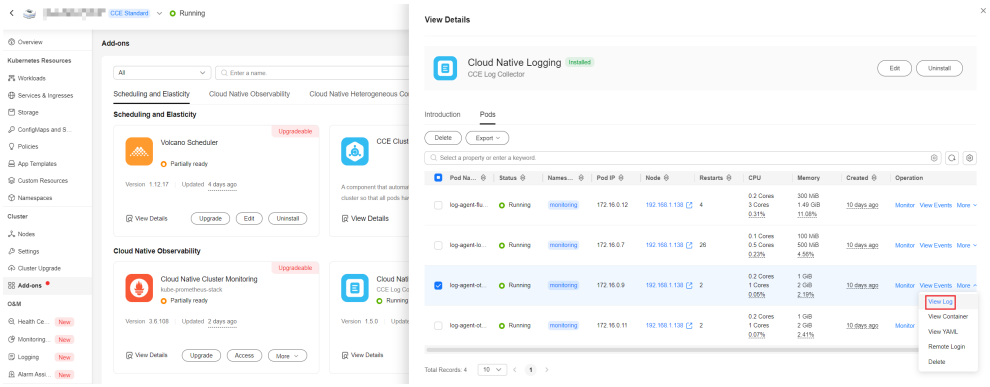
Cloud Native Log Collection is Running Normally, but Some Log Collection Policies Do Not Take Effect

Solution:

- If the log collection policy of the event type does not take effect or the add-on version is earlier than 1.5.0, check the stdout of the log-agent-otel-collector workload.

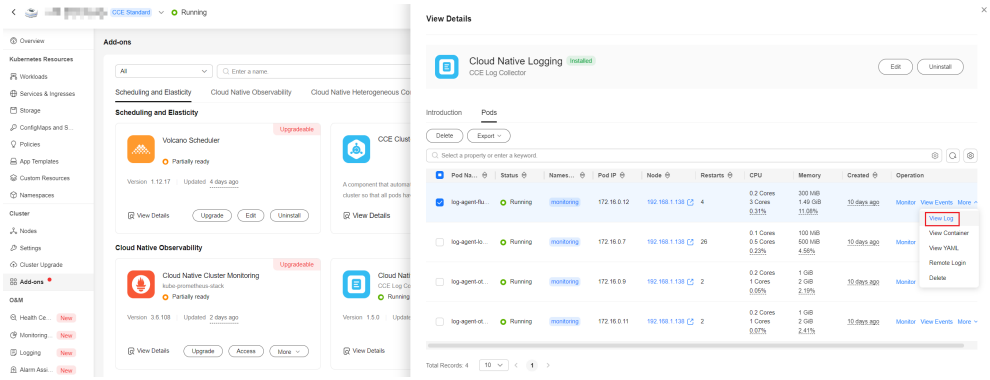
  Go to the **Add-ons** page and click the name of **Cloud Native Log Collection**. Then, click the **Pods** tab, locate log-agent-otel-collector, and choose **More** > **View Log**.

Figure 10-98 Viewing the log of the log-agent-otel-collector instance



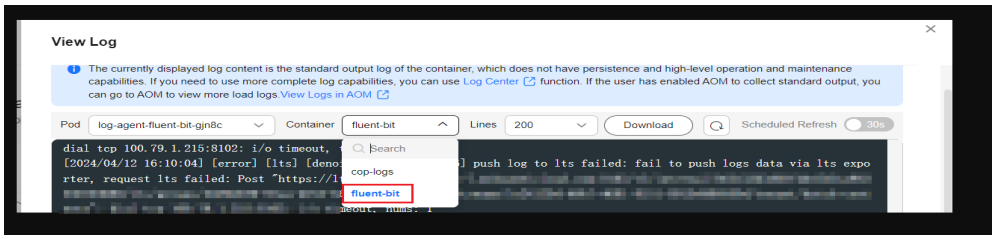
- If the log collection policy of the other type does not take effect and the add-on version is later than 1.5.0, check the log of the log-agent-fluent-bit instance on the node where the container to be monitored resides.

Figure 10-99 Viewing the log of the log-agent-fluent-bit instance



Select the fluent-bit container, search for the keyword "fail to push {event/log} data via lts exporter" in the log, and view the error message.

Figure 10-100 Viewing the log of the fluent-bit container



a. If the error message "The log streamId does not exist." is displayed, the log group or log stream does not exist. In this case, choose **Logging** > **View Log Collection Policies**, edit or delete the log collection policy, and recreate a log collection policy to update the log group or log stream.

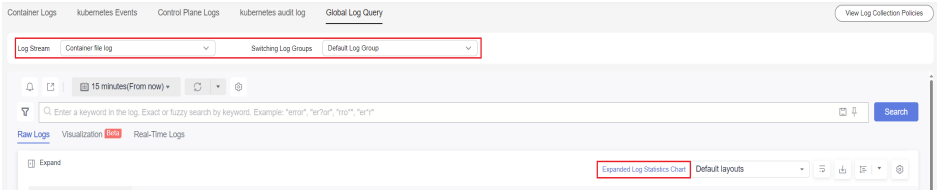
b. For other errors, go to LTS to search for the error code and view the cause.

OOM Occurs on log-agent-otel-collector

Troubleshooting:

1. View the stdout of the log-agent-otel-collector component to check whether errors occur recently.
   ```
   kubectl logs -n monitoring log-agent-otel-collector-xxx
   ```
   If an error is reported, handle the error first and ensure that logs can be collected normally.
2. If no error is reported recently and OOM still occurs, perform the following steps:
   a. Go to Logging, click the **Global Log Query** tab, and click **Expand Log Statistics Chart** to view the log statistics chart. If the reported log group and log stream are not the default ones, click the **Global Log Query** tab and select the reported log group and log stream.

      Figure 10-101 Viewing log statistics

   b. Calculate the number of logs reported per second based on the bar chart in the statistics chart and check whether the number of logs exceeds the log collection performance specification.

      If the number of logs exceeds the log collection performance specification, you can increase the number of log-agent-otel-collector copies or increase the memory upper limit of log-agent-otel-collector.
   c. If the CPU usage exceeds 90%, increase the CPU upper limit of log-agent-otel-collector.

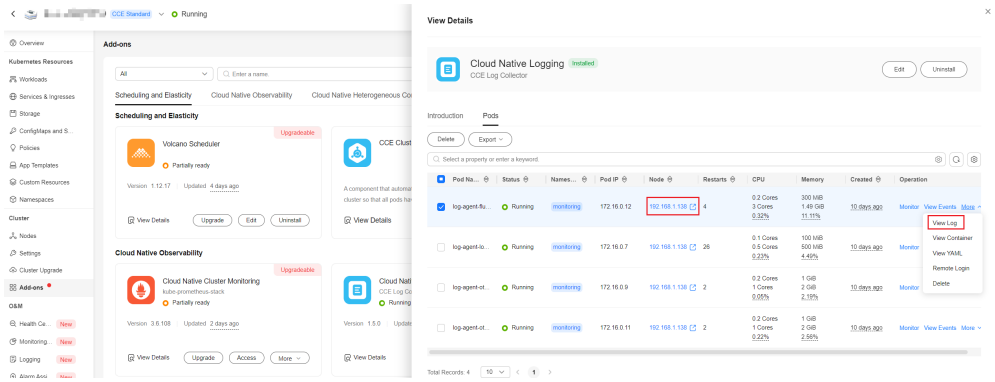
Some Pod Information Is Missing During Log Collection Due to Excessive Node Load

When the Cloud Native Log Collection add-on version is later than 1.5.0, some pod information, such as the pod ID and name, is missing from container file logs or stdout logs.

Troubleshooting:

Go to the **Add-ons** page and click the name of **Cloud Native Log Collection**. Then, click the **Pods** tab, locate the log-agent-fluent-bit of the corresponding node, and choose **More** > **View Log**.

Figure 10-102 Viewing the log of the log-agent-fluent-bit instance



Select the fluent-bit container and search for the keyword "cannot increase buffer: current=512000 requested=*** max=512000" in the log.

Figure 10-103 Viewing the log of the fluent-bit container

```
[2024/03/29 10:49:34] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:34] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:35] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
[2024/03/29 10:49:36] [ warn] [http_client] cannot increase buffer: current=512000 requested=544768 max=512000
```

Solution:

Run the **kubectl edit deploy -n monitoring log-agent-log-operator** command on the node and add **--kubernetes-buffer-size=20MB** to the command lines of the log-operator container. The default value is **16MB**. You can estimate the value based on the total size of pod information on the node. **0** indicates no limits.

CAUTION

If the Cloud Native Log Collection add-on is upgraded, you need to reconfigure **kubernetes-buffer-size**.

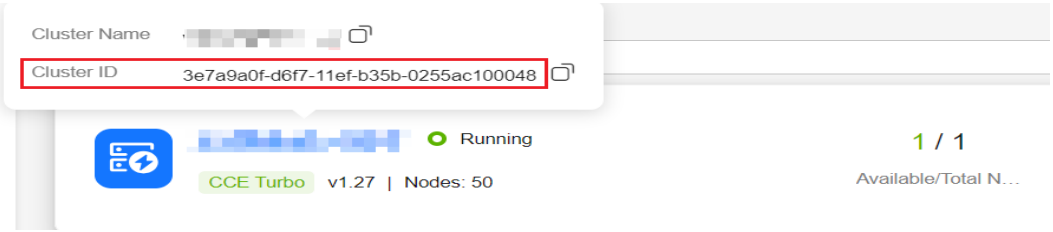
Figure 10-104 Modifying the command line parameter of the log-operator container

```
containers:
- command:
  - /var/paas/log-operator/log-operator
  - --enable-leader-election
  - --log-print-type=console
  - --enable-fluent-bit-use-kubelet=true
  - --enable-webhook=true
  - --buffer-chunk-size=128k
  - --buffer-max-size=512k
  - --mem-buf-limit=40MB
  - --aom-endpoint=https://aomperform.[redacted].com
  - --lts-endpoint=https://lts.[redacted].com
  - --lts-access-endpoint=https://lts-access.[redacted].com:8102
  - --enable-ssl=true
  - --kubernetes-buffer-size=20MB
  env:
```

How Do I Change the Log Storage Period on Logging?

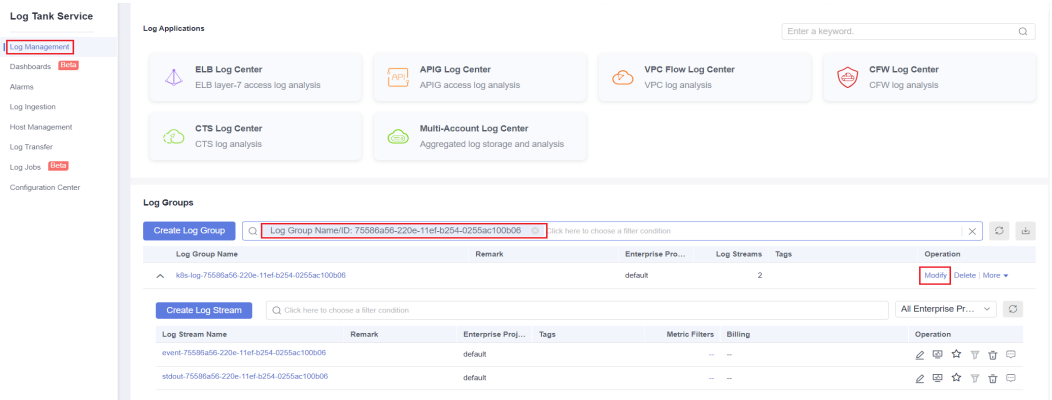
Step 1 On the **Clusters** page, hover the cursor over the cluster name to view the current cluster ID.

Figure 10-105 Viewing the cluster ID



Step 2 Log in to the LTS console. Then, query the log group and log stream by cluster ID.

Figure 10-106 Querying the log group

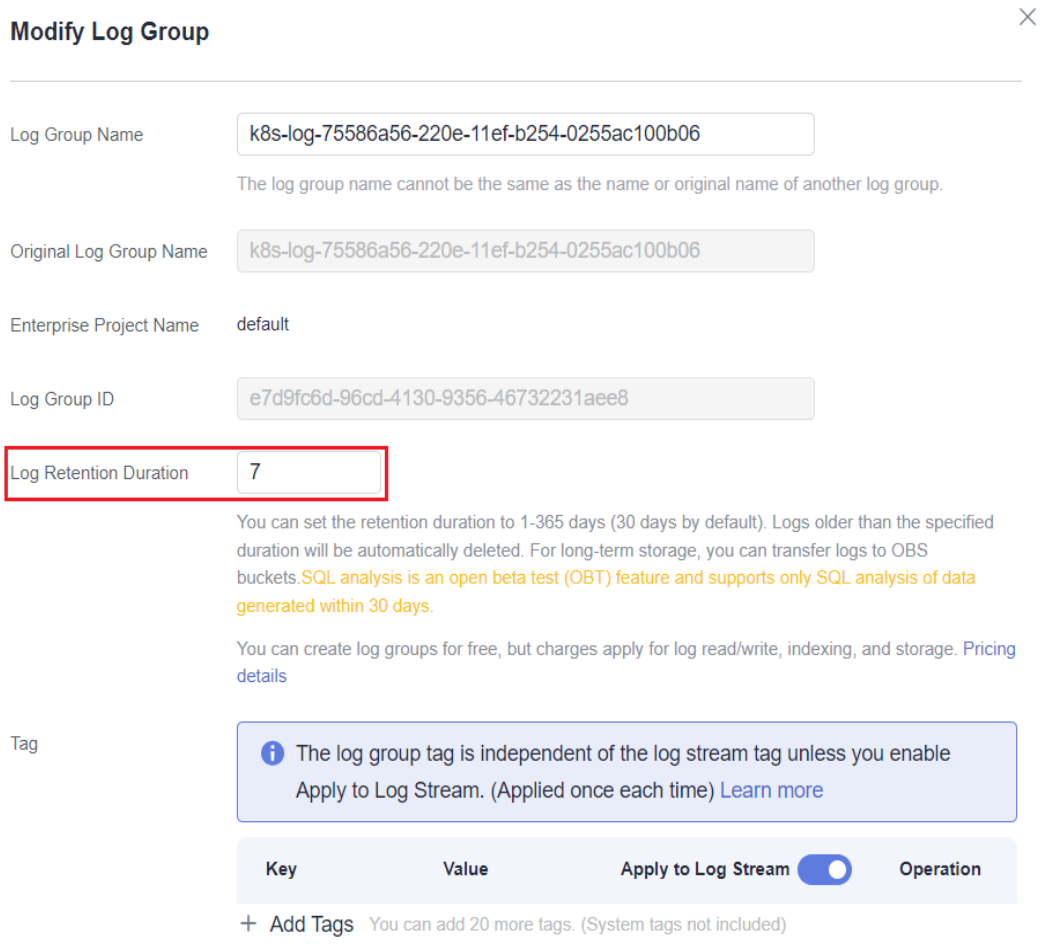


Step 3 Locate the log group and click **Modify** to configure the log storage period.

NOTE

The log retention period affects log storage expenditures.

Figure 10-107 Changing the log retention period



----End

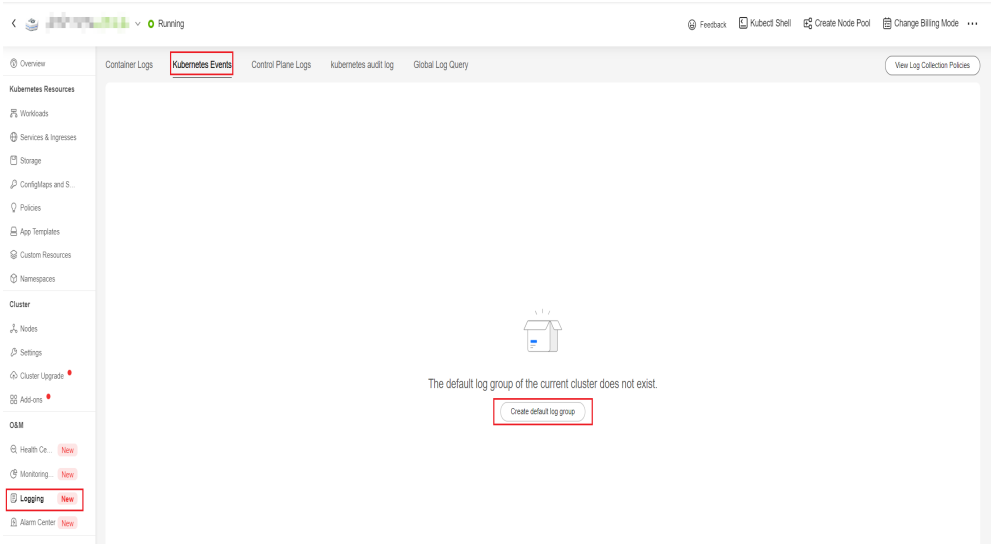
What Can I Do If the Log Group (Stream) in the Log Collection Policy Does Not Exist?

- **Scenario 1: The default log group (stream) does not exist.**

  Take Kubernetes events as an example: If the default log group (stream) does not exist, the **Kubernetes Events** page on the console displays a message indicating that the current log group (stream) does not exist. In this case, you can create the default log group (stream) again.

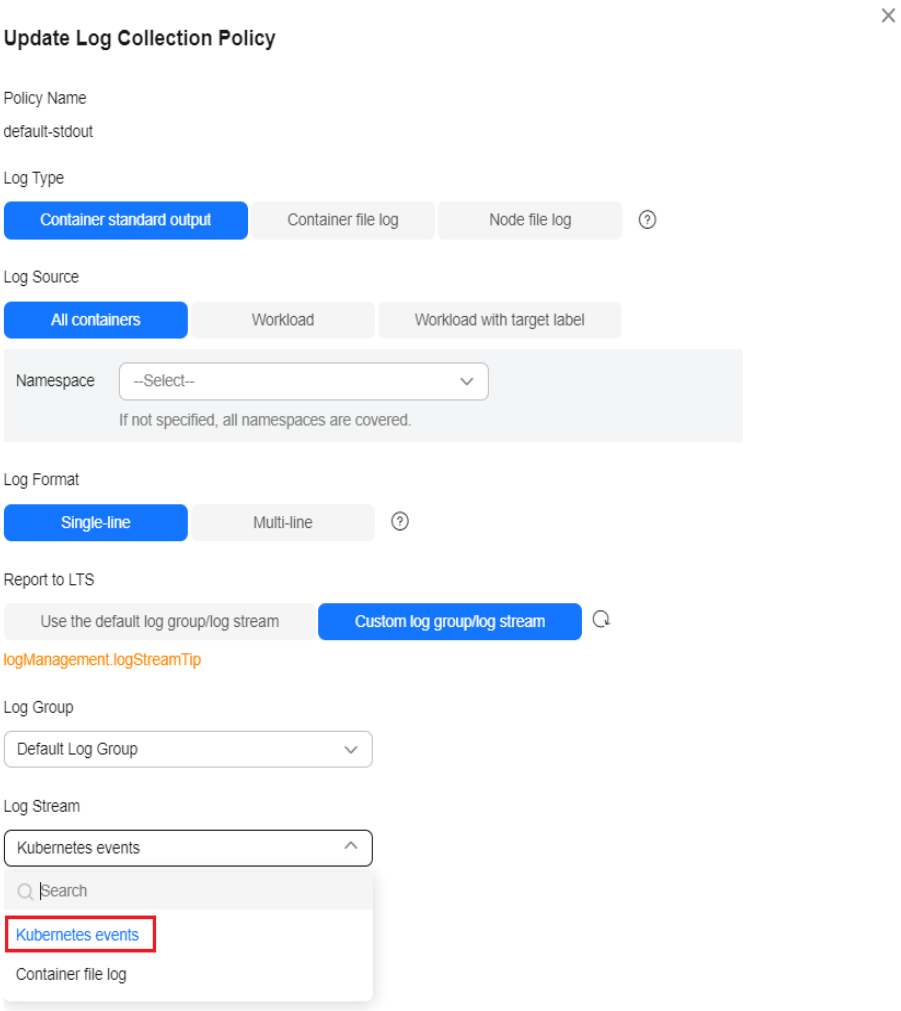
  After the recreation, the ID of the default log group (stream) changes, and the existing log collection policy of the default log group (stream) does not take effect. In this case, you can rectify the fault by referring to **Scenario 2**.

Figure 10-108 Creating a default log group (stream)



- **Scenario 2: The default log group (stream) exists but is inconsistent with the log collection policy.**
  - The log collection policy, for example, **default-stdout**, can be modified as follows:
    i. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.
    ii. In the upper right corner, click **View Log Collection Policies**. Then, locate the log collection policy and click **Edit** in the **Operation** column.
    iii. Select **Custom Log Group/Log Stream** and configure the default log group (stream).

Figure 10-109 Configuring the default log group (stream)



  - If a log collection policy cannot be modified, for example, **default-event**, you need to re-create a log collection policy as follows:

    i. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.

    ii. In the upper right corner, click **View Log Collection Policies**. Then, locate the log collection policy and click **Delete** in the **Operation** column.

    iii. Click **Create Log Collection Policy**. Then, select **Kubernetes events** in **Policy Template** and click **OK**.

- **Scenario 3: The custom log group (stream) does not exist.**

  CCE does not support the creation of non-default log groups (streams). You can create a non-default log group (stream) on the LTS console.

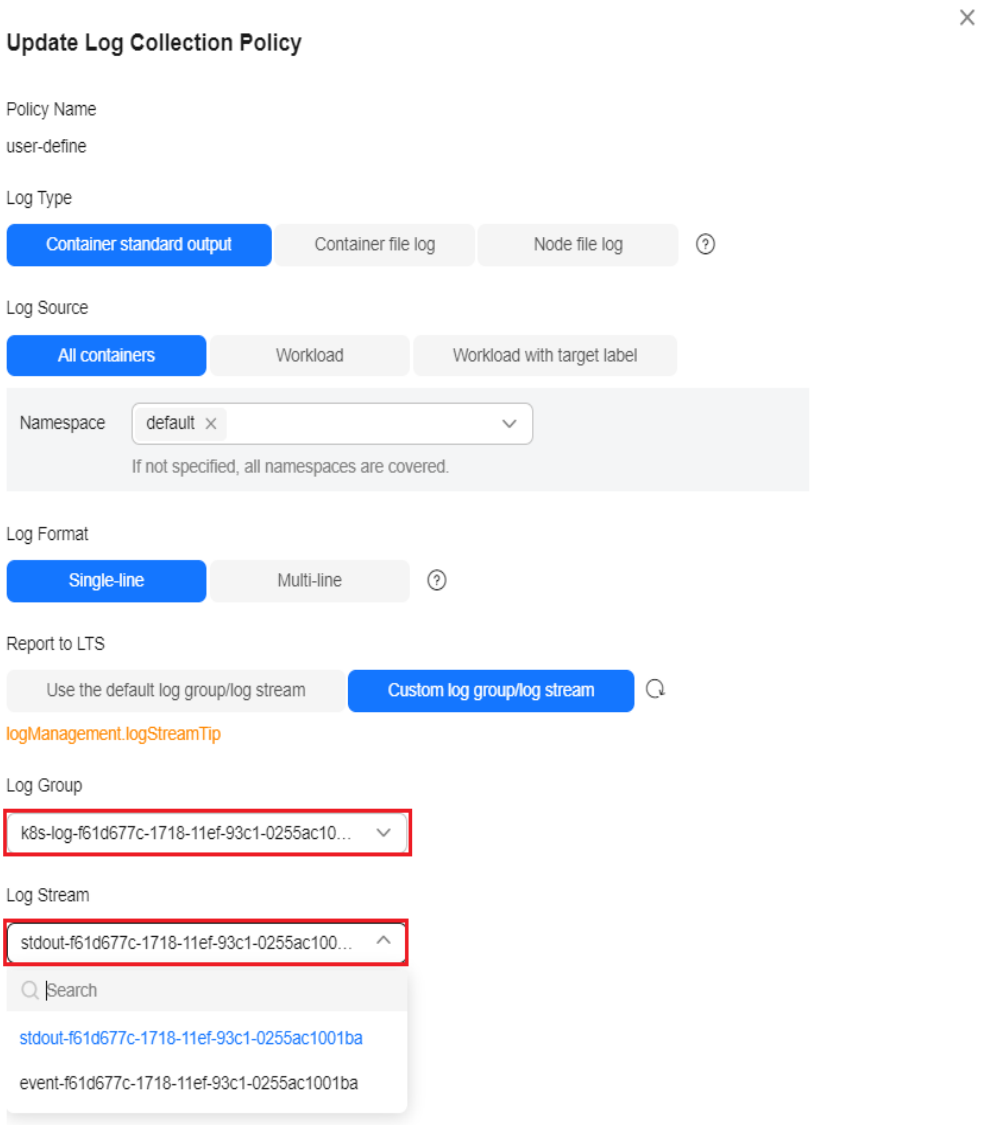
  After the creation is complete, take the following steps:

  a. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Logging**.

  b. In the upper right corner, click **View Log Collection Policies**. Then, locate the log collection policy and click **Edit** in the **Operation** column.

c. Select **Custom Log Group/Log Stream** and configure a log group (stream).

Figure 10-110 Configuring a custom log group (stream)



Logs Cannot Be Collected After Pods Are Scheduled to CCI

After **pods are scheduled to CCI by using a profile**, logs cannot be collected, but the collection policies work on the CCE console.

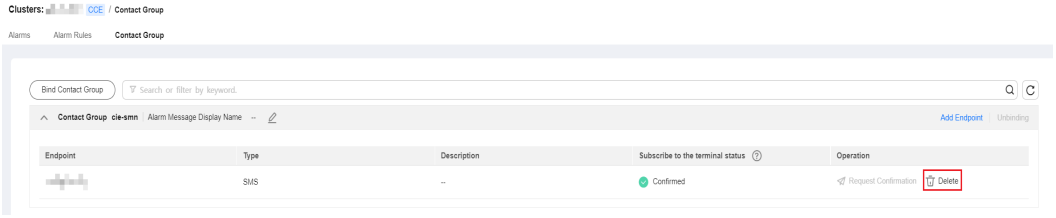
Check whether the version of the CCE Cloud Bursting Engine for CCI add-on is earlier than 1.3.54. If yes, upgrade the add-on.

10.8.4 Alarm Center FAQ

How Do I Stop Receiving Alarm Notifications?

Delete the subscribed endpoint on the **Default Contact Groups** tab (**Alarm Center** > **Default Contact Groups**).

Figure 10-111 Deleting a contact group



Why Are Alarms Still Sent After They Are Cleared?

When an alarm is cleared, only the statistics on the **Alarm Rules** tab are cleared. If the alarm continuously reaches the threshold or abnormal events occur continuously, the alarm is still generated.

Does the Contact Group of Alarm Center Can Be a Third-Party System?

Third-party systems, such as DingTalk and Lark, cannot be created on the **Default Contact Groups** tab of Alarm Center. You need to enable them on the SMN console. For details, see the **SMN documentation**.

10.9 O&M Best Practices

10.9.1 Cloud Native Cluster Monitoring Is Compatible with Self-Built Prometheus

Compatibility Mode of Cloud Native Cluster Monitoring

- If you have deployed Prometheus and your Prometheus is open-source and has not been deeply customized or integrated with your monitoring system, you are advised to uninstall it and use the Cloud Native Cluster Monitoring add-on instead. By doing so, you do not need to enable **Compatibility Mode**.

  NOTE

  To uninstall your self-built Prometheus, ensure that all workloads of your self-built Prometheus and all CRDs ending with monitoring.coreos.com are deleted. For details, see **How Do I Delete Self-Built Prometheus?**.

- If your self-built Prometheus cannot be uninstalled and you need to use functions such as Cost Insights and Monitoring Center, you can enable **Compatibility Mode** when your self-built Prometheus meets **Compatibility Requirements**. For details, see **Enabling Monitoring Center**.

  NOTE

  The compatibility mode does not support all functions of the Cloud Native Cluster Monitoring add-on. For details, see **Restrictions on the Compatibility Mode**.

Compatibility Requirements

Ensure that your self-built Prometheus meets the following requirements, or it cannot run normally.

- If your Prometheus is not based on kube-prometheus of the Operator community, the compatibility requirements are met.
- If your Prometheus is based on kube-prometheus of the Operator community, the prometheuses.monitoring.coreos.com version must be 0.8.0 or later.

  You can run the following command in the cluster to check the version:

  ```
  kubectl get crd prometheuses.monitoring.coreos.com -oyaml | grep controller-gen.kubebuilder.io/version
  ```

  If the following information is displayed, the compatibility requirements are met:

  ```
  Error from server (NotFound): customresourcedefinitions.apiextensions.k8s.io "prometheuses.monitoring.coreos.com" not found
  ```

  If the following information is displayed and the version is 0.8.0 or later, the compatibility requirements are met:

  ```
  controller-gen.kubebuilder.io/version: v0.9.2
  ```

Restrictions on the Compatibility Mode

- In compatibility mode, the Cloud Native Cluster Monitoring add-on is installed in the **cce-monitoring** namespace. By default, the ServiceMonitor and PodMonitor of another namespace are not identified.
- The Cloud Native Cluster Monitoring add-on can only be deployed without local data storage.
- You cannot disable compatibility mode once it is enabled. To use the normal mode, you can uninstall the Cloud Native Cluster Monitoring add-on first and install it again.
- Currently, cost optimization is not supported.
- Currently, metrics discarding on the AOM page is not supported.
- Currently, the ServiceMonitor and PodMonitor cannot be started or stopped on the AOM page.

How Do I Delete Self-Built Prometheus?

Step 1 Delete all workloads, services, and other resources related to self-built Prometheus, such as DaemonSets, Deployments, StatefulSets, and Services. Generally, these resources are in the **monitoring** namespace. Replace it with the actual namespace used during the add-on installation.

Step 2 Query all CRDs ending with monitoring.coreos.com.

```
kubectl get crd | grep monitoring.coreos.com | awk '{print $1}' | xargs
```

If the command output is empty, no further action is required.

Step 3 Delete all CRDs ending with monitoring.coreos.com.

```
kubectl delete crd $(kubectl get crd | grep monitoring.coreos.com | awk '{print $1}' | xargs)
```

----End

10.9.2 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring

CCE provides the Cloud Native Cluster Monitoring add-on to monitor custom metrics using Prometheus.

The following procedure uses an Nginx application as an example to describe how to use Prometheus to monitor custom metrics:

1. **Installing and Accessing Cloud Native Cluster Monitoring**

   CCE provides an add-on that integrates Prometheus functions. You can install it with several clicks.

2. **Preparing an Application**

   Prepare an application image. The application must provide a metric monitoring API for Prometheus to collect data, and the monitoring data must **comply with the Prometheus specifications**.

3. Monitoring Custom Metrics

   Use the application image to deploy a workload in a cluster. Custom metrics will be automatically reported to Prometheus.

   Use one of the following methods to monitor custom metrics:

   - **Method 1: Configuring Pod Annotations**
   - **Method 2: Configuring Service Annotations**
   - **Method 3: Configuring PodMonitor**
   - **Method 4: Configuring ServiceMonitor**
   - **Method 5: Configuring AdditionalScrapeConfigs**

Custom Metric Billing

After Cloud Native Cluster Monitoring is interconnected with AOM, metrics will be reported to the AOM instance you select. **Basic metrics** can be monitored for free, but custom metrics are billed based on the standard pricing of AOM. For details, see **Pricing Details**.

Prometheus Monitoring Data Collection

Prometheus periodically calls the metric monitoring API (**/metrics** by default) of an application to obtain monitoring data. The application needs to provide the metric monitoring API for Prometheus to call, and the monitoring data must meet the following specifications of Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus provides clients in various languages. For details about the clients, see **Prometheus CLIENT LIBRARIES**. For details about how to develop an exporter, see **WRITING EXPORTERS**. The Prometheus community provides various third-party exporters that can be directly used. For details, see **EXPORTERS AND INTEGRATIONS**.

Constraints

- To use Prometheus to monitor custom metrics, the application needs to provide a metric monitoring API. For details, see **Prometheus Monitoring Data Collection**.

- Currently, metrics in the **kube-system** and **monitoring** namespaces cannot be collected when pod and service annotations are used. To collect metrics in the two namespaces, use PodMonitor and ServiceMonitor.
- The nginx/nginx-prometheus-exporter:0.9.0 image is pulled for the Nginx application. You need to add an EIP for the node where the application is deployed or upload the image to SWR to prevent application deployment failures.

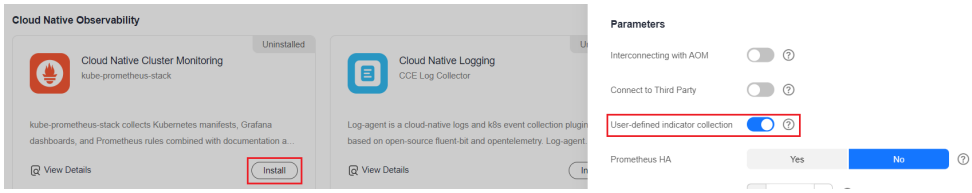
Installing and Accessing Cloud Native Cluster Monitoring

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Add-ons**. On the displayed page, locate **Cloud Native Cluster Monitoring** and click **Install**. In addition to the monitoring capabilities, this add-on interconnects monitoring data with Monitoring Center.

When installing this add-on, pay attention to the following configurations. Configure other parameters as required. For details, see **Cloud Native Cluster Monitoring**.

- For 3.8.0 or later, enable custom metric collection.



- For 3.8.0 or earlier, do not enable custom metric collection.

Step 3 After this add-on is installed, deploy workloads and Services. The Prometheus server will be deployed as a StatefulSet in the **monitoring** namespace.

You can create a public network **LoadBalancer Service** so that Prometheus can be accessed from external networks.

1. Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.
2. Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service.

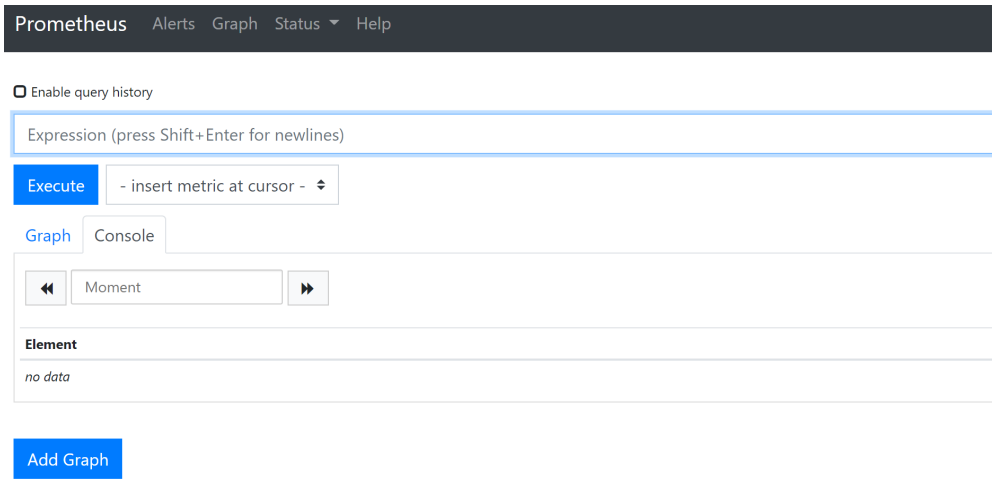
```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb     # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace 038ff*** with the ID of the public network load balancer in the VPC that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88            # Service port, which is customizable.
      targetPort: 9090     # Default Prometheus port. Retain the default value.
  selector:               # The label selector can be adjusted based on the label of a Prometheus server instance.
```



```
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

3. After the Service is created, enter *Public IP address of the load balancer*:*Service port* in the address box of the browser to access Prometheus.

Figure 10-112 Accessing Prometheus



----End

Preparing an Application

User-developed applications must provide a metric monitoring API, and the monitoring data must comply with the Prometheus specifications. For details, see **Prometheus Monitoring Data Collection**.

This section uses Nginx as an example to describe how to collect monitoring data. There is a module named **ngx_http_stub_status_module** in Nginx, which provides basic monitoring functions. You can configure the **nginx.conf** file to provide an interface for external systems to access Nginx monitoring data.

Step 1 Log in to a Linux VM that can access to the Internet and run Docker commands.

Step 2 Create an **nginx.conf** file. Add the server configuration under **http** to enable Nginx to provide an interface for the external systems to access the monitoring data.

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
```



```
    sendfile        on;
    #tcp_nopush     on;
    keepalive_timeout  65;
    #gzip  on;
    include /etc/nginx/conf.d/*.conf;

    server {
      listen 8080;
      server_name  localhost;
      location /stub_status {
         stub_status on;
         access_log off;
      }
    }
}
```

Step 3 Use this configuration to create an image and a Dockerfile file.

```
vi Dockerfile
```

The content of Dockerfile is as follows:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Step 4 Use this Dockerfile to create an image and upload it to SWR. The image name is **nginx:exporter**. For details about how to upload an image, see **Uploading an Image Through a Container Engine Client**.

1. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog box, click **Generate a temporary login command**. Then, click  to copy the command.
2. Run the login command copied in the previous step on the node. If the login is successful, the message "Login Succeeded" is displayed.
3. Run the following command to build an image named nginx. The image version is exporter.

   ```
   docker build -t nginx:exporter .
   ```

4. Tag the image and upload it to the image repository. Change the image repository address and organization name based on your requirements.

   ```
   docker tag nginx:exporter swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   docker push swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   ```

Step 5 View application metrics.

1. Use **nginx:exporter** to create a workload.
2. **Access the container** and use http://<ip_address>:8080/stub_status to obtain nginx monitoring data. **<ip_address>** indicates the IP address of the container. Information similar to the following is displayed.

   ```
   # curl http://127.0.0.1:8080/stub_status
   Active connections: 3
   server accepts handled requests
    146269 146269 212
   Reading: 0 Writing: 1 Waiting: 2
   ```

----End

Method 1: Configuring Pod Annotations

When the annotation settings of pods comply with the Prometheus data collection rules, Prometheus automatically collects the metrics exposed by the pods.

The format of the monitoring data provided by **nginx:exporter** does not meet the requirements of Prometheus. Convert the data format to the format required by Prometheus. To convert the format of Nginx metrics, use **nginx-prometheus-exporter**. Deploy **nginx:exporter** and **nginx-prometheus-exporter** in the same pod and add the following annotations during deployment. Then Prometheus can automatically collect metrics.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "9113"
        prometheus.io/path: "/metrics"
        prometheus.io/scheme: "http"
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter'      # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

Where,

- **prometheus.io/scrape** indicates whether to enable Prometheus to collect pod monitoring data. The value is **true**.
- **prometheus.io/port** indicates the port for collecting monitoring data, which varies depending on the application. In this example, the port is 9113.
- **prometheus.io/path** indicates the URL of the API for collecting monitoring data. If this parameter is not set, the default value **/metrics** is used.
- **prometheus.io/scheme**: protocol used for data collection. The value can be **http** or **https**.

After the application is successfully deployed, **access Prometheus** to query custom metrics by job name.

The custom metrics related to Nginx can be queried. In the following, the job name indicates that the metrics are reported based on the pod configuration.

```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE",
cluster_name="cce-test", container="container-0", instance="10.0.0.46:9113", job="monitoring/kubernetes-
```



```
pods", kubernetes_namespace="default", kubernetes_pod="nginx-exporter-77bf4d4948-zsb59",
namespace="default", pod="nginx-exporter-77bf4d4948-zsb59", prometheus="monitoring/server"}
```

Method 2: Configuring Service Annotations

When the annotation settings of Services comply with the Prometheus data collection rules, Prometheus automatically collects the metrics exposed by the Services.

You can use Service annotations in the same way as pod annotations. However, their application scenarios are different. Pod annotations focus on pod resource usage metrics while Service annotations focus on metrics such as requests for a Service.

The following is an example configuration:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-test
  template:
    metadata:
      labels:
        app: nginx-test
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter'      # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

The following is an example Service configuration:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-test
  labels:
    app: nginx-test
  namespace: default
  annotations:
    prometheus.io/scrape: "true"  # Value true indicates that service discovery is enabled.
    prometheus.io/port: "9113"  # Set it to the port on which metrics are exposed.
    prometheus.io/path: "/metrics" # Enter the URI path under which metrics are exposed. Generally, the
value is /metrics.
spec:
  selector:
    app: nginx-test
```



```
externalTrafficPolicy: Cluster
ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 8080
    protocol: TCP
  - name: cce-service-1
    protocol: TCP
    port: 9113
    targetPort: 9113
type: NodePort
```

After the application is successfully deployed, **access Prometheus** to query custom metrics. In the following, the Service name indicates the metrics are reported based on the Service configuration.

```
nginx_connections_accepted{app="nginx-test", cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE", cluster_name="cce-test", instance="10.0.0.38:9113", job="nginx-test", kubernetes_namespace="default", kubernetes_service="nginx-test", namespace="default", pod="nginx-test-78cfb65889-gtv7z", prometheus="monitoring/server", service="nginx-test"}
```

Method 3: Configuring PodMonitor

Cloud Native Cluster Monitoring allows you to configure metric collection tasks based on PodMonitor and ServiceMonitor. Prometheus Operator watches PodMonitor. The reload mechanism of Prometheus is used to trigger a hot update of the Prometheus collection tasks to the Prometheus instance.

To use CRDs defined by Prometheus Operator on GitHub, visit **https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/charts/crds/crds**.

The following is an example configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-test2
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-test2
  template:
    metadata:
      labels:
        app: nginx-test2
    spec:
      containers:
      - image: nginx:exporter     # Replace it with the address of the image you uploaded to SWR.
        name: container-0
        ports:
        - containerPort: 9113     # Port on which metrics are exposed.
          name: nginx-test2       # Application name used when PodMonitor is configured.
          protocol: TCP
        resources:
          limits:
            cpu: 250m
            memory: 300Mi
          requests:
            cpu: 100m
            memory: 100Mi
      - name: container-1
        image: 'nginx/nginx-prometheus-exporter:0.9.0'
```



```
      command:
        - nginx-prometheus-exporter
      args:
        - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
    imagePullSecrets:
      - name: default-secret
```

The following is an example PodMonitor configuration:

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: podmonitor-nginx   # PodMonitor name
  namespace: monitoring    # Namespace that PodMonitor belongs to. monitoring is recommended.
spec:
  namespaceSelector:       # An selector matching the namespace where the workload is located
    matchNames:
    - default              # Namespace that the workload belongs to
  jobLabel: podmonitor-nginx
  podMetricsEndpoints:
  - interval: 15s
    path: /metrics           # Path under which metrics are exposed by the workload
    port: nginx-test2        # Port on which metrics are exposed by the workload
    tlsConfig:
      insecureSkipVerify: true
  selector:
    matchLabels:
      app: nginx-test2   # Label carried by the pod, which can be selected by the selector
```

After the application is successfully deployed, **access Prometheus** to query custom metrics. In the following, the job name indicates the metrics are reported based on the PodMonitor configuration.

```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE",
cluster_name="cce-test", container="container-0", endpoint="nginx-test2", instance="10.0.0.44:9113",
job="monitoring/podmonitor-nginx", namespace="default", pod="nginx-test2-746b7f8fdd-krzfp",
prometheus="monitoring/server"}
```

Method 4: Configuring ServiceMonitor

Cloud Native Cluster Monitoring allows you to configure metric collection tasks based on PodMonitor and ServiceMonitor. Prometheus Operator watches ServiceMonitor. The reload mechanism of Prometheus is used to trigger a hot update of the Prometheus collection tasks to the Prometheus instance.

To use CRDs defined by Prometheus Operator on GitHub, visit **https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/charts/crds/crds**.

The following is an example configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-test3
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-test3
  template:
    metadata:
      labels:
        app: nginx-test3
    spec:
```



```
    containers:
    - image: nginx:exporter       # Replace it with the address of the image you uploaded to SWR.
      name: container-0
      resources:
        limits:
          cpu: 250m
          memory: 300Mi
        requests:
          cpu: 100m
          memory: 100Mi
    - name: container-1
      image: 'nginx/nginx-prometheus-exporter:0.9.0'
      command:
        - nginx-prometheus-exporter
      args:
        - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
    imagePullSecrets:
      - name: default-secret
```

The following is an example Service configuration:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-test3
  labels:
    app: nginx-test3
  namespace: default
spec:
  selector:
    app: nginx-test3
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 8080
      protocol: TCP
    - name: servicemonitor-ports
      protocol: TCP
      port: 9113
      targetPort: 9113
  type: NodePort
```

The following is an example ServiceMonitor configuration:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: servicemonitor-nginx
  namespace: monitoring
spec:
  # Configure the name of the port on which metrics are exposed.
  endpoints:
  - path: /metrics
    port: servicemonitor-ports
  jobLabel: servicemonitor-nginx
  # Application scope of a collection task. If this parameter is not set, the default value default is used.
  namespaceSelector:
    matchNames:
    - default
  selector:
    matchLabels:
      app: nginx-test3
```

After the application is successfully deployed, **access Prometheus** to query custom metrics. In the following, the endpoint name indicates the metrics are reported based on the ServiceMonitor configuration.


```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE", cluster_name="cce-test", endpoint="servicemonitor-ports", instance="10.0.0.47:9113", job="nginx-test3", namespace="default", pod="nginx-test3-6f8bccd9-f27hv", prometheus="monitoring/server", service="nginx-test3"}
```

Method 5: Configuring AdditionalScrapeConfigs

NOTICE

Cloud Native Cluster Monitoring 3.10.1 or later has been installed.

AdditionalScrapeConfigs allows you to specify a secret key to attach your additional Prometheus scrape configuration to Cloud Native Cluster Monitoring.

This mechanism bypasses the common scrape configuration generation logic and directly transfers the configuration to Prometheus. Therefore, you need to ensure that the configuration is correct. You are advised to refer to the official **scrape_config documentation**.

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Use YAML to create the following secret:

```
kind: Secret
apiVersion: v1
type: Opaque
metadata:
  name: additional-scrape-configs
  namespace: monitoring  # monitoring is only an example. The namespace must be the same as that of Cloud Native Cluster Monitoring.
stringData:
  # The following is a metric collection example of Cloud Native Cluster Monitoring without local data storage. You need to replace the settings as needed.
  prometheus-additional.yaml: |-
    - job_name: custom-job-test
      metrics_path: /metrics
      relabel_configs:
      - action: keep
        source_labels:
        - __meta_kubernetes_pod_label_app
        - __meta_kubernetes_pod_labelpresent_app
        regex: (prometheus-lightweight);true
      - action: keep
        source_labels:
        - __meta_kubernetes_pod_container_port_name
        regex: web
      kubernetes_sd_configs:
      - role: pod
        namespaces:
          names:
          - monitoring
```

Step 3 Edit the **persistent-user-config** configuration item to enable AdditionalScrapeConfigs.

```
kubectl edit configmap persistent-user-config -n monitoring
```

Add *--common.prom.default-additional-scrape-configs-key=prometheus-additional.yaml* under **operatorConfigOverride** to enable AdditionalScrapeConfigs as follows:

```
...
data:
```



```
lightweight-user-config.yaml: |
  customSettings:
    additionalScrapeConfigs: []
    agentExtraArgs: []
    metricsDeprecated:
      globalDeprecateMetrics: []
    nodeExporterConfigOverride: []
    operatorConfigOverride:
    - --common.prom.default-additional-scrape-configs-key=prometheus-additional.yaml
...
```

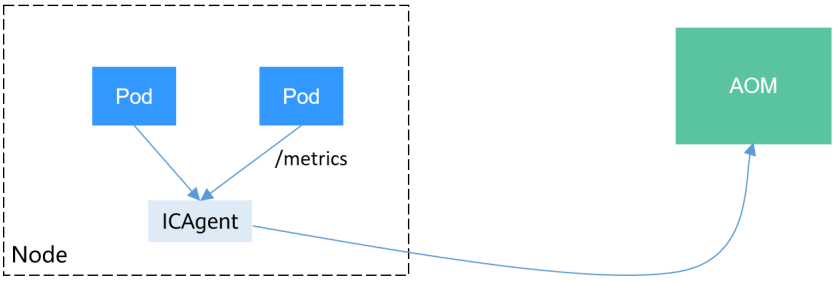
Step 4 Go to the Grafana or AOM page to check whether your custom metrics are collected.

----End

10.9.3 Monitoring Custom Metrics on AOM

CCE allows you to upload custom metrics to AOM. ICAgent on a node periodically calls the metric monitoring API configured on a workload to read monitoring data and then uploads the data to AOM.

Figure 10-113 Using ICAgent to collect monitoring metrics



The custom metric API of a workload can be configured when the workload is created. The following procedure uses an Nginx application as an example to describe how to report custom metrics to AOM.

1. **Preparing an Application**

   Prepare an application image. The application must provide a metric monitoring API for ICAgent to collect data, and the monitoring data must **comply with the Prometheus specifications**.

2. **Deploying Applications and Converting Nginx Metrics**

   Use the application image to deploy a workload in a cluster. Custom metrics are automatically reported.

3. **Verification**

   Go to AOM to check whether the custom metrics are successfully collected.

Constraints

- The ICAgent is compatible with the monitoring data specifications of **Prometheus**. The custom metrics provided by pods can be collected by the ICAgent only when they meet the monitoring data specifications of Prometheus. For details, see **Prometheus Monitoring Data Collection**.
- The ICAgent supports only **Gauge** metrics.

- The interval for the ICAgent to call the custom metric API is 1 minute, which cannot be changed.

Prometheus Monitoring Data Collection

Prometheus periodically calls the metric monitoring API (**/metrics** by default) of an application to obtain monitoring data. The application needs to provide the metric monitoring API for Prometheus to call, and the monitoring data must meet the following specifications of Prometheus:

```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

Prometheus provides clients in various languages. For details about the clients, see **Prometheus CLIENT LIBRARIES**. For details about how to develop an exporter, see **WRITING EXPORTERS**. The Prometheus community provides various third-party exporters that can be directly used. For details, see **EXPORTERS AND INTEGRATIONS**.

Preparing an Application

User-developed applications must provide a metric monitoring API, and the monitoring data must comply with the Prometheus specifications. For details, see **Prometheus Monitoring Data Collection**.

This section uses Nginx as an example to describe how to collect monitoring data. There is a module named **ngx_http_stub_status_module** in Nginx, which provides basic monitoring functions. You can configure the **nginx.conf** file to provide an interface for external systems to access Nginx monitoring data.

Step 1 Log in to a Linux VM that can access to the Internet and run Docker commands.

Step 2 Create an **nginx.conf** file. Add the server configuration under **http** to enable Nginx to provide an interface for the external systems to access the monitoring data.

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    #tcp_nopush     on;
    keepalive_timeout  65;
    #gzip  on;
    include /etc/nginx/conf.d/*.conf;
```



```
    server {
      listen 8080;
      server_name  localhost;
      location /stub_status {
        stub_status on;
        access_log off;
      }
    }
}
```

Step 3 Use this configuration to create an image and a Dockerfile file.

```
vi Dockerfile
```

The content of Dockerfile is as follows:

```
FROM nginx:1.21.5-alpine
ADD nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Step 4 Use this Dockerfile to create an image and upload it to SWR. The image name is **nginx:exporter**. For details about how to upload an image, see **Uploading an Image Through a Container Engine Client**.

1. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog box, click **Generate a temporary login command**. Then, click  to copy the command.
2. Run the login command copied in the previous step on the node. If the login is successful, the message "Login Succeeded" is displayed.
3. Run the following command to build an image named nginx. The image version is exporter.

   ```
   docker build -t nginx:exporter .
   ```

4. Tag the image and upload it to the image repository. Change the image repository address and organization name based on your requirements.

   ```
   docker tag nginx:exporter swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   docker push swr.ap-southeast-1.myhuaweicloud.com/dev-container/nginx:exporter
   ```

Step 5 View application metrics.

1. Use **nginx:exporter** to create a workload.
2. **Access the container** and use http://<ip_address>:8080/stub_status to obtain nginx monitoring data. **<ip_address>** indicates the IP address of the container. Information similar to the following is displayed.

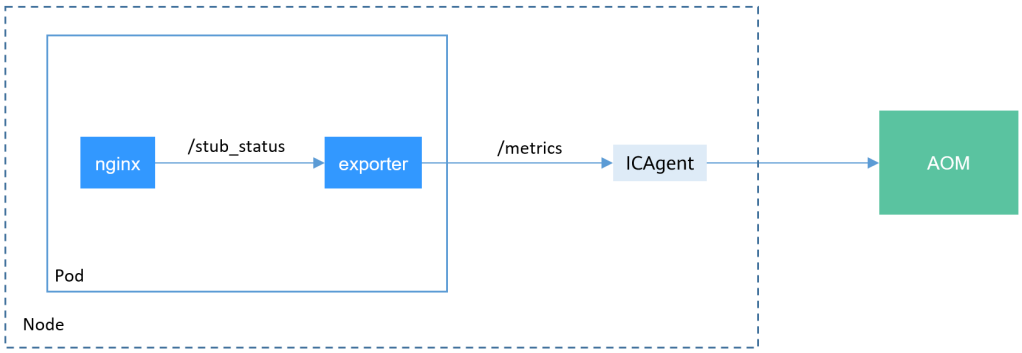
   ```
   # curl http://127.0.0.1:8080/stub_status
   Active connections: 3
   server accepts handled requests
    146269 146269 212
   Reading: 0 Writing: 1 Waiting: 2
   ```

----End

Deploying Applications and Converting Nginx Metrics

The format of the monitoring data provided by **nginx:exporter** does not meet the requirements of Prometheus. Convert the data format to the format required by Prometheus. To convert the format of Nginx metrics, use **nginx-prometheus-exporter**, as shown in the following figure.

Figure 10-114 Using exporter to convert the data format



Deploy **nginx:exporter** and **nginx-prometheus-exporter** in the same pod.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx-exporter
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-exporter
  template:
    metadata:
      labels:
        app: nginx-exporter
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/
metrics","port":"9113","names":""}]'
    spec:
      containers:
        - name: container-0
          image: 'nginx:exporter'  # Replace it with the address of the image you uploaded to SWR.
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
        - name: container-1
          image: 'nginx/nginx-prometheus-exporter:0.9.0'
          command:
            - nginx-prometheus-exporter
          args:
            - '-nginx.scrape-uri=http://127.0.0.1:8080/stub_status'
      imagePullSecrets:
        - name: default-secret
```

NOTE

The **nginx/nginx-prometheus-exporter:0.9.0** image needs to be pulled from the public network. Therefore, a public IP address needs to be bound to each node in the cluster.

nginx-prometheus-exporter requires a startup command. **nginx-prometheus-exporter -nginx.scrape-uri=http://127.0.0.1:8080/stub_status** is used to obtain Nginx monitoring data.

In addition, add an annotation **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"9113","names":""}]'** to the pod.

Verification

After an application is deployed, you can access Nginx to construct some access data and check whether the corresponding monitoring data can be obtained in AOM.

Step 1 Obtain the pod name of Nginx.

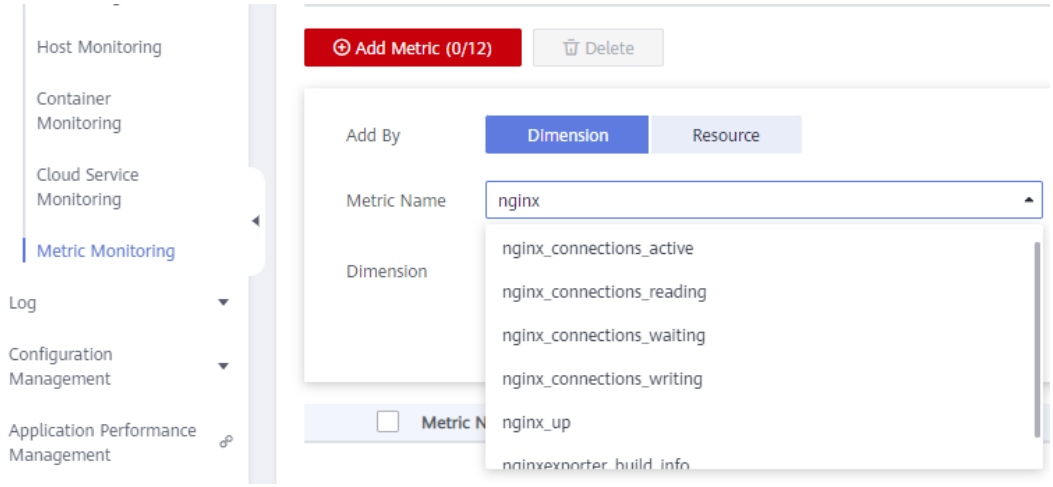
```
$ kubectl get pod
NAME                              READY   STATUS    RESTARTS   AGE
nginx-exporter-78859765db-6j8sw   2/2     Running   0          4m
```

Step 2 Log in to the container and run commands to access Nginx.

```
$ kubectl exec -it nginx-exporter-78859765db-6j8sw -- /bin/sh
Defaulting container name to container-0.
Use 'kubectl describe pod/nginx-exporter-78859765db-6j8sw -n default' to see all of the containers in this pod.
/ # curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Step 3 Log in to AOM. In the navigation pane, choose **Monitoring** > **Metric Monitoring** to view Nginx-related metrics, for example, **nginx_connections_active**.

Figure 10-115 Viewing monitoring metrics



----End

10.9.4 Monitoring Metrics of Master Node Components Using Prometheus

This section describes how to use Prometheus to monitor the kube-apiserver, kube-controller, kube-scheduler and etcd-server components on the master node.

Viewing the Metrics of Master Node Components in Monitoring Center

Monitoring Center can monitor the kube-apiserver component on the master node. After enabling Monitoring Center (**Cloud Native Cluster Monitoring** 3.5.0 or later installed) in the cluster, you can view the API metrics in the API server view (**API Server View**) on the dashboard.

To monitor the kube-controller, kube-scheduler, and etcd-server components, perform the following steps.

NOTE

The basic container metrics do not contain the metrics of the kube-controller, kube-scheduler, and etcd-server components. After Monitoring Center reports the metrics of the three components to AOM, you will be charged. Therefore, Monitoring Center does not collect these component metrics by default.

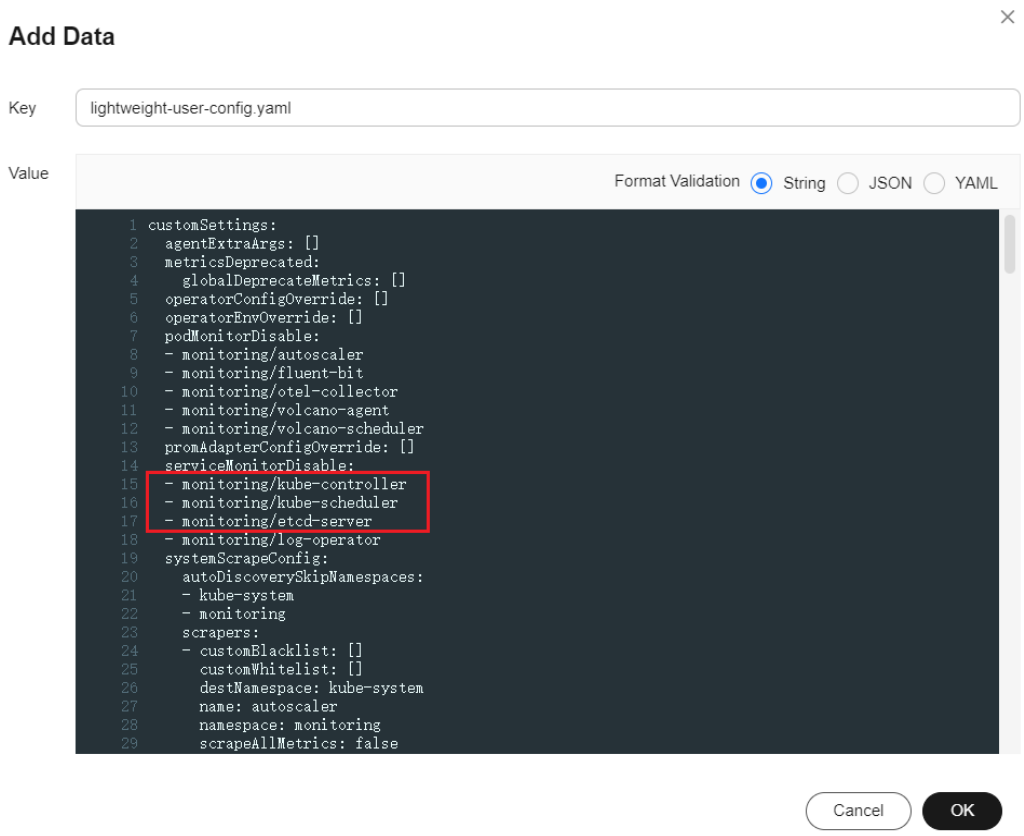
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **ConfigMaps and Secrets**, switch to the **monitoring** namespace, and locate the **persistent-user-config** configuration item.

Step 3 Click **Update** to edit the configuration data and delete the following configuration from the **serviceMonitorDisable** field.

```
serviceMonitorDisable:
- monitoring/kube-controller
- monitoring/kube-scheduler
- monitoring/etcd-server
- monitoring/log-operator
```


Figure 10-116 Deleting the configuration



Step 4 Click **OK**.

Step 5 Wait for five minutes. Then, go to the AOM console, locate the AOM instance reported by the cluster on the **Monitoring** > **Metric Monitoring** page, and view the metrics of the preceding components.

Figure 10-117 Viewing metrics

----End

Collecting Metrics of Master Node Components Using Self-Built Prometheus

This section describes how to collect metrics of master node components using self-built Prometheus.

NOTICE

- The cluster version must be 1.19 or later.
- You need to install self-built Prometheus using Helm by referring to **Prometheus**. You need to use prometheus-operator to manage installed Prometheus by referring to **Prometheus Operator**.

  The Prometheus (**Prometheus (EOM)**) add-on is end of maintenance and does not support this function. Therefore, do not use this add-on.

Step 1 Use **kubectl** to connect to the cluster.

Step 2 Modify the ClusterRole of Prometheus.

```
kubectl edit ClusterRole prometheus -n {namespace}
```

Add the following content under the **rules** field:

```
rules:
...
- apiGroups:
  - proxy.exporter.k8s.io
  resources:
  - "*"
  verbs: ["get", "list", "watch"]
```

Step 3 Create a file named **kube-apiserver.yaml** and edit it.

```
vi kube-apiserver.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: apiserver
  name: kube-apiserver
  namespace: monitoring    # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    interval: 30s
    metricRelabelings:
    - action: keep
      regex: (aggregator_unavailable_apiservice|
apiserver_admission_controller_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_bucket|
apiserver_admission_webhook_admission_duration_seconds_count|
apiserver_client_certificate_expiration_seconds_bucket|apiserver_client_certificate_expiration_seconds_count|
apiserver_current_inflight_requests|apiserver_request_duration_seconds_bucket|apiserver_request_total|
go_goroutines|kubernetes_build_info|process_cpu_seconds_total|process_resident_memory_bytes|
rest_client_requests_total|workqueue_adds_total|workqueue_depth|
workqueue_queue_duration_seconds_bucket|aggregator_unavailable_apiservice_total|
rest_client_request_duration_seconds_bucket)
      sourceLabels:
      - __name__
    - action: drop
      regex: apiserver_request_duration_seconds_bucket;(0.15|0.25|0.3|0.35|0.4|0.45|0.6|0.7|0.8|0.9|1.25|1.5|1.75|
2.5|3|3.5|4.5|6|7|8|9|15|25|30|50)
      sourceLabels:
```



```
    - __name__
    - le
  port: https
  scheme: https
  tlsConfig:
    caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    serverName: kubernetes
 jobLabel: component
 namespaceSelector:
  matchNames:
  - default
 selector:
  matchLabels:
    component: apiserver
    provider: kubernetes
```

Create a ServiceMonitor:

```
kubectl apply -f kube-apiserver.yaml
```

Step 4 Create a file named **kube-controller.yaml** and edit it.

```
vi kube-controller.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 labels:
   app.kubernetes.io/name: kube-controller
 name: kube-controller-manager
 namespace: monitoring    # Change it to the namespace where Prometheus will be installed.
spec:
 endpoints:
   - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
     interval: 15s
     honorLabels: true
     port: https
     relabelings:
       - regex: (.+)
         replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-controller-proxy/${1}/metrics
         sourceLabels:
           - __address__
         targetLabel: __metrics_path__
       - regex: (.+)
         replacement: ${1}
         sourceLabels:
           - __address__
         targetLabel: instance
       - replacement: kubernetes.default.svc.cluster.local:443
         targetLabel: __address__
     scheme: https
     tlsConfig:
       caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
 jobLabel: app
 namespaceSelector:
   matchNames:
     - kube-system
 selector:
   matchLabels:
     app: kube-controller-proxy
     version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f kube-controller.yaml
```

Step 5 Create a file named **kube-scheduler.yaml** and edit it.

```
vi kube-scheduler.yaml
```


Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-scheduler
  name: kube-scheduler
  namespace: monitoring    # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/kube-scheduler-proxy/${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
            - __address__
          targetLabel: instance
        - replacement: kubernetes.default.svc.cluster.local:443
          targetLabel: __address__
      scheme: https
      tlsConfig:
        caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  jobLabel: app
  namespaceSelector:
    matchNames:
      - kube-system
  selector:
    matchLabels:
      app: kube-scheduler-proxy
      version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f kube-scheduler.yaml
```

Step 6 Create a file named **etcd-server.yaml** and edit it.

```
vi etcd-server.yaml
```

Example file content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: etcd-server
  name: etcd-server
  namespace: monitoring    # Change it to the namespace where Prometheus will be installed.
spec:
  endpoints:
    - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
      interval: 15s
      honorLabels: true
      port: https
      relabelings:
        - regex: (.+)
          replacement: /apis/proxy.exporter.k8s.io/v1beta1/etcd-server-proxy/${1}/metrics
          sourceLabels:
            - __address__
          targetLabel: __metrics_path__
        - regex: (.+)
          replacement: ${1}
          sourceLabels:
```



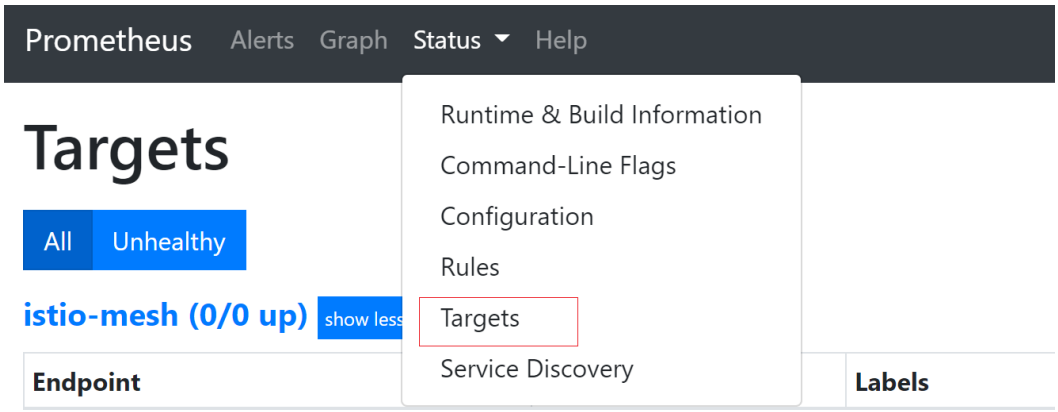
```
        - __address__
        targetLabel: instance
      - replacement: kubernetes.default.svc.cluster.local:443
        targetLabel: __address__
    scheme: https
    tlsConfig:
      caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
jobLabel: app
namespaceSelector:
  matchNames:
    - kube-system
selector:
  matchLabels:
    app: etcd-server-proxy
    version: v1
```

Create a ServiceMonitor:

```
kubectl apply -f etcd-server.yaml
```

Step 7 Access Prometheus and choose **Status** > **Targets**.

The preceding master node components are displayed.



----End

10.9.5 Monitoring Metrics of NGINX Ingress Controller

You can use Prometheus and Grafana to observe the metrics of NGINX Ingress Controller.

The following uses Prometheus as an example to describe how to view the metrics of NGINX Ingress Controller of a cluster.

1. **Accessing Prometheus**

   (Optional) Bind a LoadBalancer Service to Prometheus so that Prometheus can be accessed from external networks.

2. **Monitoring Metrics of NGINX Ingress Controller**

   Enable metric collection for NGINX Ingress Controller so that NGINX Ingress Controller metrics are automatically reported.

Prerequisites

- The cloud native cluster monitoring add-on 3.9.5 or later has been installed in the cluster. For details about this add-on, see **Cloud Native Cluster Monitoring**.

- NGINX Ingress Controller 2.5.4 or later has been installed in the cluster, and metric collection has been enabled. For details about this add-on, see **NGINX Ingress Controller**.

Accessing Prometheus

After **the cloud native cluster monitoring add-on** is installed, you can deploy workloads and Services. The Prometheus server will be deployed as a StatefulSet in the **monitoring** namespace.

You can create a public network **LoadBalancer Service** so that Prometheus can be accessed from external networks.

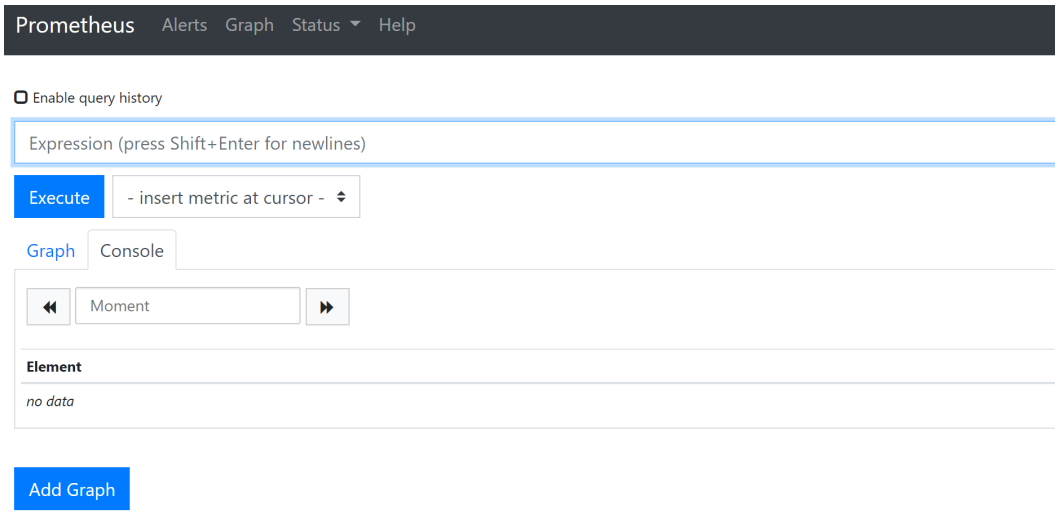
Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

Step 2 Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service.

```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb     # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace it with the ID of the public network load balancer in the VPC that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88            # Service port, which is customizable.
      targetPort: 9090     # Default Prometheus port. Retain the default value.
  selector:               # The label selector can be adjusted based on the label of a Prometheus server instance.
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

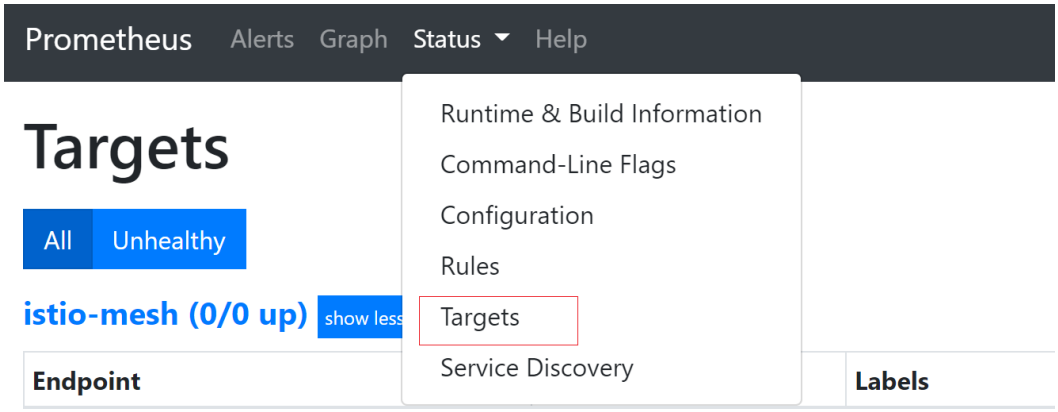
Step 3 After the Service is created, enter *Public IP address of the load balancer*:*Service port* in the address box of the browser to access Prometheus.

Figure 10-118 Accessing Prometheus



Step 4 Choose **Status** > **Targets** to view the targets monitored by Prometheus.

Figure 10-119 Viewing monitored targets



----End

Monitoring Metrics of NGINX Ingress Controller

Log in to Prometheus and click **Graph** to view the metrics of NGINX Ingress Controller.

Figure 10-120 Viewing the metrics of NGINX Ingress Controller

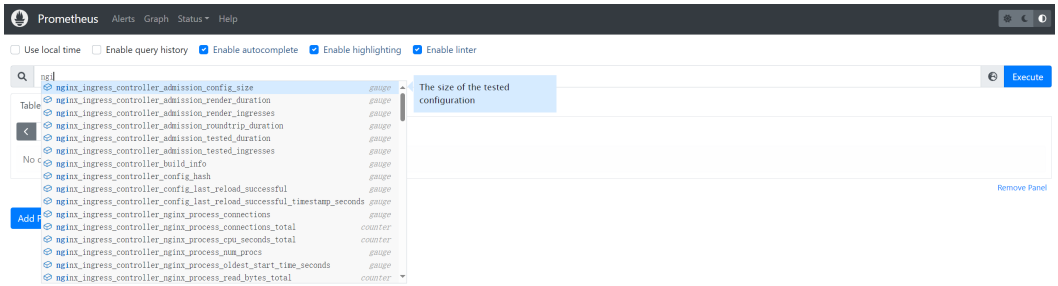


Table 10-56 Metrics of NGINX Ingress Controller

Metric	Type	Description
nginx_ingress_controller_bytes_sent	Basic metric	Number of bytes sent to the client
nginx_ingress_controller_connect_duration_seconds	Basic metric	Duration for connecting to the upstream server
nginx_ingress_controller_header_duration_seconds	Basic metric	Time required for receiving the first header from the upstream server
nginx_ingress_controller_ingress_upstream_latency_seconds	Basic metric	Upstream service latency
nginx_ingress_controller_request_duration_seconds	Basic metric	Time required for processing a request, in milliseconds
nginx_ingress_controller_request_size	Basic metric	Length of a request, including the request line, header, and body
nginx_ingress_controller_requests	Basic metric	Total number of client requests
nginx_ingress_controller_response_duration_seconds	Basic metric	Time required for receiving the response from the upstream server
nginx_ingress_controller_response_size	Basic metric	Length of a response, including the request line, header, and request body
nginx_ingress_controller_nginx_process_connections	Basic metric	Number of client connections in the {active, reading, writing, or waiting} state
nginx_ingress_controller_nginx_process_connections_total	Basic metric	Total number of connections in the {accepted or handled} state
nginx_ingress_controller_nginx_process_cpu_seconds_total	Basic metric	CPU usage, in seconds
nginx_ingress_controller_nginx_process_num_procs	Basic metric	Number of processes
nginx_ingress_controller_nginx_process_oldest_start_time_seconds	Basic metric	Start time, in seconds elapsed since 00:00:00 on January 1, 1970
nginx_ingress_controller_nginx_process_read_bytes_total	Basic metric	Number of bytes read
nginx_ingress_controller_nginx_process_requests_total	Basic metric	Total number of client requests
nginx_ingress_controller_nginx_process_resident_memory_bytes	Basic metric	Number of bytes of resident memory in use

Metric	Type	Description
nginx_ingress_controller_nginx_process_virtual_memory_bytes	Basic metric	Number of bytes of virtual memory in use
nginx_ingress_controller_nginx_process_write_bytes_total	Basic metric	Number of bytes written
nginx_ingress_controller_build_info	Basic metric	A metric with a constant '1' labeled with information about the build
nginx_ingress_controller_check_success	Basic metric	Cumulative count of syntax check operations of NGINX Ingress Controller
nginx_ingress_controller_config_hash	Basic metric	Hash value of running NGINX Ingress Controller
nginx_ingress_controller_config_last_reload_successful	Basic metric	Whether the last configuration reload attempt was successful
nginx_ingress_controller_config_last_reload_successful_timestamp_seconds	Basic metric	Timestamp of the last successful configuration reload
nginx_ingress_controller_ssl_certificate_info	Basic metric	All labels associated with a certificate
nginx_ingress_controller_success	Basic metric	Cumulative count of reload operations of NGINX Ingress Controller
nginx_ingress_controller_orphan_ingress	Basic metric	Status of an orphaned ingress (**1** indicates an orphaned ingress). • **namespace**: character string used to identify the namespace of the ingress • **ingress**: ingress name • **type**: status of the ingress. The value can be **no-service** or **no-endpoint**.
nginx_ingress_controller_admission_config_size	Basic metric	Size of the tested configuration
nginx_ingress_controller_admission_render_duration	Basic metric	Time required for the admission controller to render an ingress
nginx_ingress_controller_admission_render_ingresses	Basic metric	Length of an ingress rendered by the admission controller
nginx_ingress_controller_admission_roundtrip_duration	Basic metric	Complete duration of the admission controller at the time to process a new event (float seconds)

Metric	Type	Description
nginx_ingress_controller_admission_tested_duration	Basic metric	Time required for admission controller tests (float seconds)
nginx_ingress_controller_admission_tested_ingresses	Basic metric	Length of an ingress handled by the admission controller

NOTE

When NGINX Ingress Controller is heavily loaded, memory leakage occurs when full metric collection is enabled. For details, see **the community issue**. It has been verified that the memory usage increase can be effectively suppressed after the following metrics are shielded. To prevent service loss caused by memory leakage, NGINX Ingress Controller shields the following metrics by default. We will continue to pay attention to the latest news in the community and fix this issue in a timely manner.

- nginx_ingress_controller_success
- nginx_ingress_controller_header_duration_seconds
- nginx_ingress_controller_ingress_upstream_latency_seconds

10.9.6 Monitoring Container Network Metrics of CCE Turbo Clusters

CCE Network Metrics Exporter is an add-on for monitoring and managing container network traffic. It collects traffic statistics of containers that do not use the host network in CCE Turbo clusters and performs node-wide container connectivity checks. The monitoring data has been adapted to Prometheus. You can call the Prometheus API to view monitoring data.

The following describes how to view the container network metrics of a CCE Turbo cluster using Prometheus.

1. **Installing the Add-ons**
2. **Monitoring Container Network Metrics**
3. **(Optional) Viewing Graphs on Grafana**

Prerequisites

- A CCE Turbo cluster has been created.
- The cluster has required node resources (at least 4 vCPUs and 8 GiB of memory) for installing the Cloud Native Cluster Monitoring (**Cloud Native Cluster Monitoring**) and CCE Network Metrics Exporter (**CCE Network Metrics Exporter**) add-ons.
- You can access the cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.

Installing the Add-ons

Step 1 Log in to the CCE console and click the CCE Turbo cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.

Step 2 Locate the **Cloud Native Cluster Monitoring** add-on and click **Install**.

When you use this add-on to monitor container network metrics in a CCE Turbo cluster, pay attention to the following parameters. Other parameters of this add-on can be configured as required. For details, see **Cloud Native Cluster Monitoring**.

- **Local Data Storage**: Enable this option. The monitoring data will be stored locally. You can determine whether to connect this add-on to AOM or a third-party monitoring platform.
- **Custom Metric Collection**: Enable this option in this practice. If this option is not enabled, container network metrics cannot be collected.
- (Optional) **Install Grafana**: After installing Grafana, you can view metrics in graphs.

  NOTE

  This parameter is only available for the add-on earlier than v3.9.0. For the add-on of v3.9.0 or later, if Grafana is required, **install Grafana separately**.

Step 3 Locate the CCE Network Metrics Exporter add-on and click **Install**.

No parameter needs to be configured for the current add-on.

Step 4 (Optional) (For Cloud Native Cluster Monitoring v3.9.0 or later, Grafana is not provided by default.) Locate the independent Grafana add-on and click **Install**.

If you enable **Public Access**, a LoadBalancer Service named **grafana-oss** will be created in the **monitoring** namespace. If the load balancer connected to the LoadBalancer Service is bound with an EIP, you can enter *{EIP}:{Port}* in the address box of a browser to access Grafana.

NOTICE

Enabling public access will allow open-source Grafana access over a public network. You are advised to evaluate security risks and create access control policies.

----End

Monitoring Container Network Metrics

Step 1 Add the port information to the DaemonSet configuration of the CCE Network Metrics Exporter add-on.

NOTE

If the add-on version is earlier than 1.3.10, you need to manually add the port information. If the add-on version is 1.3.10 or later, the port information is automatically added so you can skip this step.

```
kubectl edit ds -nkube-system dolphin
```

Add the following content to the file:

```
...
spec:
  containers:
    - name: dolphin
```



```
    ports:
    - containerPort: 10001
      name: dolphin
      protocol: TCP
...
```

Step 2 Configure the **pod-monitor.yaml** file so that Prometheus automatically collects container network metrics.

The following shows an example:

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: dolphin
  namespace: monitoring
spec:
  namespaceSelector:
    matchNames:
    - kube-system
  jobLabel: podmonitor-dolphin
  podMetricsEndpoints:
  - interval: 15s
    path: /metrics
    port: dolphin
    tlsConfig:
      insecureSkipVerify: true
  selector:
    matchLabels:
      app: dolphin
```

Create a PodMonitor.

```
kubectl apply -f pod-monitor.yaml
```

----End

Viewing Metrics on Prometheus

Step 1 Create an example monitoring task. For details, see **Delivering a Monitoring Task**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task          # Monitoring task name.
  namespace: kube-system      # (Mandatory) The value must be kube-system.
spec:
    selector:                 # (Optional) Backend monitored by the add-on. The value has the same format as the labelSelector. By default, all pods on the node are monitored.
    matchLabels:
      app: nginx
    matchExpressions:
      - key: app
        operator: In
        values:
          - nginx
  podLabel: []                # (Optional) Pod label.
  ip4Tx:                      # (Optional) Whether to collect the number of sent IPv4 packets and the number of sent IPv4 bytes. This option is disabled by default.
    enable: true
  ip4Rx:                      # (Optional) Whether to collect the number of received IPv4 packets and the number of received IPv4 bytes. This option is disabled by default.
    enable: true
  ip4TxInternet:              # (Optional) Whether to collect the number of sent IPv4 packets and the number of sent IPv4 bytes. This option is disabled by default.
    enable: true
  healthCheck:                # (Optional) Whether to collect statistics about the latest health check results
```



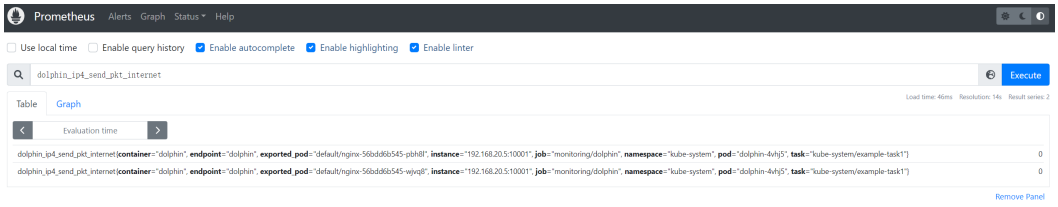
```
and the total numbers of health checks in which pods are considered healthy and of health checks in which
pods are considered unhealthy. This option is disabled by default.
    enable: true              # true false
    failureThreshold: 3       # (Optional) Number of health check failures that consider a pod is unhealthy.
If there is one check failure, the pod is considered unhealthy.
    periodSeconds: 5          # (Optional) Interval between health checks, in seconds. The default value is 60.
    command: ""               # (Optional) Health check command. The value can be ping (default), arping,
or curl.
    ipFamilies: [""]          # (Optional) Health check IP address family. The value is ipv4 by default.
    port: 80                  # (Optional) Port number, which is mandatory when curl is used.
    path: ""                  # (Optional) HTTP API path, which is mandatory when curl is used.
  monitor:
    ip:
      ipReceive:
        aggregateType: flow       # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
      ipSend:
        aggregateType: flow       # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcp:
      tcpReceive:
        aggregateType: flow       # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
      tcpSend:
        aggregateType: flow       # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
      tcpRetrans:
        aggregateType: flow       # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
      tcpRtt:
        aggregateType: flow       # (Optional) The value can be flow (monitored by flow). The unit is μs.
      tcpNewConnection:
        aggregateType: pod        # (Optional) The value can be pod (monitored by pod).
```

Step 2 Create a public network LoadBalancer Service so that Prometheus can be accessed from external networks.

```
apiVersion: v1
kind: Service
metadata:
  name: prom-lb     # Service name, which is customizable.
  namespace: monitoring
  labels:
    app: prometheus
    component: server
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace it with the ID of the public network load balancer in the VPC
that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 88            # Service port, which is customizable.
      targetPort: 9090    # Default Prometheus port. Retain the default value.
  selector:               # The label selector can be adjusted based on the label of a Prometheus server
instance.
    app.kubernetes.io/name: prometheus
    prometheus: server
  type: LoadBalancer
```

Step 3 After the Service is created, enter *Public IP address of the load balancer*:*Service port* in the address box of the browser to access Prometheus. You can search for **supported monitoring items** on Prometheus to check whether the metrics are successfully collected.

Figure 10-121 Accessing Prometheus



----End

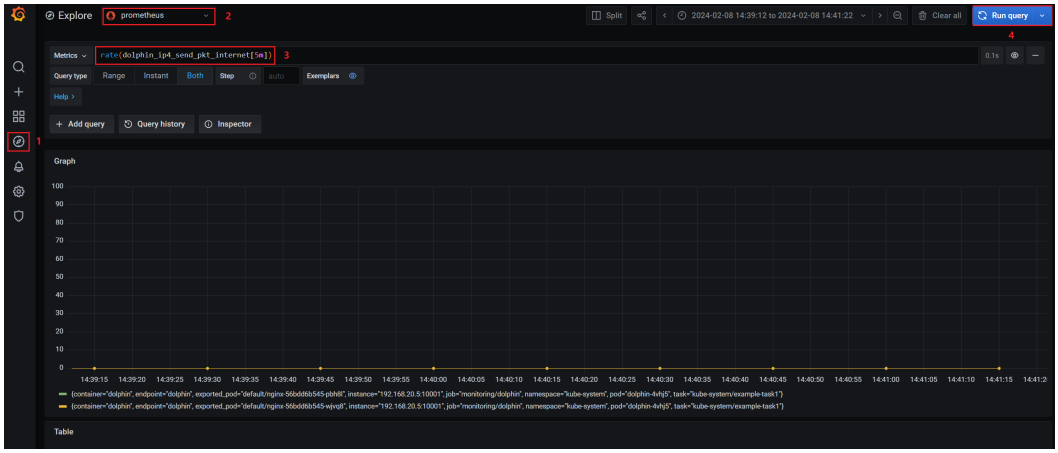
(Optional) Viewing Graphs on Grafana

Step 1 After installing Grafana in the cluster, locate the Grafana add-on on the **Add-ons** page and click **Access**.

Step 2 Enter your Grafana login account and password.

Step 3 In the navigation pane, click **Explore**. Select **prometheus** and enter the PromQL query command, for example, **rate(dolphin_ip4_send_pkt_internet[5m])**. In the upper right corner, click **Run query** to obtain the metric graph.

Figure 10-122 Grafana graph



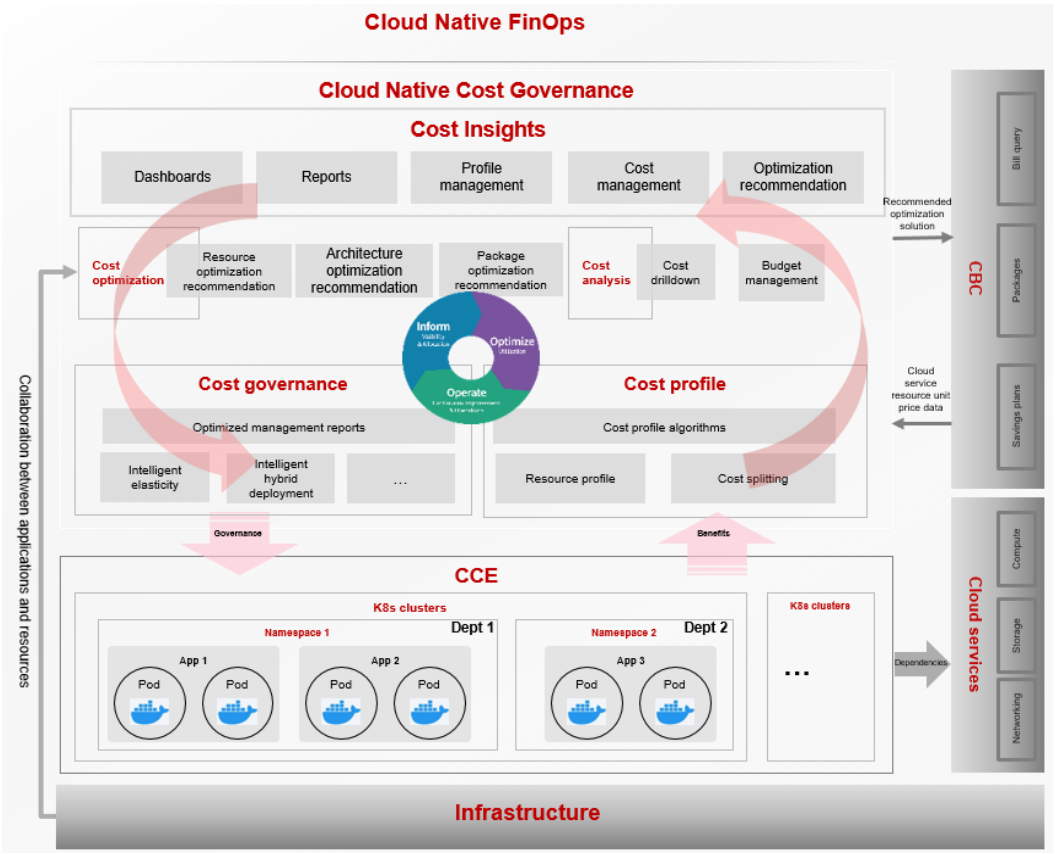
Step 4 (Optional) Use common graphs as Grafana dashboards. For details, see **Create a dashboard**.

----End

11 Cloud Native Cost Governance

11.1 Overview

Cloud native cost governance is a FinOps-based solution for managing container costs. It provides cost and resource profiles by department, cluster, or namespace. It also leverages optimization techniques like resource specifications recommendation to help IT cost management personnel improve resource utilization and reduce costs of container clusters.



Cost Insights

Cost Insights uses in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on your bills and cluster resource usages. Cost Insights helps cost management personnel analyze cluster costs and resource usages and identify resource waste for cost optimization.

11.2 Agency Permissions

Cloud Native Cost Governance works closely with cloud services such as CCE, AOM, OBS, and CBC for cost governance. When you use Cloud Native Cost Governance for the first time, you need to use an account with the Security Administrator permissions to grant the cloud resource permissions of the current region to CCE.

To minimize authorization, CCE fine-grained permissions are optimized. The permissions defined by system policies are now defined using API actions. (Each API has one API action.) If you have authorized the cloud services, you can optimize the permissions in one click.

After you agree to the authorization, agencies are automatically created in IAM to delegate required resource operation permissions in your account to Huawei Cloud CCE and AOM. For details about agencies, see **Cloud Service Delegation**. The following are agencies automatically created in IAM:

- cia_admin_trust

  This agency is used to to delegate the permissions required by the O&M modules to access other cloud services.

  To use the O&M modules in multiple regions, you need to apply for the Tenant Guest, CCE Administrator, and SWR Administrator permissions in each region. (Go to the IAM console, choose **Agencies**, and click **cia_admin_trust** to view the authorization records in each region.)

- aom_admin_trust

  For details about this agency, see **AOM Cloud Service Authorization**.

NOTE

The O&M modules may fail to run as expected if the required permissions are not granted. When using the O&M modules, do not delete or modify **cia_admin_trust** and **aom_admin_trust**.

Permissions Before Optimization

Table 11-1 cia_admin_trust permissions

Granted To	Policy/Role	Description
CCE	IAM ReadOnlyAccess	IAM users need to be able to access Monitoring Center and Alarm Center.

Granted To	Policy/Role	Description
CCE	Tenant Guest	Monitoring Center and Alarm Center check the configurations of global resources (for example OBS or DNS) associated with clusters to identify invalid or inappropriate configurations.
CCE	CCE Administrator	Monitoring Center and Alarm Center need to be able to access CCE to obtain information about clusters, nodes, workloads, and other resources, so that they can help ensure resource health.
CCE	SWR Administrator	Monitoring Center and Alarm Center need to be able to access SWR to obtain image information.
CCE	SMN Administrator	Monitoring Center and Alarm Center need to be able to access SMN to obtain contact groups.
CCE	AOM Administrator	Monitoring Center and Alarm Center need to be able to access AOM to obtain metrics.
CCE	LTS Administrator	Monitoring Center and Alarm Center need to be able to access LTS to obtain logs.

Table 11-2 aom_admin_trust permissions

Granted To	Policy/Role	Description
AOM	DMS UserAccess	AOM obtains subscription data from DMS.
AOM	ECS CommonOperations	AOM obtains system metrics and logs using UniAgents and ICAgents installed on ECSs.
AOM	CES ReadOnlyAccess	AOM synchronizes metrics from Cloud Eye.
AOM	CCE FullAccess	AOM synchronizes container metrics from CCE.
AOM	RMS ReadOnlyAccess	AOM CMDB manages cloud service instances.
AOM	ECS ReadOnlyAccess	AOM obtains system metrics and logs using UniAgents and ICAgents installed on ECSs.
AOM	LTS FullAccess	AOM obtains logs from LTS.
AOM	CCI FullAccess	AOM synchronizes container metrics from CCI.

Permissions After Optimization

Table 11-3 cia_admin_trust permissions

Policy Name	Policy Type	Policy Scope	Permission Set	Description
CCE Administrator	System-defined policy	Project	cce:*:*	CCE administrator permissions
CIACostGlobalPolicy	Custom policy	Global	obs:object:GetObject	Obtains object content and metadata.
			obs:bucket:HeadBucket	Obtains bucket metadata.
			obs:bucket:CreateBucket	Creates a bucket.
			obs:bucket:ListBucket	Lists objects in a bucket.
			OBS:*:*:object:cost/daily_cost_{region_id}	Resource object path restriction
			OBS:*:*:bucket:cce-cost-{region_id}-{domain_id}	Resource bucket limit
CIACostProjectPolicy	Custom policy	Project	cce:cluster:get	Obtains details about a cluster.
			cce:cluster:list	Lists all clusters.
			cce:addonInstance:list	Lists all add-on instances.
			cce:addonInstance:create	Creates an add-on instance.
			cce:addonInstance:delete	Deletes an add-on instance.
			cce:addonInstance:update	Updates an add-on instance.
			cce:node:get	Obtains details about a node.
			cce:node:list	Lists nodes.
			cce:nodepool:list	Lists all node pools in a cluster.

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			aom:metric:set	Modifies monitoring configuration.
			aom:metric:get	Queries details about a metric.
			aom:metric:list	Lists metrics.
			aom:agency:get	Queries AOM authorization.
			bss:costtag:update	Activates or deactivates cost tags.
			bss:costtag:view	Views cost tags.
			bss:costdetailreport:view	Views the task list for exporting cost details to OBS.
			bss:costdetailreport:update	Creates, modifies, or deletes the tasks of exporting cost details to OBS.
			apm:icmgr:get	Obtains AOM 2.0 permissions.
			apm:icmgr:create	Grants AOM 2.0 permissions.

Table 11-4 aom_admin_trust permissions

Policy Name	Policy Type	Policy Scope	Permission Set	Description
AOM Global Access	Custom policy	Global	rms:*:list	Lists RMS resources.
			rms:*:get	Queries details about an RMS resource.
			rms:resources:listTagsForResource	Lists resource tags.
			rms:resources:listTags	Lists project tags.
			rms:resources:listResourcesByTag	List resource instances.

Policy Name	Policy Type	Policy Scope	Permission Set	Description
AOM UserAccess	Custom policy	Project	lts:topics:*	Full permissions for performing operations on log topics
			lts:groups:*	Full permissions for performing operations on log groups
			aom:metric:*	Full permissions for performing operations on a metric (AOM)
			aom:cmdbSubApplication:*	Full permissions for performing operations on a sub-application (AOM)
			aom:cmdbResources:*	Full permissions for performing operations on resources (AOM)
			aom:cmdbEnvironment:*	Full permissions for performing operations on the environment (AOM)
			aom:cmdbComponent:*	Full permissions for performing operations on a component (AOM)
			aom:cmdbApplication:*	Full permissions for performing operations on an application (AOM)
			ecs:cloudServers:showServer	Queries details about an ECS.
			ecs:cloudServers:list	Lists ECSs.
			dms:instance:get	Queries details about a DMS instance.
			ces:metrics:list	Lists metrics (Cloud Eye).

Policy Name	Policy Type	Policy Scope	Permission Set	Description
			ces:metricData:list	Queries metrics (Cloud Eye).
			cci:namespace:list	Lists all namespaces.
			cce:cluster:list	Lists all clusters.
			cce:cluster:get	Obtains details about a cluster.
			cce:node:list	List nodes.
			cce:node:get	Obtains details about a node.
			apm:icmgr:*	Full permissions for performing operations on the APM collection component
			lts:*:*	Full permissions for performing operations on LTS logs
			aom:*:list	Lists AOM instances.

11.3 Cost Insights

11.3.1 Overview

Cost Insights uses in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on your bills and cluster resource usages. Cost Insights helps cost management personnel analyze cluster costs and resource usages and identify resource waste for cost optimization.

Cost Insights displays container costs by region or cluster.

- Cost Insights for a region provides enterprise management personnel with an overall analysis report of container costs in that region. Enterprise management personnel can divide departments by cluster or namespace and obtain the cost analysis report of each department. By checking reports, enterprise management personnel can identify cost increase changes and compare department costs to design better cost management solutions.
- Cost Insights for a cluster helps cost O&M personnel analyze cluster costs and resource usages from multiple dimensions, such as namespace, application, and node pool, to identify applications that can be optimized.

Key Capabilities

- Diverse cost types: Both the management costs of CCE clusters and the costs of ECS and EVS resources associated with each CCE cluster are analyzed and displayed.
- Precise cost calculation: Your bills are used to collect statistics on cluster costs for accurate cost allocation.
- Flexible cost allocation policies: Costs are displayed and allocated by multiple dimensions, such as cluster, namespace, node pool, and application.
- Long-term cost data storage and retrieval: Costs of up to two years can be analyzed.
- Per-minute workload billing: Workload discovery in minutes and per-minute billing are available for rapid scaling of applications. All costs will be billed.

Constraints

- Currently, only EVS disk storage costs are collected.
- Node costs can be split by CPU and memory. However, if heterogeneous resources are used, such as GPUs and NPUs, the costs cannot be split. For example, when the costs of a GPU node are split, the core-hour unit price is higher.
- After Cost Insights is enabled, analysis results can be displayed two days later.
- Cost Insights displays cost analysis results by day.

11.3.2 Cost Calculation Model

Workload Cost Calculation Principle

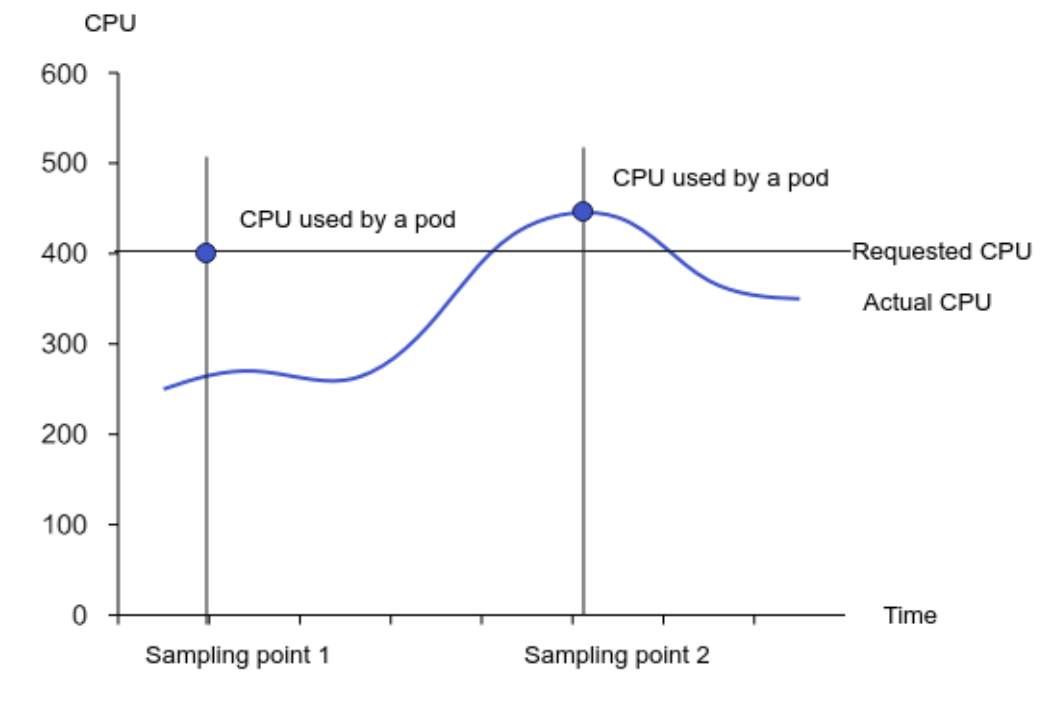
The costs of a workload are the total costs of all pods for running that workload.

- Pod costs are calculated based on the percentage of total node resources (CPU and memory) that are used by a pod, plus the cost of PVC storage used by that pod.

$$\text{Pod costs} = \frac{\max(\text{CPU requested by a pod}, \text{CPU used by that pod})}{\text{Node CPU capacity}} \times \text{Node cost} \times \text{CPU usage} + \frac{\max(\text{Memory requested by that pod}, \text{Memory used by that pod})}{\text{Node memory capacity}} \times \text{Node cost} \times \text{Memory usage} + \text{Cost of PVC storage used by that pod}$$

  The requested CPU and the actual CPU are compared, and the greater value is used to calculate the costs.

Figure 11-1 Workload cost calculation principle



For example, at sampling point 1, the value of requested CPU is greater than that of actual CPU, the requested CPU is used for calculation.

At sampling point is 2, the value of requested CPU is less than that of actual CPU, the actual CPU is used for calculation.

- The costs of a workload are the total costs of all pods for running that workload.

$$\text{Workload costs} = \sum_{\text{Workload}}^{all} \text{Pod costs}$$

Namespace Cost Calculation Principle

The costs of a namespace are the total costs of all workloads in that namespace.

$$\text{Namespace costs} = \sum_{\text{Namespace}}^{all} \text{Workload costs}$$

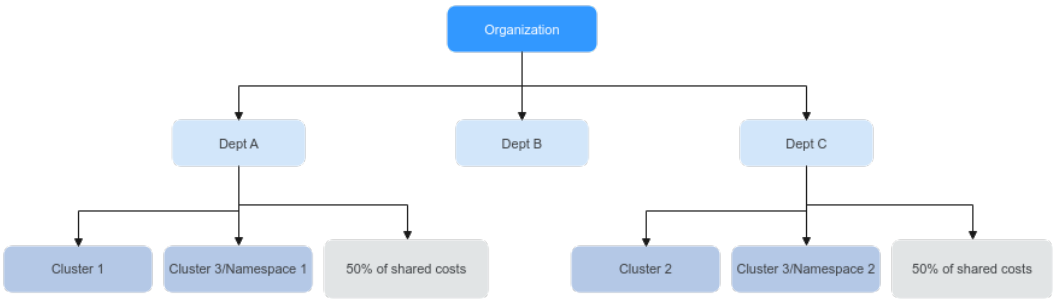
Department Cost Calculation Principle

Departments are logical cost units used to aggregate and analyze the costs of different clusters and namespaces. Generally, actual business departments are used and associated with the clusters or namespaces they used.

The costs of a single cluster consist of the namespace cost, unallocated cost, and cluster management cost (the cost for managing master nodes in a CCE cluster and the system namespace). The unallocated cost and cluster management cost

are defined as shared costs. If enterprises organized their department costs based on namespaces, the departments need to be associated with the namespaces, and the proportion of shared costs needs to be specified.

Figure 11-2 Department cost calculation example



Cluster 1 is a dedicated cluster of department A, cluster 2 is a dedicated cluster of department C, and cluster 3 is shared by departments A and C. Namespace 1 belongs to department A, and namespace 2 belongs to department C. When calculating the costs of each department, you can allocate the costs by namespace or cluster, specify the proportion of the shared costs generated in cluster 3 for department A and department C.

Common Issues During Calculation

- Are the costs of a pod calculated from the requested resources or actually-used resources? Since resource metric values like used CPU and memory change dynamically, how can the costs of a pod be estimated accurately?

  The requested CPU and the actual CPU are compared, and the greater value is used to calculate the cost. As long as an application runs for more than 1 minute, its metrics can be collected by the prometheus add-on for cost calculation.

- How are unallocated costs of a node handled?

  Unallocated costs of a node are not allocated as the workload or namespace costs, but can be shared among departments. You can specify the proportion of unallocated costs allocated to each department.

11.3.3 Enabling Cost Insights

Cost Insights uses in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on your bills and cluster resource usages. Cost Insights helps cost management personnel analyze cluster costs and resource usages and identify resource waste for cost optimization.

This section describes how to enable Cost Insights.

- **Enabling Cost Insights for a Region**
- **Enabling Cost Insights for a Cluster**

NOTICE

- After Cost Insights is enabled, Cloud Native Cluster Monitoring will be installed automatically, and metrics will be reported to the selected AOM instance. Basic metrics are free, but custom metrics are charged by AOM. For details, see **Pricing Details**. Cost Insights only uses basic metrics.
- After Cost Insights is enabled, an OBS bucket will be created in AP-Singapore to store your bills subscribed from the billing center. Then, you will be billed for the OBS storage. For details, see **Pricing Details**.

Prerequisites

You have an account in the **admin** user group to delegate CCE and its dependent services.

When you access Cloud Native Cost Governance, the **Authorize** dialog box is automatically displayed. After you click **Authorize**, CCE automatically completes the authorization. For details about permission types, see **Table 11-5**.

Table 11-5 Resource permissions

Assigned To	Permission	Description
CCE	IAM ReadOnlyAccess	IAM users need to access Cloud Native Cost Governance.
CCE	Tenant Guest	Cloud Native Cost Governance checks the configurations of global resources (such as OBS and DNS) associated with a cluster.
CCE	CCE Administrator	Cloud Native Cost Governance needs to access CCE to obtain information about clusters, nodes, and workloads for health checks.
CCE	AOM Administrator	Cloud Native Cost Governance needs to access AOM to obtain metrics.
CCE	OBS Administrator	Cloud Native Cost Governance needs to access the OBS bucket where your bills are stored.
CCE	CBC Finance	Cloud Native Cost Governance needs to periodically store your bills in the OBS bucket for later use.
AOM	DMS UserAccess	AOM retrieves data from DMS.
AOM	ECS CommonOperations	AOM obtains system metrics and logs from the UniAgent and ICAgent installed on an ECS.
AOM	CES ReadOnlyAccess	AOM synchronizes metrics from Cloud Eye.

Assigned To	Permission	Description
AOM	CCE FullAccess	AOM synchronizes container metrics from CCE.
AOM	RMS ReadOnlyAccess	AOM CMDB manages cloud service instance data.
AOM	ECS ReadOnlyAccess	AOM obtains system metrics and logs from the UniAgent and ICAgent installed on an ECS.
AOM	LTS FullAccess	AOM obtains logs from LTS.
AOM	CCI FullAccess	AOM synchronizes container metrics from CCI.

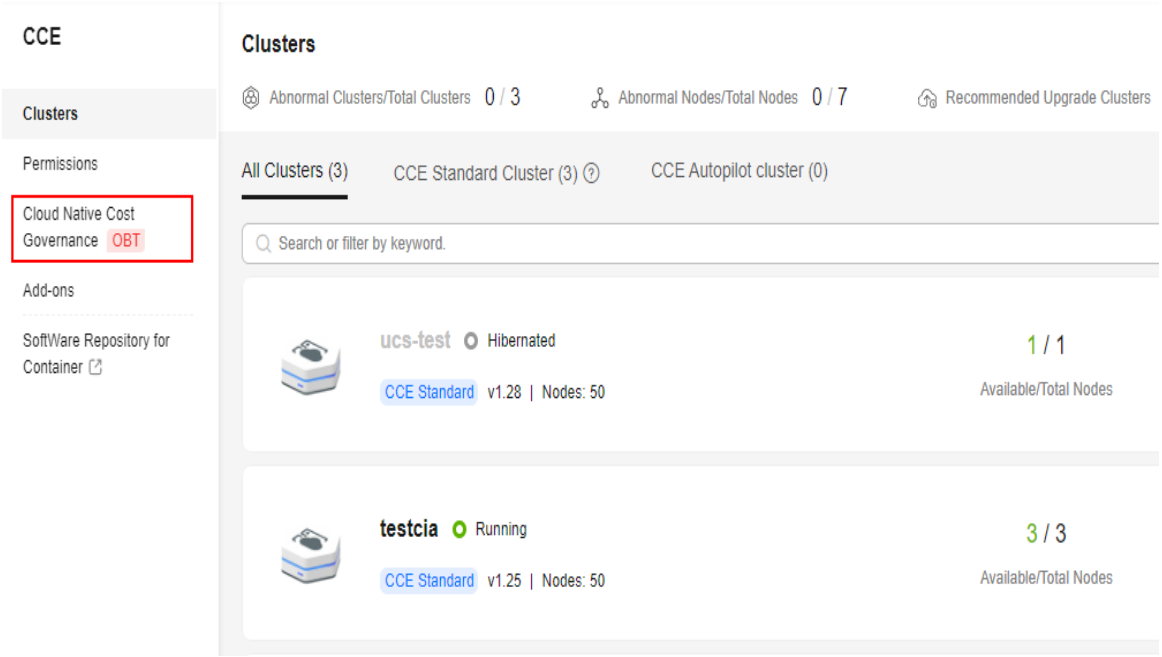
Constraints

- Cloud Native Cost Governance is only available in clusters v1.17 or later.
- Before using Cloud Native Cost Governance, you need to use an account in the **admin** user group to delegate CCE and its dependent services. After the authorization is complete, users with the CCE Administrator role or CCE FullAccess permission can perform all operations on Cloud Native Cost Governance.

Enabling Cost Insights for a Region

Step 1 Log in to the CCE console. In the navigation pane, choose **Cloud Native Cost Governance**.

Figure 11-3 Cloud Native Cost Governance

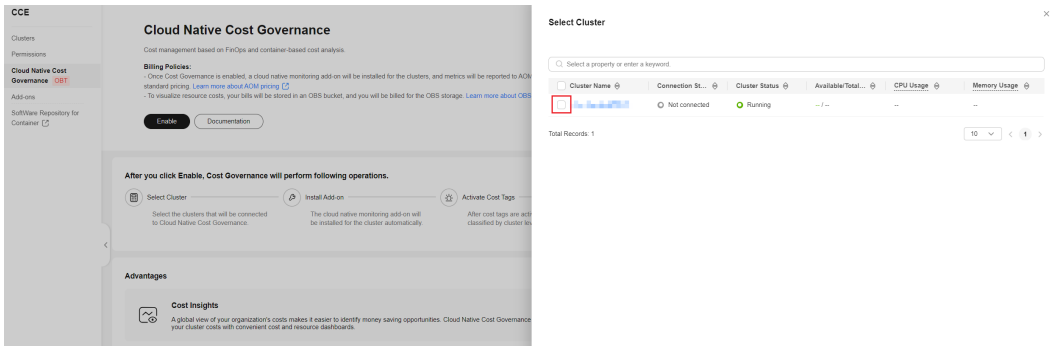


Step 2 Click **Enable**, select clusters that require Cost Insights, and click **Connect Now**.

The system will automatically perform the following operations. After 3 to 5 minutes, the **Cost Insights** page is displayed.

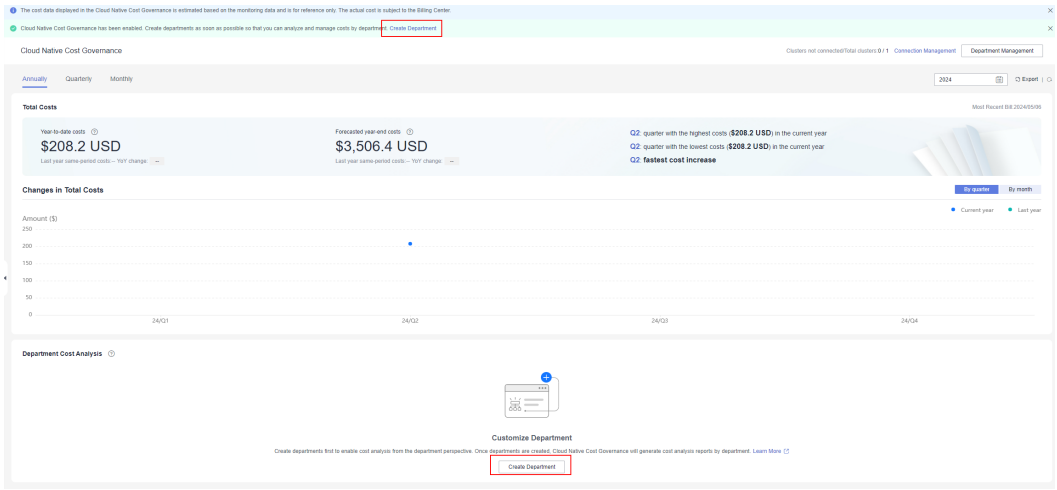
1. Install Cloud Native Cluster Monitoring to provide basic metrics for Cost Insights.
2. Activate cost tags. Cluster tags will be added to the bills exported from the billing center. The Cost Insights background classifies the bills by cluster. After this step is complete, the **CCE-Cluster-ID** and **CCE-Dynamic-Provisioning-Node** tags are activated on the cost tag page of Cloud Native Cost Governance.
3. Create an OBS bucket named **cce-cost-***{region}*-*{domain_id}* for the default tenant to store bills exported from the billing center.
4. Subscribe to bills. After bills are subscribed, the billing center will periodically push bills to the OBS bucket for cost insights.

Figure 11-4 Enabling Cost Insights for clusters



Step 3 (Optional) Click **Create Department**.

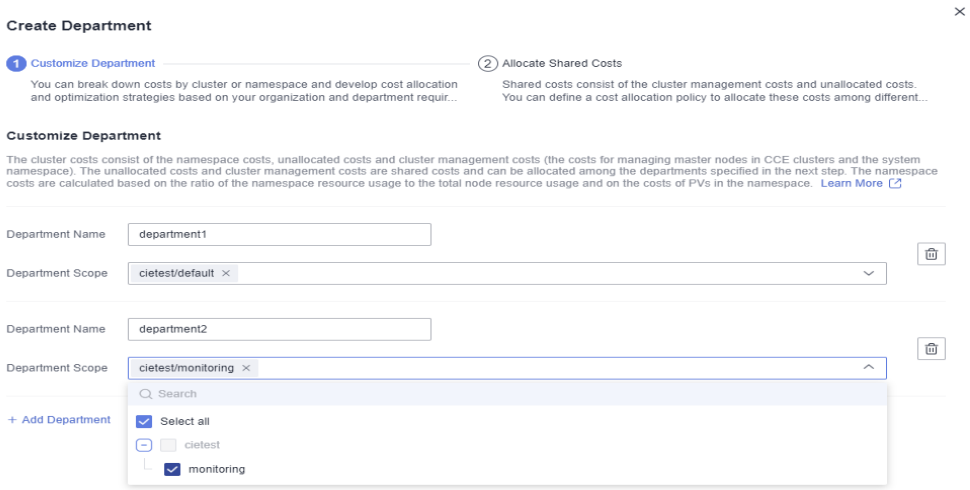
Figure 11-5 Creating departments



1. Customize departments based on the service requirements and associate them with clusters or namespaces as follows:
   - **Department Name**: Use the actual name of a business department.

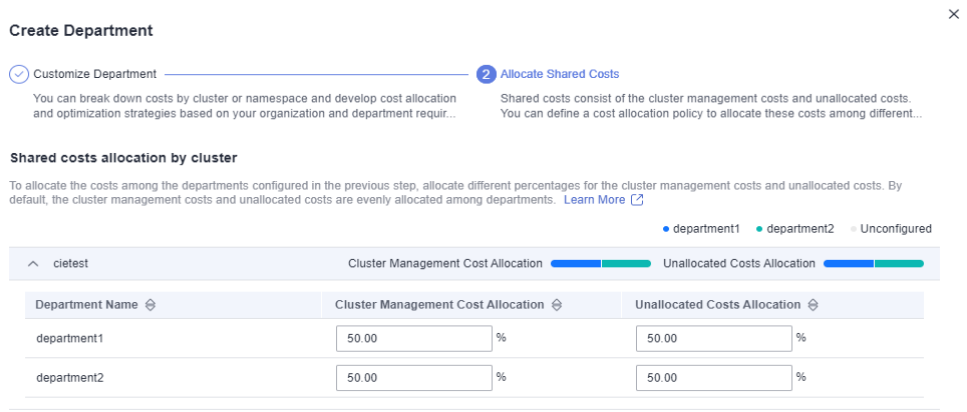
- **Department Scope**: Select the cluster or namespace for the business department. In the following example, **department1** and **department2** are configured by namespace.

Figure 11-6 Customizing departments



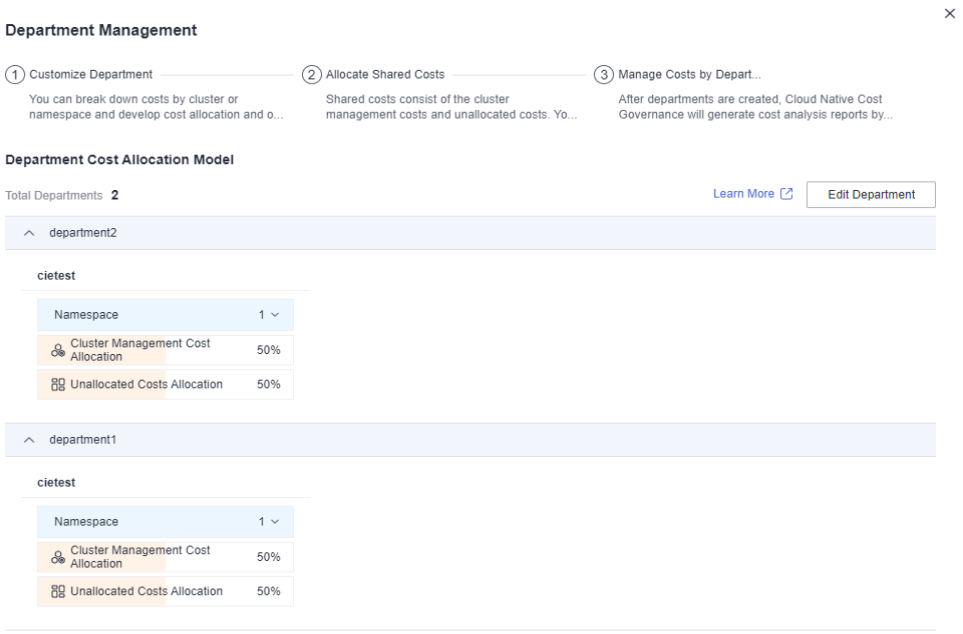
2. Allocate shared costs in a cluster to departments. Management costs and unallocated costs in the default cluster are evenly allocated among departments. The allocation ratio can be changed.

Figure 11-7 Allocating shared costs



3. Manage costs by department. After departments are created, click **Submit**. The configurations are displayed on the department management page. The following shows example department configurations.

Figure 11-8 Department configurations



----End

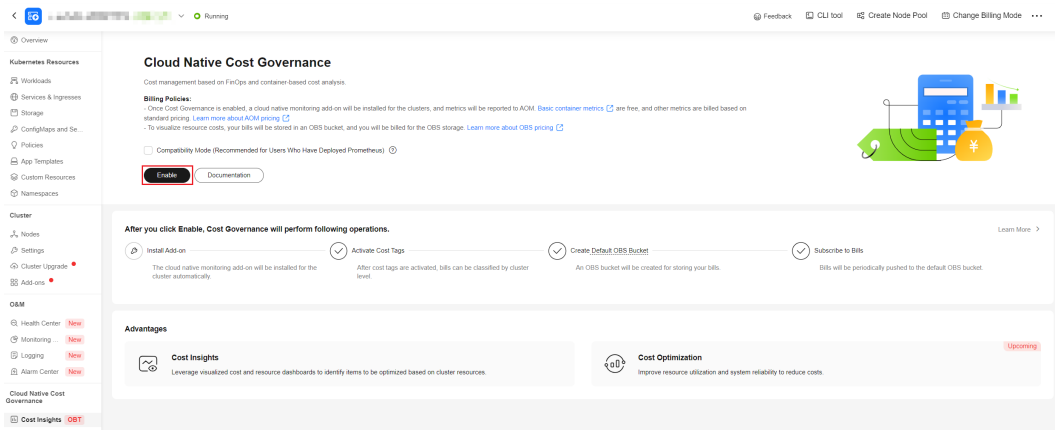
Enabling Cost Insights for a Cluster

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Cloud Native Cost Governance** > **Cost Insights**.

Step 3 Click **Enable**. The system automatically installs Cloud Native Cluster Monitoring, activates cost tags, creates an OBS bucket for the default tenant, and subscribes to bills. Wait for 3 to 5 minutes. The **Cost Insights** page is displayed.

Figure 11-9 Enabling Cost Insights



----End

11.3.4 Cost Insights for a Region

Cost Insights for a region provides enterprise management personnel with an overall analysis report of container costs in that region. From the perspective of cloud native, enterprise management personnel can flexibly organize costs. Costs can be allocated to departments by cluster or namespace to generate department cost analysis reports. In addition, cost reports can be exported.

Prerequisites

- Cost Insights has been enabled.

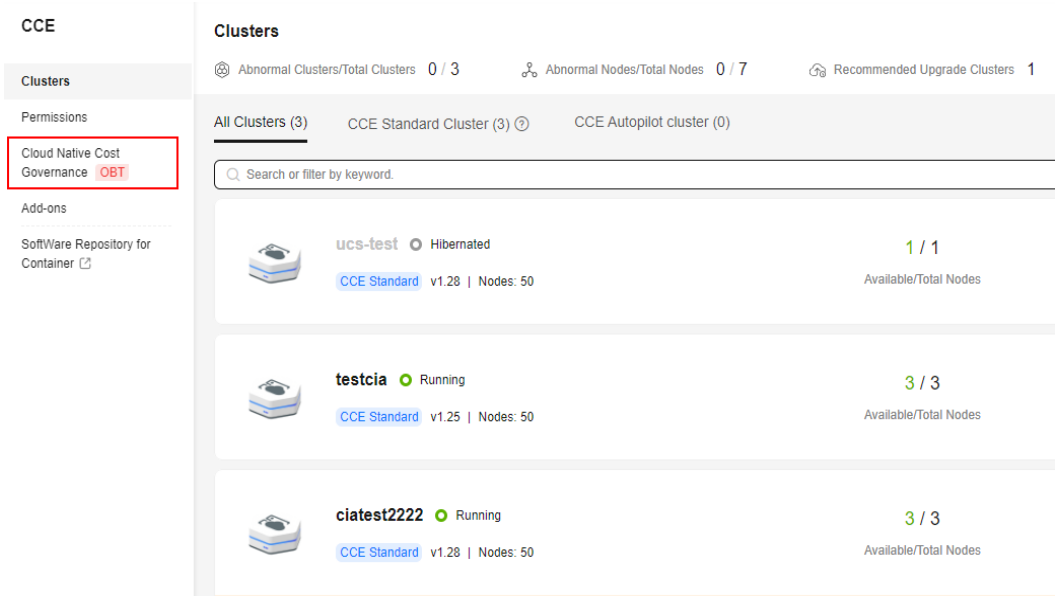
Constraints

- Processing bills takes some time. After Cost Insights is enabled, there is about two days delay before you can view your costs.
- Cloud Native Cluster Monitoring must run normally to ensure accurate data displays of namespaces, workloads, and node pools on the **Cost Insights** page.

Access Management

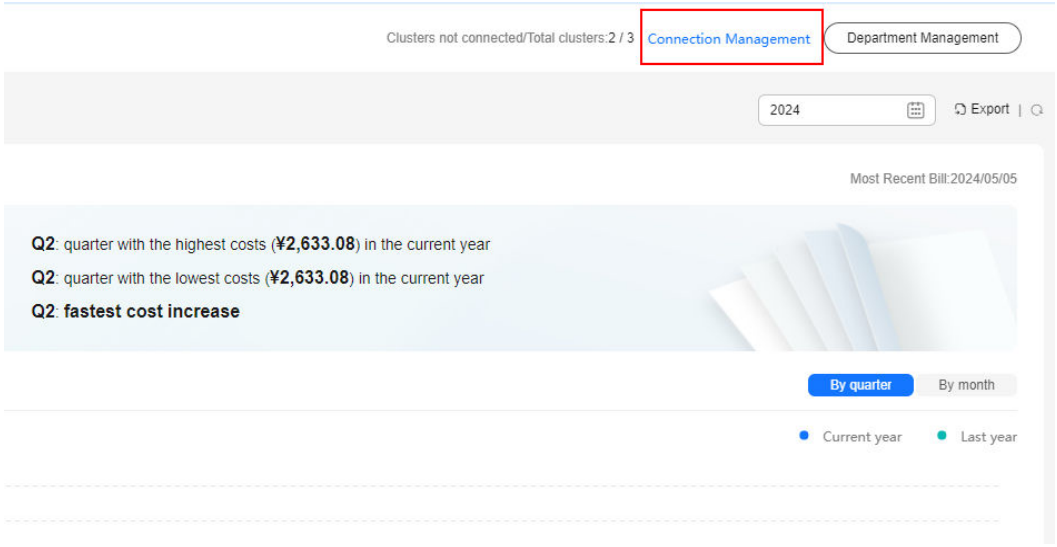
Step 1 Log in to the CCE console. In the navigation pane, choose **Cloud Native Cost Governance**.

Figure 11-10 Cloud Native Cost Governance



Step 2 Click **Connection Management** to view the cluster connection status.

Figure 11-11 Cluster connection



Step 3 Select the clusters that are not connected to Cloud Native Cost Governance and click **Batch Connect**. After Cloud Native Cost Governance is enabled, you can view the cluster connection status in the list. If your cluster is connected to Cloud Native Cost Governance for the first time, you need to wait for two days before viewing your costs.

Figure 11-12 Batch connection

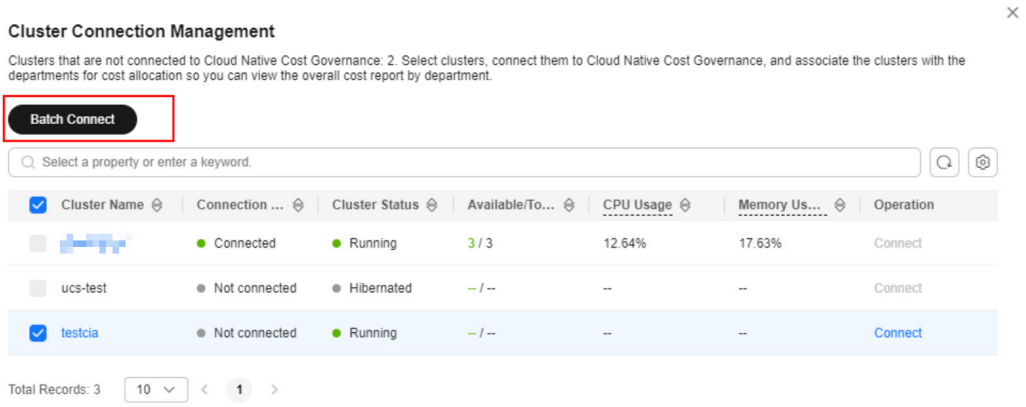
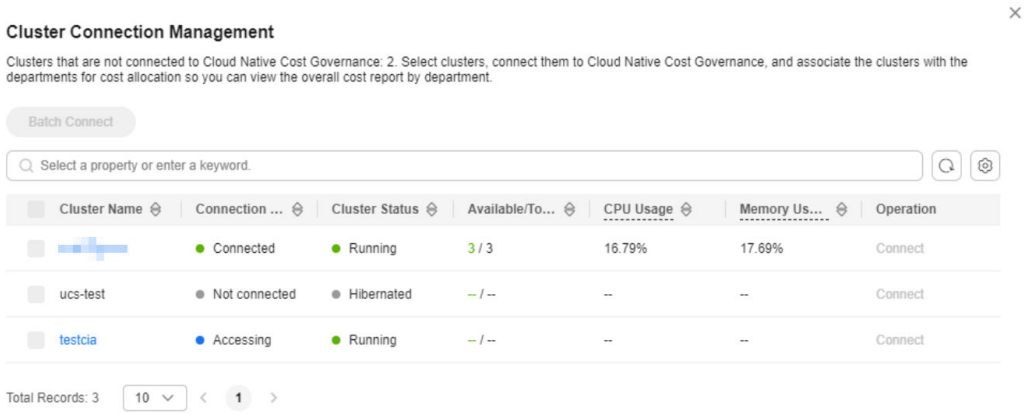


Figure 11-13 Cluster connection management

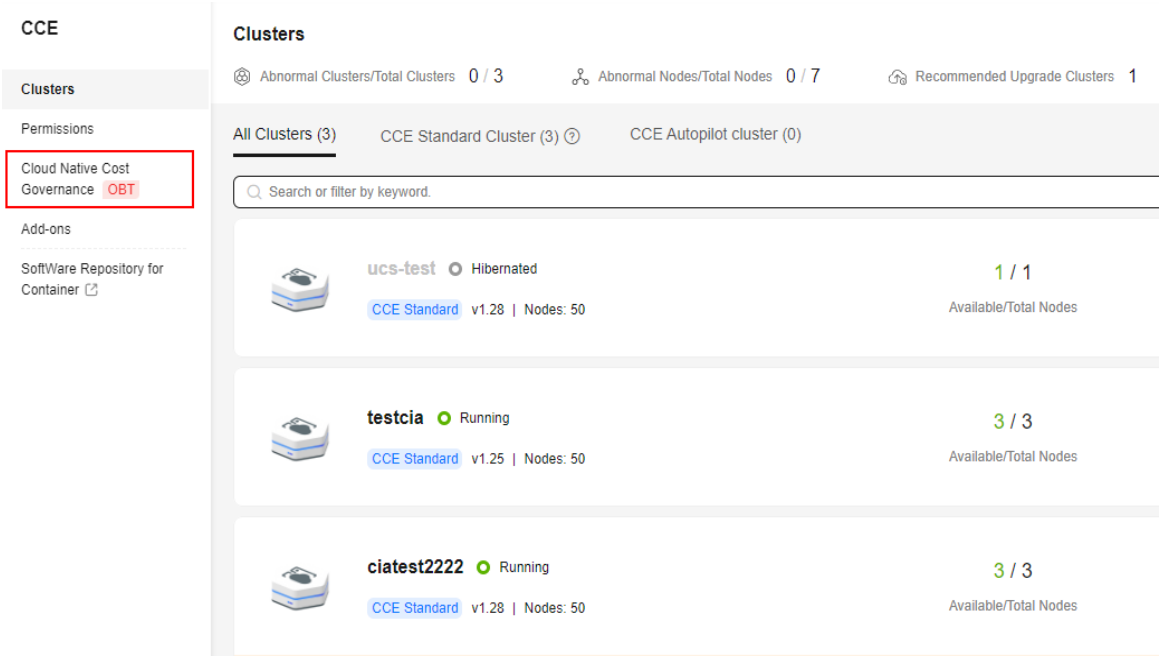


----End

Department Management

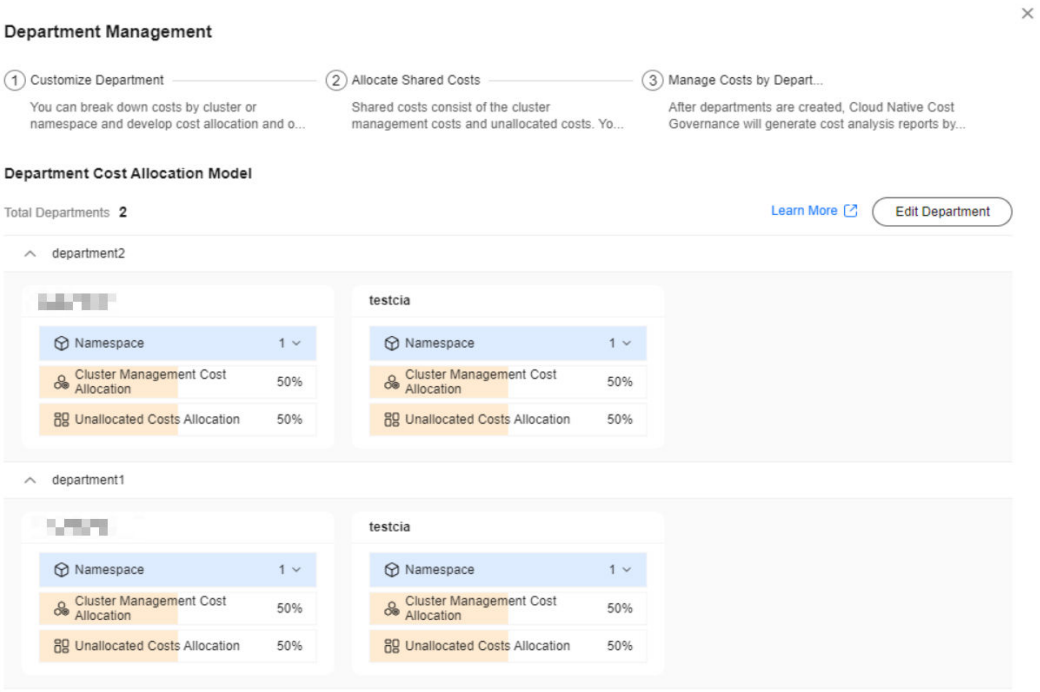
Step 1 Log in to the CCE console. In the navigation pane, choose **Cloud Native Cost Governance**.

Figure 11-14 Cloud Native Cost Governance



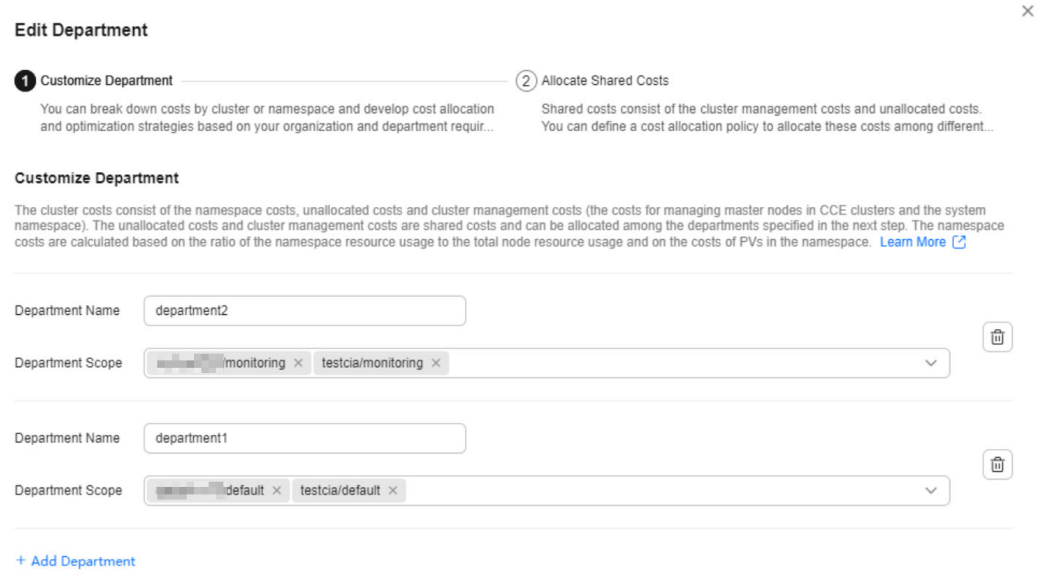
Step 2 Click **Department Management** to view the department configurations.

Figure 11-15 Viewing the department configurations



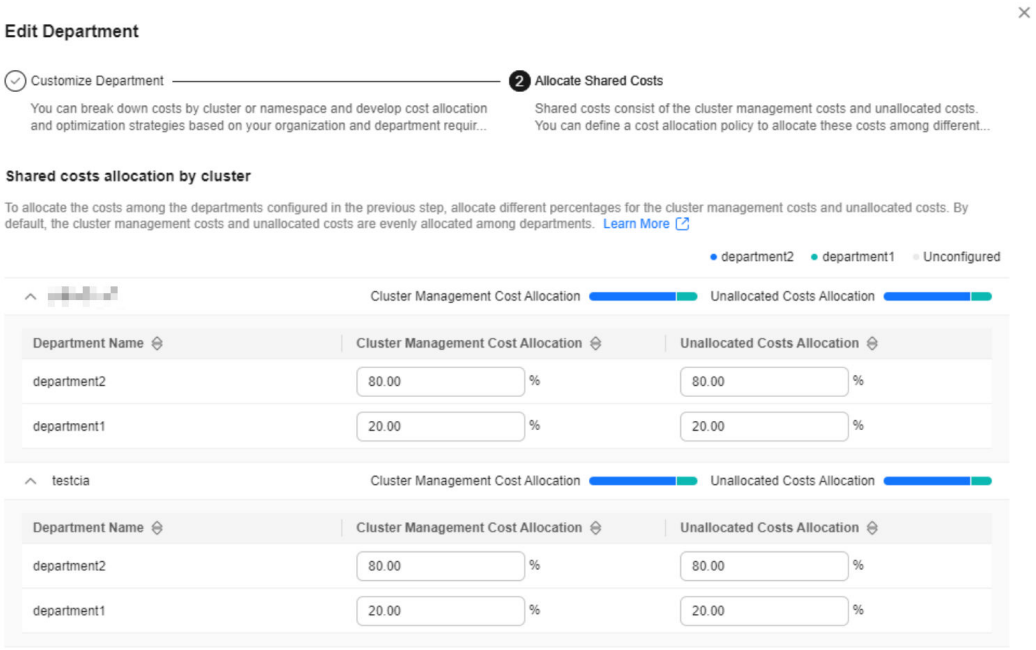
Step 3 Click **Edit Department** to modify the custom department configurations.

Figure 11-16 Modifying the department configurations



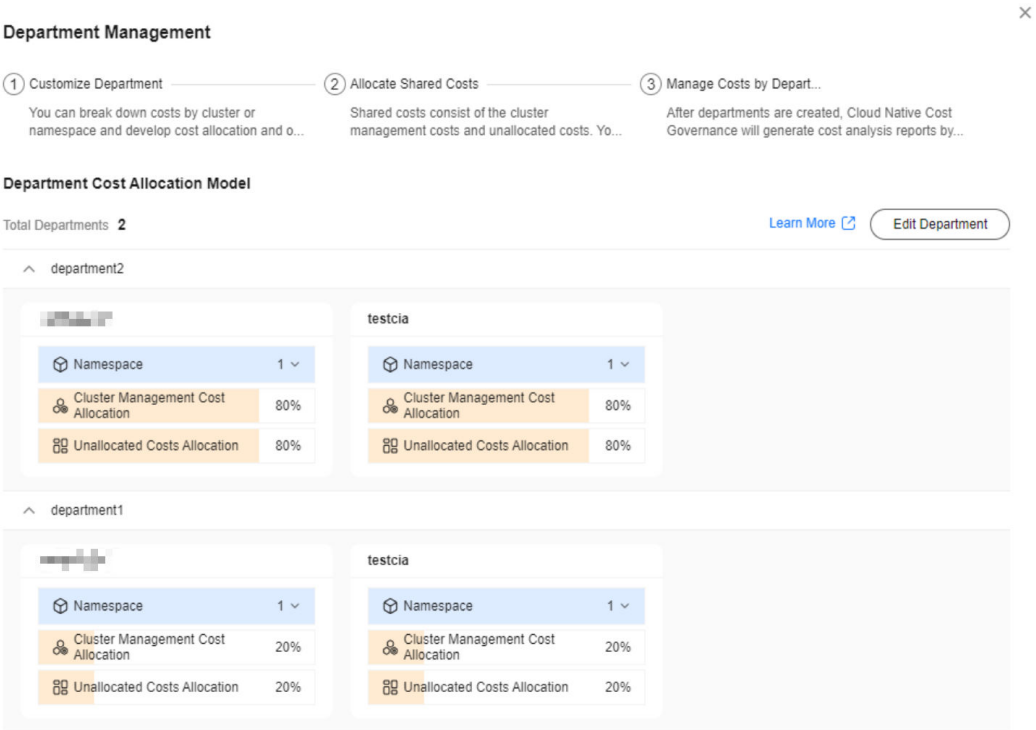
Step 4 Click **Next** to modify the percentages of shared costs allocated to departments.

Figure 11-17 Modifying shared costs allocation



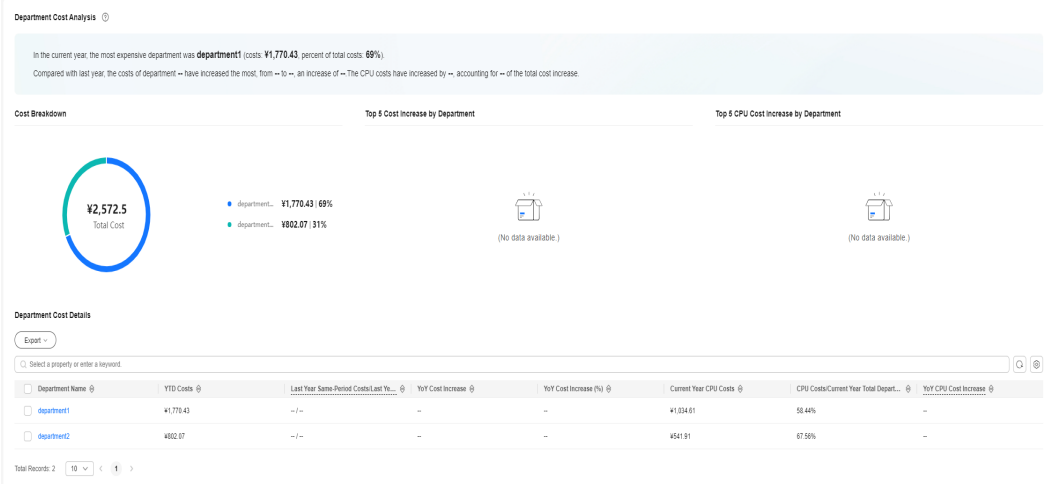
Step 5 Click **Submit**. The configurations are displayed on the department management page.

Figure 11-18 Submitting the modification



Step 6 Close the department management page and view cost analysis reports in **Department Cost Analysis**.

Figure 11-19 Viewing cost analysis reports



----End

Using Cost Insights

After Cloud Native Cost Governance is enabled, go to the **Cost Insights** page to view the annual, quarterly, or monthly cost analysis reports of connected clusters.

Figure 11-20 Viewing the cost status

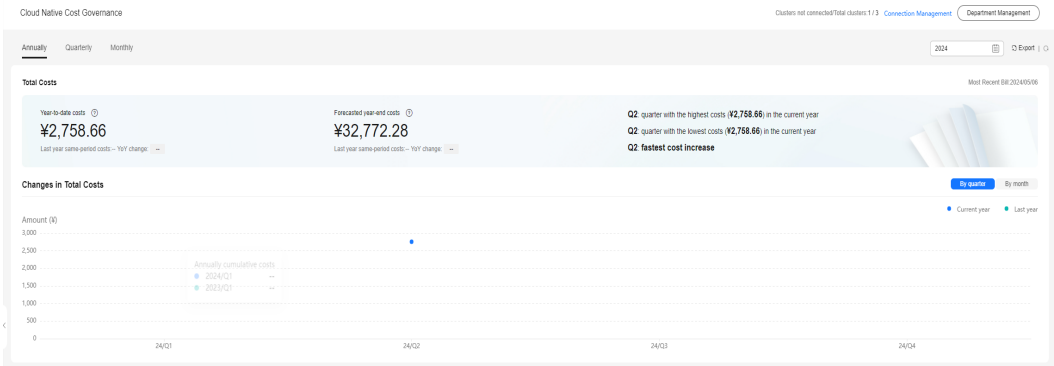


Table 11-6 Functions on the Cost Insights page

Parameter	Report	Description
Year-to-date costs/Last year same-period costs/YoY change	Annually	**Year-to-date costs**: costs generated from the start of the current year to the date of the most recent bill **Last year same-period costs**: costs generated during the same period in the last year **YoY change**: (Year-to-date costs – Last year same-period costs)/Last year same-period costs

Parameter	Report	Description
Forecasted year-end costs/Last year same-period costs/YoY change	Annually	**Forecasted year-end costs**: estimated total costs by the end of the current year **Last year same-period costs**: costs generated in last year **YoY change**: (Forecasted year-end costs – Last year same-period costs)/Last year same-period costs
Quarter-to-date costs/ Last quarter same-period costs/QoQ change	Quarterly	**Quarter-to-date costs**: costs generated from the start of the current quarter to the date of the most recent bill **Last quarter same-period costs**: costs generated during the same period in the last quarter **QoQ change**: (Quarter-to-date costs – Last quarter same-period costs)/Last quarter same-period costs
Forecasted quarter-end costs/Last quarter same-period costs/QoQ change	Quarterly	**Forecasted quarter-end costs**: estimated total costs by the end of the current quarter **Last quarter same-period costs**: costs generated in last quarter **QoQ change**: (Forecasted quarter-end costs – Last quarter same-period costs)/ Last quarter same-period costs
Month-to-date costs/Last month same-period costs/MoM change	Monthly	**Month-to-date costs**: costs generated from the start of the current month to the date of the most recent bill **Last month same-period costs**: costs generated during the same period in the last month **MoM change**: (Month-to-date costs – Last month same-period costs)/Last month same-period costs
Forecasted month-end costs/Last month same-period costs/MoM change	Monthly	**Forecasted month-end costs**: estimated total costs by the end of the current month **Last month same-period costs**: costs generated in last month **MoM change**: (Forecasted month-end costs – Last month same-period costs)/ Last month same-period costs

Parameter	Report	Description
Changes in Total Costs	Annual, quarterly, and monthly	Cost details of the current year, quarter, and month, as well as cost changes compared with the last year, quarter, and month

Department Cost Analysis

View department cost analysis reports in **Department Cost Analysis**.

Figure 11-21 Viewing department cost analysis reports

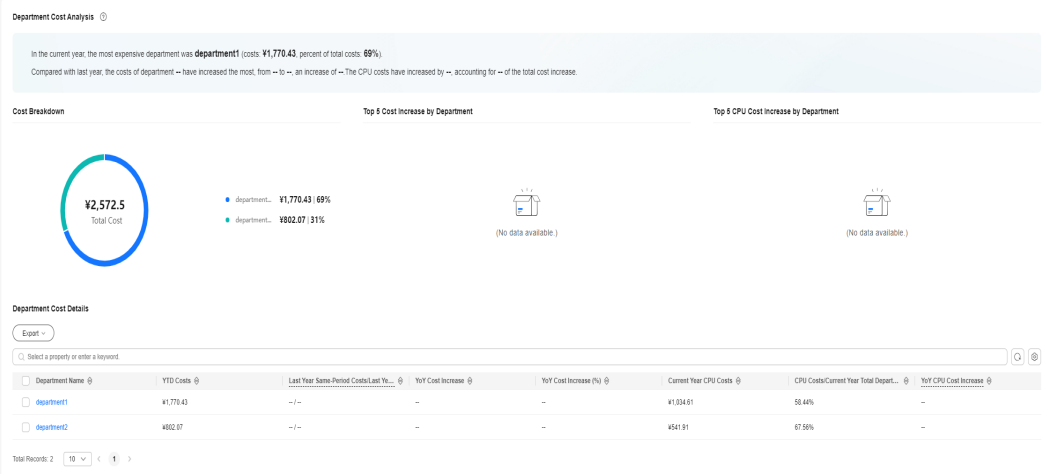


Table 11-7 Parameters in Department Cost Details

Parameter	Report	Description
Department Name	Annual, quarterly, and monthly	Department name you set.
YTD Costs	Annually	Costs generated from the start of the current year to the date of the most recent bill
Last Year Same-Period Costs	Annually	Costs generated during the same period in the last year
Last Year Total Costs	Annually	Costs generated in last year
YoY Cost Increase	Annually	Year-to-date costs – Last year same-period costs

Parameter	Report	Description
YoY Cost Increase (%)	Annually	(Year-to-date costs – Last year same-period costs)/Last year same-period costs
Current Year CPU Costs	Annually	CPU costs generated from the start of the current year to the date of the most recent bill
CPU Costs/Current Year Total Department Costs	Annually	Current year CPU costs/Year-to-date costs (department costs)
YoY CPU Cost Increase	Annually	Current year CPU costs – Last year same-period CPU costs
QTD Costs	Quarterly	Costs generated from the start of the current quarter to the date of the most recent bill
Last Quarter Same-Period Costs	Quarterly	Costs generated during the same period in the last quarter
Last Quarter Total Costs	Quarterly	Costs generated in last quarter
QoQ Cost Increase	Quarterly	Quarter-to-date costs – Last quarter same-period costs
QoQ Cost Increase (%)	Quarterly	(Quarter-to-date costs – Last quarter same-period costs)/Last quarter same-period costs
Current Quarter CPU Costs	Quarterly	CPU costs generated from the start of the current quarter to the date of the most recent bill
CPU Costs/Current Quarter Total Department Costs	Quarterly	Current quarter CPU costs/Quarter-to-date costs (department costs)
QoQ CPU Cost Increase	Quarterly	Current quarter CPU costs – Last quarter same-period CPU costs
MTD Costs	Monthly	Costs generated from the start of the current month to the date of the most recent bill
Last Month Same-Period Costs	Monthly	Costs generated during the same period in the last month
Last Month Total Costs	Monthly	Costs generated in last month.
MoM Cost Increase	Monthly	Month-to-date costs – Last month same-period costs

Parameter	Report	Description
MoM Cost Increase (%)	Monthly	(Month-to-date costs – Last month same-period costs)/Last month same-period costs
Current Month CPU Costs	Monthly	CPU costs generated from the start of the current month to the date of the most recent bill
CPU Costs/Current Month Total Department Costs	Monthly	Current month CPU costs/Month-to-date costs (department costs)
MoM CPU Cost Increase	Monthly	Current month CPU costs – Last-month same-period CPU costs

11.3.5 Cost Insights for a Department

Cost Insights for a department provides the cost analysis report of that department. The department cost analysis module displays the overall department cost status. You click the name of a department in the department list to view the cost details.

Prerequisites

- Cost Insights has been enabled.
- Department configurations are complete.

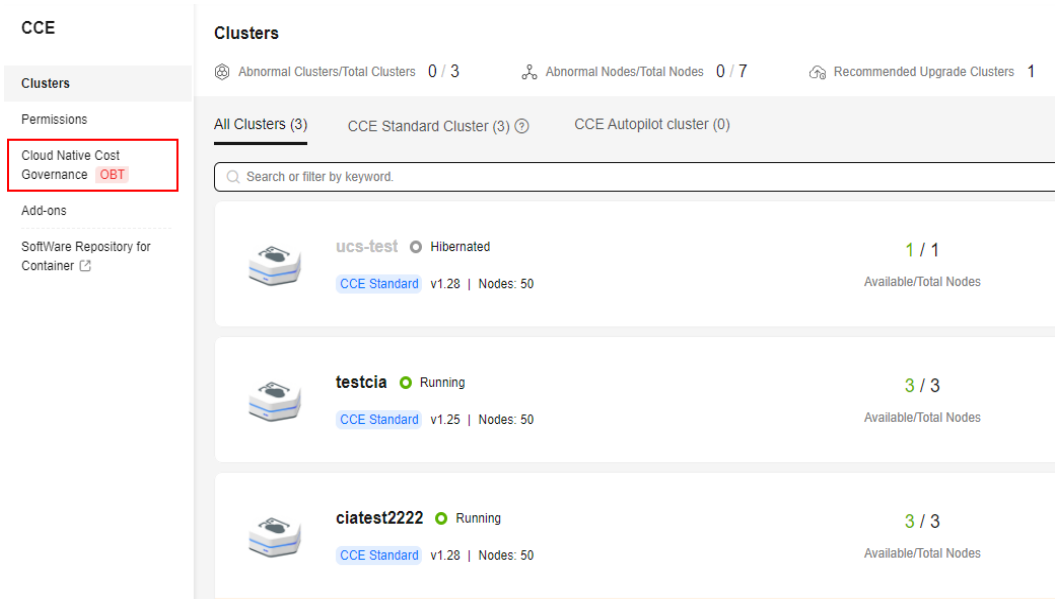
Constraints

- Processing bills takes some time. After Cost Insights is enabled, there is about two days delay before you can view your costs.
- Cloud Native Cluster Monitoring must run normally to ensure accurate data displays of namespaces, workloads, and node pools on the **Cost Insights** page.

Navigation Path

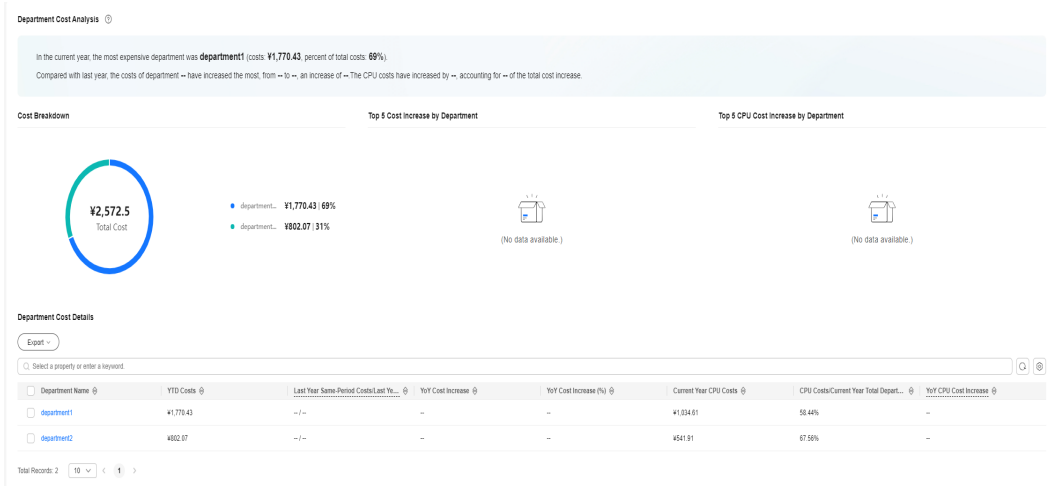
Step 1 Log in to the CCE console. In the navigation pane, choose **Cloud Native Cost Governance**.

Figure 11-22 Cloud Native Cost Governance



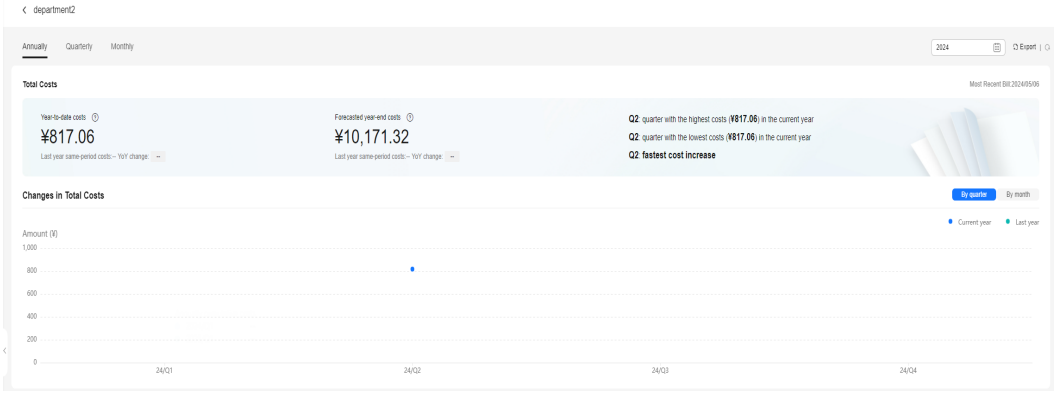
Step 2 View cost analysis reports in **Department Cost Analysis**.

Figure 11-23 Viewing department cost analysis reports



Step 3 In **Department Cost Details**, click the name of a department to view the cost details.

Figure 11-24 View the costs of a department



----End

Table 11-8 Parameters on the cost details page of a department

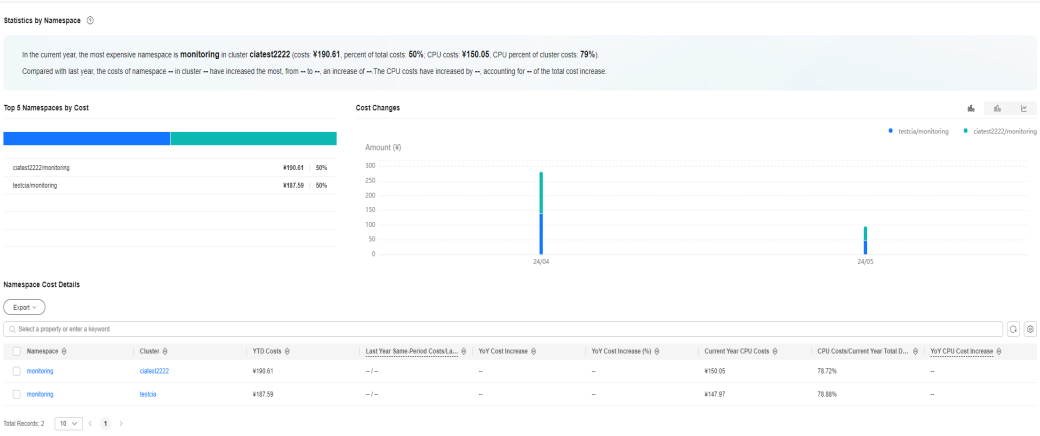
Parameter	Report	Description
Year-to-date costs/Last year same-period costs/YoY change	Annually	**Year-to-date costs**: costs generated by the current department from the start of the current year to the date of the most recent bill **Last year same-period costs**: costs generated by the current department during the same period in the last year **YoY change**: (Year-to-date costs – Last year same-period costs)/Last year same-period costs
Forecasted year-end costs/Last year same-period costs/YoY change	Annually	**Forecasted year-end costs**: estimated total costs of the current department by the end of the current year **Last year same-period costs**: costs generated by the current department in last year **YoY change**: (Forecasted year-end costs – Last year same-period costs)/ Last year same-period costs

Parameter	Report	Description
Quarter-to-date costs/ Last quarter same-period costs/QoQ change	Quarterly	**Quarter-to-date costs**: costs generated by the current department from the start of the current quarter to the date of the most recent bill **Last quarter same-period costs**: costs generated by the current department during the same period in the last quarter **QoQ change**: (Quarter-to-date costs – Last quarter same-period costs)/ Last quarter same-period costs
Forecasted quarter-end costs/Last quarter same-period costs/QoQ change	Quarterly	**Forecasted quarter-end costs**: estimated total costs of the current department by the end of the current quarter **Last quarter same-period costs**: costs generated by the current department in last quarter **QoQ change**: (Forecasted quarter-end costs – Last quarter same-period costs)/Last quarter same-period costs
Month-to-date costs/Last month same-period costs/MoM change	Monthly	**Month-to-date costs**: costs generated by the current department from the start of the current month to the date of the most recent bill **Last month same-period costs**: costs generated by the current department during the same period in the last month **MoM change**: (Month-to-date costs – Last month same-period costs)/Last month same-period costs
Forecasted month-end costs/Last month same-period costs/MoM change	Monthly	**Forecasted month-end costs**: estimated total costs of the current department by the end of the current month **Last month same-period costs**: costs generated by the current department in last month **MoM change**: (Forecasted month-end costs – Last month same-period costs)/Last month same-period costs

Parameter	Report	Description
Changes in Total Costs	Annual, quarterly, and monthly	Cost details of the current year, quarter, and month, as well as cost changes compared with the last year, quarter, and month

Cost statistics vary depending on department cost allocation model. The cost statistics do not involve shared costs. In the following example, department costs are allocated only by namespace, so only the cost statistics by namespace are displayed.

Figure 11-25 Viewing the cost statistics by namespace



11.3.6 Cost Insights for a Cluster

Cost Insights for a cluster helps cost O&M personnel analyze cluster costs and resource usages from multiple dimensions, such as namespace, application, and node pool, to identify applications that can be optimized. Currently, the cluster and namespace dimensions are supported.

Prerequisites

- Cost Insights has been enabled.

Constraints

- Processing bills takes some time. After Cost Insights is enabled, there is about two days delay before you can view your costs.
- Cloud Native Cluster Monitoring must run normally to ensure accurate data displays of namespaces, workloads, and node pools on the **Cost Insights** page.

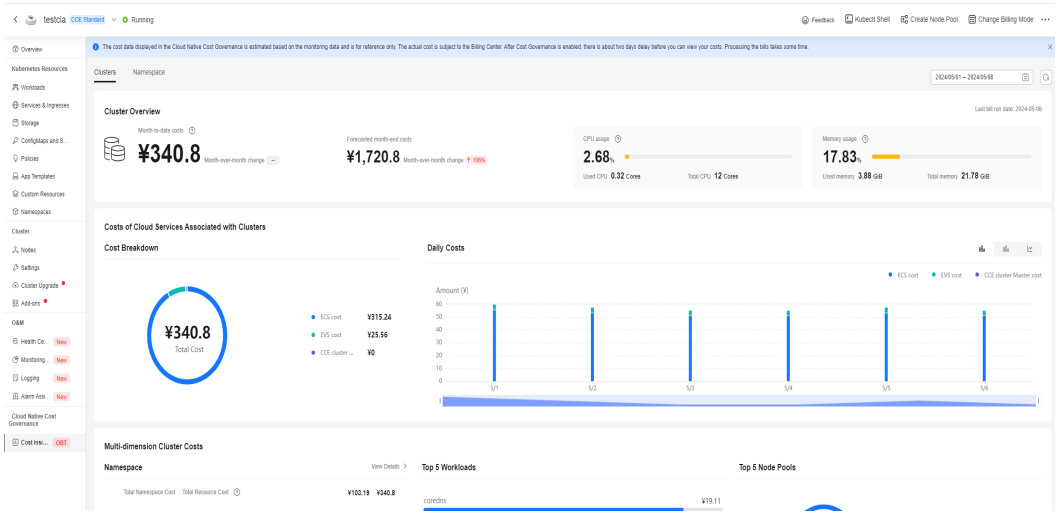
Navigation Path

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Cloud Native Cost Governance** > **Cost Insights**.

Step 3 On the displayed page, analyze costs.

Figure 11-26 Cost Insights for a cluster



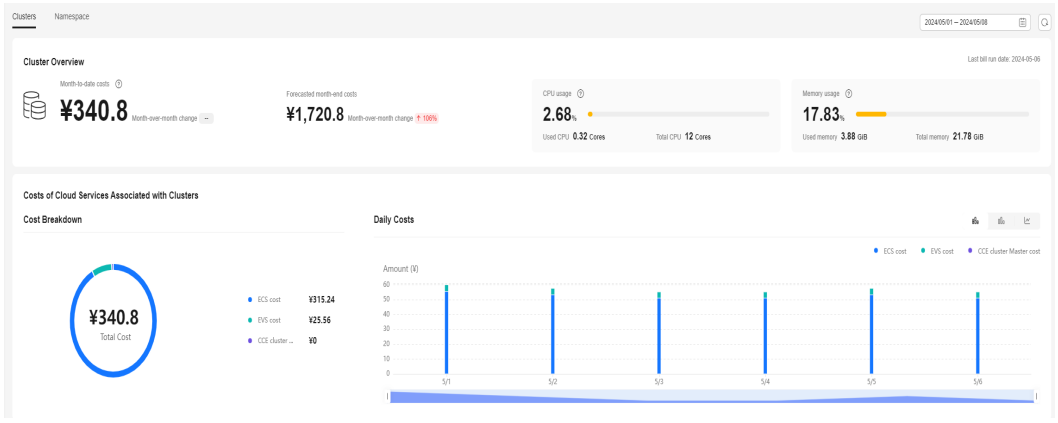
----End

Clusters

The **Clusters** tab displays the cost overview on a cluster, including the cost overhead and resource consumption of namespaces, workloads, and node pools. This helps O&M personnel identify applications with high cost overhead and low resource utilization.

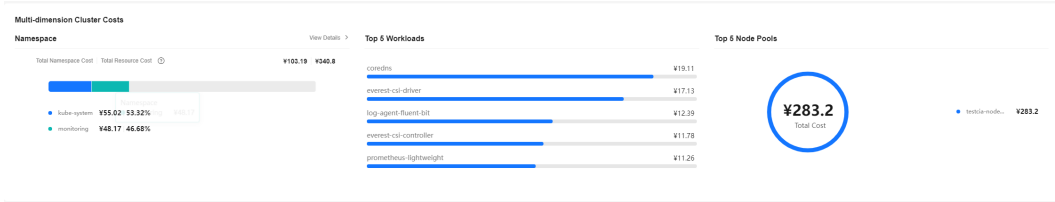
You can filter data by time in the upper right corner.

Figure 11-27 Cost overview of a cluster



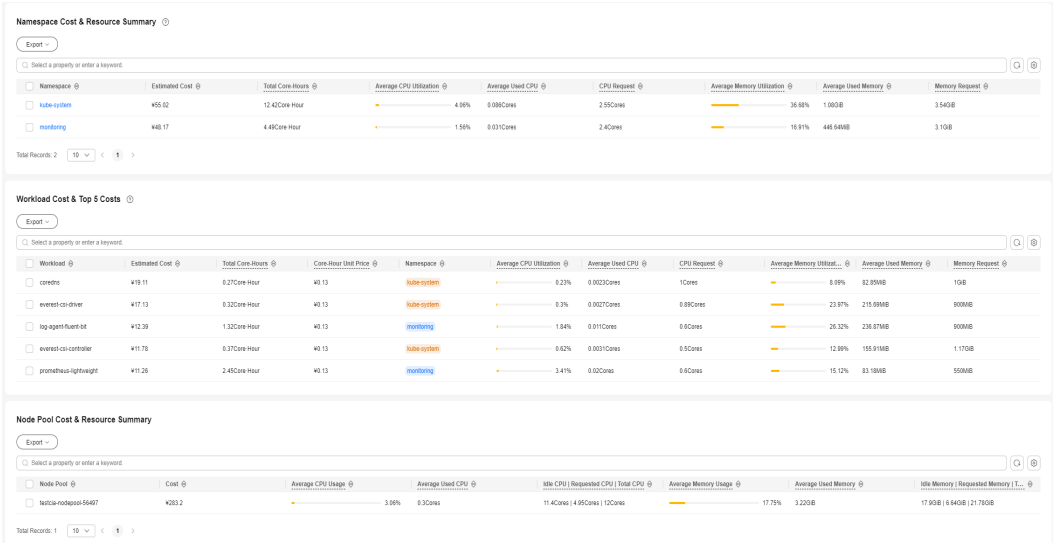
Parameter	Description
Month-to-date costs Month-over-month change	**Month-to-date costs**: costs generated by the cluster from the start of the current month to the date of the most recent bill. If Cost Insights is enabled in the current month, the cost is accumulated from the date when Cost Insights is enabled to the date of the most recent bill. **Month-over-month change**: (Month-to-date costs – Last month same-period costs)/Last month same-period costs
Forecasted month-end costs Month-over-month change	**Forecasted month-end costs**: estimated total costs by the end of the current month. **Month-over-month change**: (Forecasted month-end costs – Last month total costs)/Last month total costs
CPU usage Used CPU Total CPU	**CPU usage**: average CPU usage of all nodes in the cluster at the current time. Formula: CPU usage = Used CPU on all nodes/Total CPU on all nodes × 100%
Memory usage Used memory Total memory	Memory usage: average memory usage of all nodes in the cluster at the current time. Formula: Memory usage = Used memory on all nodes/Total memory on all nodes × 100%
Cost Breakdown	Cost breakdown during the specified time period, including the ECS cost, EVS cost, and CCE cluster management cost.
Daily Costs	Daily cost breakdown, which can be used to identify resources with high costs in a cluster.

Figure 11-28 Multi-dimension cluster costs



Parameter	Description
Namespace	**Total Namespace Cost** indicates the total cost (CPU cost, memory cost, and EVS cost) of the workloads in each selected namespace. **Total Resource Cost** indicates the total cost of compute resources (all ECSs and EVS disks) used by a cluster. Total resource cost = Total namespace cost + Unallocated cost The larger the gray area is, the more resources are not used, causing resource wastes.
Top 5 Workloads	Top 5 workloads with high costs.
Top 5 Node Pools	Top 5 mode pools with high costs.

Figure 11-29 Cost & resource summary



Module	Parameter	Description
Namespace Cost & Resource Summary	Namespace	Namespace name.
	Estimated Cost	Namespace cost: the cost calculated based on the percentage of total node resources (CPU and memory) that are used in the namespaces, plus the cost of storage used by the workloads in the namespaces.
	Total Core-Hours	Total number of core-hours consumed by the workloads in a namespace during the specified time period.

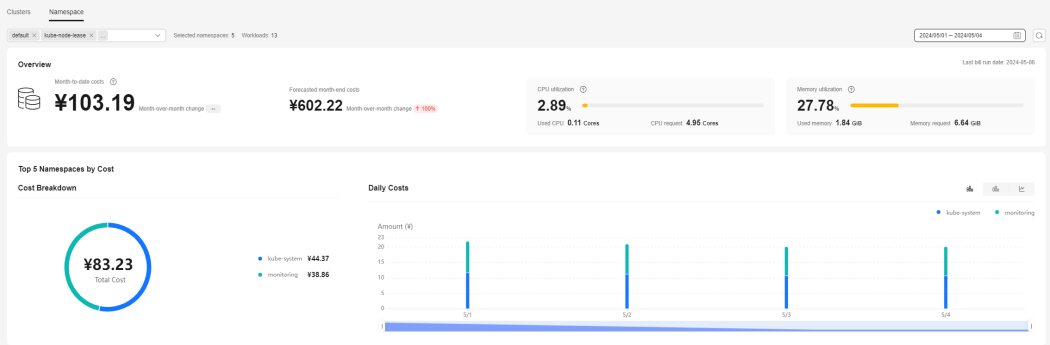
Module	Parameter	Description
	Average CPU Utilization	Average CPU usage of the workloads in a namespace during the specified time period. CPU usage = Used CPU/CPU request x 100%
	Average Used CPU	Average CPU used by the workloads in a namespace during the specified time period.
	CPU Request	CPU request in a namespace on the last day of the specified time period. It is the total CPU request of the workloads in the namespace.
	Average Memory Utilization	Average memory usage of the workloads in a namespace during the specified time period. Memory usage = Used memory/Memory request x 100%
	Average Used Memory	Average memory used by the workloads in a namespace during the specified time period.
	Memory Request	Memory request in a namespace on the last day of the specified time period. It is the total memory request of the workloads in the namespace.
Workload Cost & Top 5 Costs	Workload	Workload name.
	Estimated Cost	Workload cost: the cost calculated based on the percentage of total node resources (CPU and memory) that are used by a workload, plus the cost of storage used by the workload.
	Total Core-Hours	Total number of core-hours consumed by a workload during the specified time period.
	Core-Hour Unit Price	Price per CPU core per hour. If the core-hour unit price of a workload or namespace is high, you can change the node type to reduce costs and improve resource utilization.
	Namespace	Namespace that a workload belongs to.
	Average CPU Utilization	Average CPU usage of a workload during the specified time period. CPU usage = Used CPU/CPU request x 100%
	Average Used CPU	Average CPU used by a workload during the specified time period.
	CPU Request	CPU request of a workload on the last day of the specified time period.

Module	Parameter	Description
	Average Memory Utilization	Average memory usage of a workload during the specified time period. Memory usage = Used memory/Memory request × 100%
	Average Used Memory	Average memory used by a workload during the specified time period.
	Memory Request	Memory request of a workload on the date of the most recent bill during the specified time period.
Node Pool Cost & Resource Summary	Node Pool	Node pool name.
	Cost	Cost of nodes in a node pool during the specified time period.
	Average CPU Usage	Average CPU usage of a node pool during the specified time period. CPU usage = Total CPU used by nodes in a node pool/Total CPU in a node pool × 100%
	Average Used CPU	Average CPU used by a node pool during the specified time period.
	Idle CPU | Requested CPU | Total CPU	**Idle CPU**: total idle CPUs of all nodes in a node pool on the last day of the specified time period. **Requested CPU**: total CPUs requested by all nodes in a node pool on the last day of the specified time period. **Total CPU**: total CPUs of all nodes in a node pool on the last day of the specified time period.
	Average Memory Usage	Average memory usage of a node pool during the specified time period. Memory usage = Total memory used by nodes in a node pool/Total memory in a node pool x 100%
	Average Used Memory	Average memory used by a node pool during the specified time period.
	Idle Memory | Requested Memory | Total Memory	**Idle Memory**: total idle memory of all nodes in a node pool on the last day of the specified time period. **Requested Memory**: total memory requested by all nodes in a node pool on the last day of the specified time period. **Total Memory**: total memory of all nodes in a node pool on the last day of the specified time period.

Namespaces

The **Namespaces** tab displays the cost optimization analysis on the selected namespace and workloads in that namespace, which allow you to identify workloads with high overhead and low utilization for coat optimization.

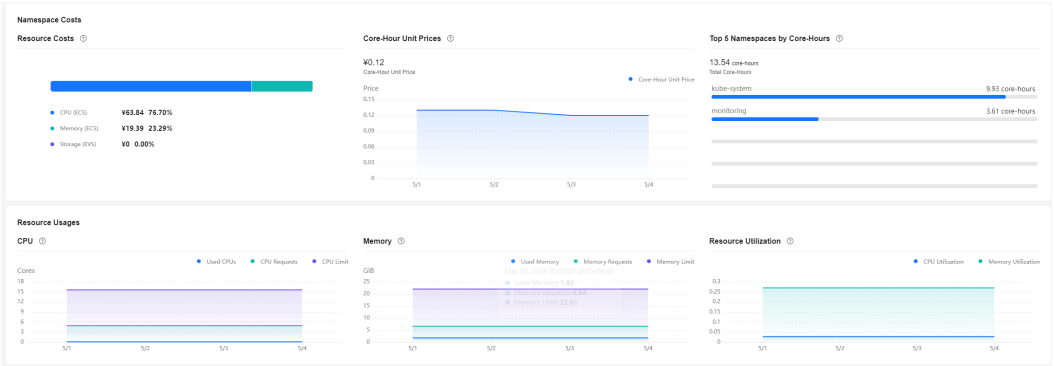
Figure 11-30 Cost overview by namespace



Parameter	Description
Month-to-date costs Month-over-month change	**Month-to-date costs**: costs generated in selected namespaces from the start of the current month to the date of the most recent bill. If Cost Insights is enabled in the current month, the cost is accumulated from the date when Cost Insights is enabled to the date of the most recent bill. **Month-over-month change**: (Month-to-date costs – Last month same-period costs)/Last month same-period costs
Forecasted month-end costs Month-over-month change	**Forecasted month-end costs**: estimated total costs in the selected namespaces by the end of the current month. **Month-over-month change**: (Forecasted month-end costs – Last month total costs)/Last month total costs
CPU utilization Used CPU CPU request	**CPU utilization**: average CPU usage in the selected namespaces at the current time. Formula: CPU usage = Total used CPU in the selected namespaces/Total CPU requests in the selected namespaces × 100%
Memory utilization Used memory Memory request	**Memory utilization**: Average memory usage in the selected namespaces at the current time. Formula: Memory usage = Total used memory in the selected namespaces/Total memory requests in the selected namespaces × 100%
Cost Breakdown	Cost breakdown of top 5 namespaces among the selected namespaces during the specified time period.

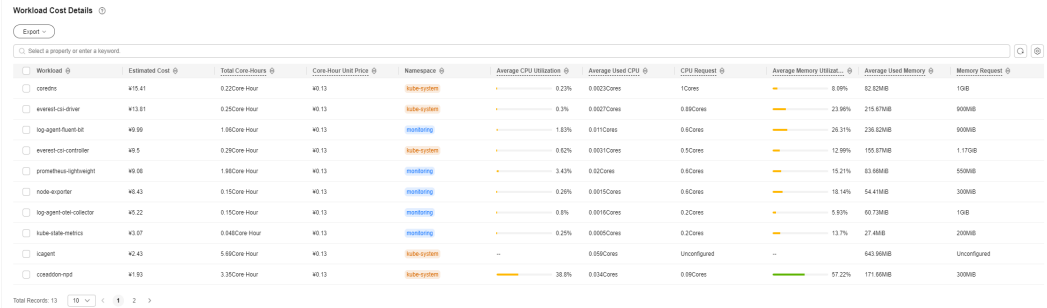
Parameter	Description
Daily Costs	Daily cost breakdown of the selected namespaces for identifying namespaces with high costs.

Figure 11-31 Namespace costs



Parameter	Description
Resource Costs	Costs of resources in the selected namespaces during the specified time period, which consist of the CPU cost, memory cost, and storage cost.
Core-Hour Unit Prices	Average unit prices of core-hours consumed by workloads on the nodes in the selected namespaces.
Top 5 Namespaces by Core-Hours	Total core-hours consumed by resources in top 5 namespaces among the selected namespaces during the specified time period.
CPU	Changes of CPU usages, requests, and limits in the selected namespaces.
Memory	Changes of memory usages, requests, and limits in the selected namespaces.
Resource Utilization	Changes of CPU usages and memory usages in the selected namespaces.

Figure 11-32 Workload cost details



Parameter	Description
Workload	Workload name.
Estimated Cost	The cost calculated based on the percentage of total node resources (CPU and memory) that are used by a workload, plus the cost of storage used by the workload.
Total Core-Hours	Total number of core-hours consumed by a workload during the specified time period, reflecting the CPU usage.
Core-Hour Unit Price	Price per CPU core per hour of the node where the workload is located, which can be used for node model optimization.
Namespace	Namespace that a workload belongs to.
Average CPU Utilization	Average CPU usage of a workload during the specified time period. CPU usage = Used CPU/CPU request × 100%
Average Used CPU	Average CPU used by a workload during the specified time period.
CPU Request	CPU request of a workload on the last day of the specified time period.
Average Memory Utilization	Average memory usage of a workload during the specified time period. Memory usage = Used memory/Memory request × 100%
Average Used Memory	Average memory used by a workload during the specified time period.
Memory Request	Memory request of a workload on the last day of the specified time period.

12 Namespaces

12.1 Creating a Namespace

Scenario

A namespace is a collection of resources and objects. Multiple namespaces can be created inside a cluster and isolated from each other. This enables namespaces to share the same cluster Services without affecting each other.

For example, you can deploy workloads in a development environment into one namespace, and deploy workloads in a testing environment into another namespace.

Prerequisites

At least one cluster has been created.

Notes and Constraints

A maximum of 6000 Services can be created in each namespace. The Services mentioned here indicate the Kubernetes Service resources added for workloads.

Namespace Types

Namespaces can be categorized based on how they are created.

- Created automatically: When a cluster is up, the **default**, **kube-public**, **kube-system**, and **kube-node-lease** namespaces are created by default.
  - **default**: All objects for which no namespace is specified are allocated to this namespace.
  - **kube-public**: Resources in this namespace can be accessed by all users (including unauthenticated users) so that some resources in the cluster can be readable in the entire cluster. This is a reserved Kubernetes namespace. Its common attributes are only conventions but not requirements.

  - **kube-system**: All resources created by Kubernetes are in this namespace.
  - **kube-node-lease**: Each node has an associated Lease object in this namespace. The object is periodically updated by the node. Both NodeStatus and NodeLease are considered as heartbeats from a node. In versions earlier than v1.13, only NodeStatus is available. The NodeLease feature is introduced in v1.13. NodeLease is more lightweight than NodeStatus. This feature significantly improves the cluster scalability and performance.

- Created manually: You can create namespaces to serve separate purposes. For example, you can create three namespaces, one for a development environment, one for joint debugging environment, and one for test environment. You can also create one namespace for login services and one for game services.
- Service mesh namespace: When ASM is used, namespaces named **istio-system**, **asm-system**, **istio-operator**, and **mantis-system** will be automatically created. These namespaces do not support quota management. Manually creating namespaces with these names will also affect these namespaces.

Creating a Namespace on the Console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Namespaces** in the navigation pane and click **Create Namespace** in the upper right corner.

Step 3 Set namespace parameters based on **Table 12-1**.

Table 12-1 Parameters for creating a namespace

Parameter	Description
Name	Unique name of the created namespace.
Description	Description about the namespace.
Quota Management	Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace. **NOTICE** **Configure resource quotas for each namespace to prevent overloading the cluster or nodes with excessive resources.** For example, if your cluster resources are limited but you need to use multiple namespaces to isolate resources of different services, you can configure resource quotas for each namespace to maintain cluster stability and proper resource allocation. Enter a value. If a resource quota is not specified, there will be no limit imposed on the resource. If you want to limit the CPU or memory quota, you must specify the CPU or memory request value when creating a workload.

Step 4 After the configuration is complete, click **OK**.

----End

Creating a Namespace Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a namespace.

- Method 1: Use a YAML file to configure the namespace and run the **kubectl apply** command to create it.
  a. Create a **custom-namespace.yaml** file with the following data:
  ```
  apiVersion: v1
  kind: Namespace
  metadata:
    name: custom-namespace
  ```
  b. Run the following command to create the namespace:
  ```
  kubectl apply -f custom-namespace.yaml
  ```
  Command output:
  ```
  namespace/custom-namespace created
  ```
- Method 2: Run the **kubectl create namespace** command to create a namespace.
  ```
  kubectl create namespace custom-namespace
  ```

Step 3 Run the following command to verify the creation:

```
kubectl get namespaces
```

The **custom-namespace** namespace is displayed in the list.

For more details about how to configure a namespace, see **Manage Memory, CPU, and API Resources**.

----End

12.2 Managing Namespaces

Using Namespaces

- When creating a workload, you can select a namespace to isolate resources or users.
- When querying workloads, you can select a namespace to view all workloads in the namespace.

Isolating Namespaces

- **Isolating namespaces by environment**

  An application generally goes through the development, joint debugging, and testing stages before it is launched. In this process, the workloads deployed in each environment (stage) are the same, but are logically defined. There are two ways to define them:

  – Group them in different clusters for different environments.

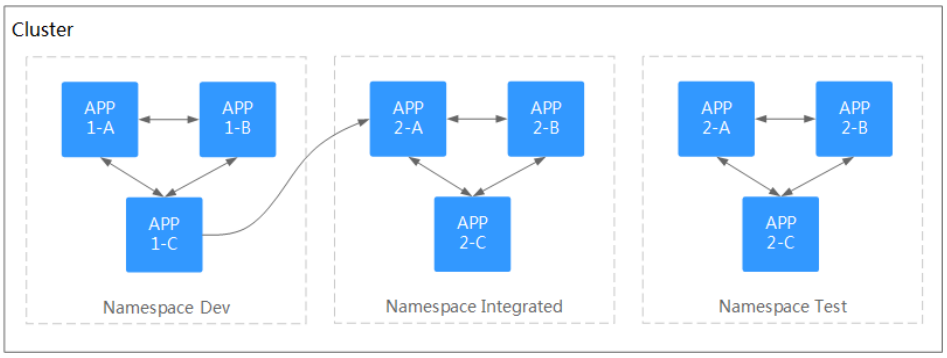
Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

- Group them in different namespaces for different environments.

  Workloads in the same namespace can be mutually accessed by using the Service name. Cross-namespace access can be implemented by using the Service name or namespace name.

  The following figure shows namespaces created for the development, joint debugging, and testing environments, respectively.

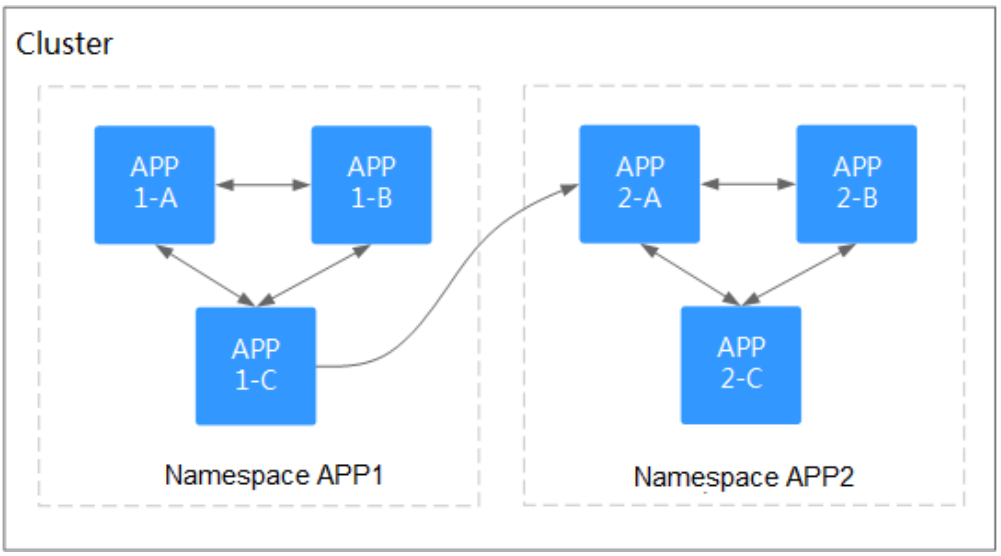
  Figure 12-1 One namespace for one environment



- **Isolating namespaces by application**

  You are advised to use this method if a large number of workloads are deployed in the same environment. For example, in the following figure, different namespaces (APP1 and APP2) are created to logically manage workloads as different groups. Workloads in the same namespace access each other using the Service name, and workloads in different namespaces access each other using the Service name or namespace name.

  Figure 12-2 Grouping workloads into different namespaces



Managing Namespace Labels

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Namespaces**.

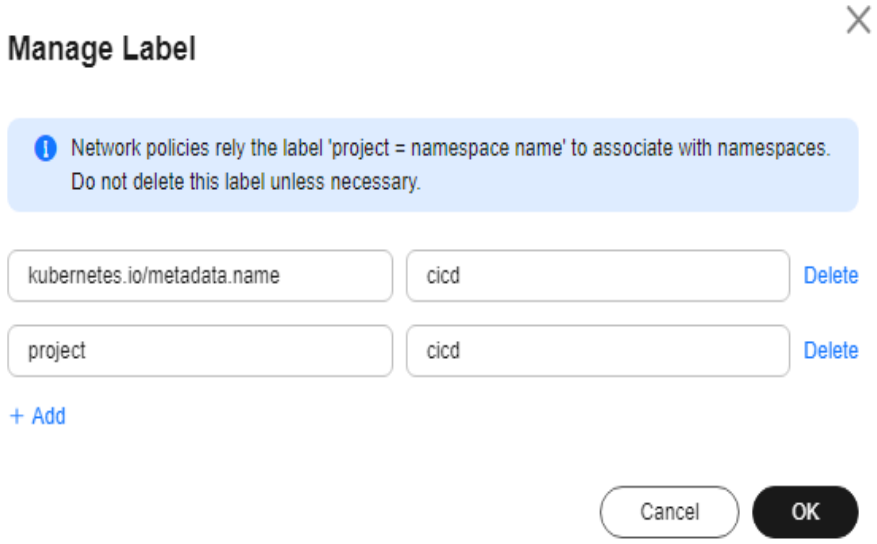
Step 2 Locate the row containing the target namespace and choose **More** > **Manage Label** in the **Operation** column.

Step 3 In the dialog box that is displayed, the existing labels of the namespace are displayed. Modify the labels as needed.

- Adding a label: Click the add icon, enter the key and value of the label to be added, and click **OK**.

  For example, the key is **project** and the value is **cicd**, indicating that the namespace is used to deploy CICD.
- Deleting a label: Click **Delete** next the label to be deleted and then **OK**.

Figure 12-3 Adding or deleting a namespace label



Step 4 Switch to the **Manage Label** dialog box again and check the modified labels.

----End

Enabling Node Affinity in a Namespace

After node affinity is enabled in a namespace, the workloads newly created in the namespace can be scheduled only to nodes with specific labels. For details, see **PodNodeSelector**.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Namespaces**.

Step 2 Locate the target namespace and click in the **Node Affinity** column.

Step 3 In the displayed dialog box, select **Enable** and click **OK**.

After node affinity is enabled, new workloads in the current namespace will be scheduled only to nodes with specified labels. For example, in namespace **test**, the

workloads in the namespace can be scheduled only to the node whose label key is **kubelet.kubernetes.io/namespace** and label value is **test**.

Step 4 You can add specified labels to a node in **Labels and Taints** on the **Nodes** page. For details, see **Managing Node Labels**.

----End

Deleting a Namespace

If a namespace is deleted, all resources (such as workloads, jobs, and ConfigMaps) in this namespace will also be deleted. Exercise caution when deleting a namespace.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Namespaces** in the navigation pane. On the displayed page, click **More** in the row of the target namespace and choose **Delete**.

Follow the prompts to delete the namespace. The default namespaces cannot be deleted.

----End

12.3 Configuring Resource Quotas

Kubernetes provides namespaces for you to group resources in a cluster. Namespaces serve different purposes to meet the needs of multiple users, environments, and applications. To prevent resource abuse and ensure cluster reliability, you can configure quotas for resources like CPUs, memory, and pods for each namespace. For details, see **Resource Quotas**.

In clusters of v1.21 or later, the default resource quotas will be created when a namespace is created if you have enabled **enable-resource-quota** in **Cluster Configuration Management**. **Table 12-2** lists the resource quotas based on cluster specifications. You can modify them according to your service requirements.

Table 12-2 Default resource quotas

Cluster Scale	Pod	Deployment	Secret	ConfigMap	Service
50 nodes	2000	1000	1000	1000	1000
200 nodes	2000	1000	1000	1000	1000
1000 nodes	5000	2000	2000	2000	2000
2000 nodes	5000	2000	2000	2000	2000

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, click **Namespaces**.

Step 3 Click **Quota Management** next to the target namespace.

This operation cannot be performed on system namespaces **kube-system** and **kube-public**.

Step 4 Configure the resource quotas and click **OK**.

NOTICE

- After configuring CPU and memory quotas for a namespace, you must specify the request and limit values of CPU and memory resources when creating a workload. Otherwise, the workload cannot be created. If the quota of a resource is set to **0**, the resource usage is not limited.
- Accumulated quota usage includes the resources used by CCE to create default components, such as the Kubernetes Services (which can be viewed using kubectl) created under the **default** namespace. Therefore, you are advised to set a resource quota greater than expected to reserve resource for creating default components.
- Kubernetes provides optimistic concurrency control (OCC), also known as optimistic locking, for frequent data updates. You can use optimistic locking by defining the **resourceVersion** field. This field is in the object metadata. This field identifies the internal version number of the object. When the object is modified, this field is modified accordingly. You can use kube-apiserver to check whether an object has been modified. When the API server receives an update request containing the **resourceVersion** field, the server compares the requested data with the resource version number of the server. If they are different, the object on the server has been modified when the update is submitted. In this case, the API server returns a conflict error (409). Obtain the server data, modify the data, and submit the data to the server again. Resource quotas limit the total resource usage of each namespace and maintain a record of resource information in the cluster. Enabling resource quotas may result in more resource creation conflicts in high-concurrency scenarios, which can affect the performance of batch resource creation.

----End

13 ConfigMaps and Secrets

13.1 Creating a ConfigMap

Scenario

A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. After creating ConfigMaps, you can use them as files or environment variables in a containerized workload.

ConfigMaps allow you to decouple configuration files from container images to enhance the portability of workloads.

Benefits of ConfigMaps:

- Manage configurations of different environments and services.
- Deploy workloads in different environments. Multiple versions are supported for configuration files so that you can update and roll back workloads easily.
- Quickly import configurations in the form of files to containers.

Notes and Constraints

- The size of a ConfigMap resource file cannot exceed 1 MB.
- ConfigMaps cannot be used in **static pods**.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **ConfigMaps and Secrets** in the navigation pane and click **Create ConfigMap** in the upper right corner.

Step 3 Configure parameters.

Table 13-1 Parameters for creating a ConfigMap

Parameter	Description
Name	Name of the ConfigMap you create, which must be unique in a namespace.
Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of the ConfigMap.
Data	Data of a ConfigMap, in the key-value pair format. Click + to add data. The value can be in string, JSON, or YAML format.
Label	Label of the ConfigMap. Enter a key-value pair and click **Confirm**.

Step 4 Click **OK**.

The new ConfigMap is displayed in the ConfigMap list.

----End

Creating a ConfigMap Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **cce-configmap.yaml** and edit it.

vi cce-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

Table 13-2 Key parameters

Parameter	Description
apiVersion	The value is fixed at **v1**.
kind	The value is fixed at **ConfigMap**.
metadata.name	ConfigMap name, which can be customized.
data	ConfigMap data. The value must be key-value pairs.

Step 3 Run the following commands to create a ConfigMap.

kubectl create -f cce-configmap.yaml

Run the following commands to view the created ConfigMap:

kubectl get cm

```
NAME            DATA      AGE
cce-configmap   3         7m
```

----End

Related Operations

After creating a ConfigMap, you can update or delete it as described in **Table 13-3**.

Table 13-3 Related operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target ConfigMap resides to edit its YAML file.
Updating a ConfigMap	1. Select the name of the ConfigMap to be updated and click **Update**. 2. Modify the secret data. For more information, see **Table 13-1**. 3. Click **OK**.
Deleting a ConfigMap	Select the configuration you want to delete and click **Delete**. Follow the prompts to delete the ConfigMap.

13.2 Using a ConfigMap

After a ConfigMap is created, it can be used in three workload scenarios: environment variables, command line parameters, and data volumes.

- **Configuring Environment Variables of a Workload**
- **Configuring Command Line Parameters**
- **Mounting a ConfigMap to the Workload Data Volume**

The following example shows how to use a ConfigMap.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```


NOTICE

- When a ConfigMap is used in a workload, the workload and ConfigMap must be in the same cluster and namespace.
- When a ConfigMap is mounted as a data volume and the ConfigMap is updated, Kubernetes updates the data in the data volume at the same time.

  For a ConfigMap data volume mounted in **subPath** mode, Kubernetes cannot automatically update data in the data volume when the ConfigMap is updated.
- When a ConfigMap is used as an environment variable, data is not automatically updated when the ConfigMap is updated. To update the data, restart the pod.

Configuring Environment Variables of a Workload

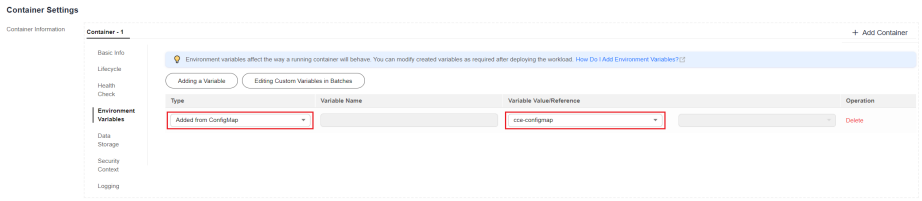
Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. Then, click **Create Workload** in the upper right corner.

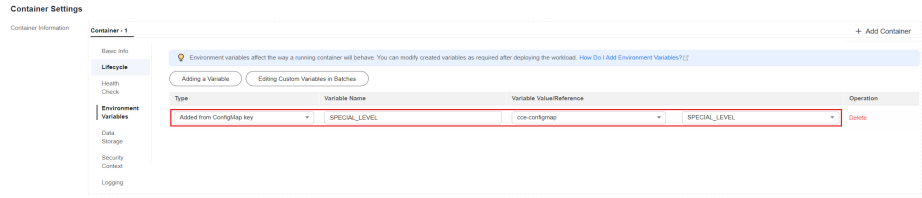
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **Added from ConfigMap**: Select a ConfigMap to import all of its keys as environment variables.



- **Added from ConfigMap key**: Import a key in a ConfigMap as the value of an environment variable.
  - **Variable Name**: name of an environment variable in the workload. The name can be customized and is set to the key name selected in the ConfigMap by default.
  - **Variable Value/Reference**: Select a ConfigMap and the key to be imported. The corresponding value is imported as a workload environment variable.

  For example, after you import the value **Hello** of **SPECIAL_LEVEL** in ConfigMap **cce-configmap** as the value of workload environment variable **SPECIAL_LEVEL**, an environment variable named **SPECIAL_LEVEL** with its value **Hello** exists in the container.



Step 3 Configure other workload parameters and click **Create Workload**.

After the workload runs properly, **log in to the container** and run the following statement to check whether the ConfigMap has been set as an environment variable of the workload:

```
printenv SPECIAL_LEVEL
```

The example output is as follows:

```
Hello
```

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

Content of the YAML file:

- **Added from ConfigMap**: To add all data in a ConfigMap to environment variables, use the **envFrom** parameter. The keys in the ConfigMap will become names of environment variables in the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        envFrom:                    # Use envFrom to specify a ConfigMap to be referenced by environment variables.
        - configMapRef:
            name: cce-configmap     # Name of the referenced ConfigMap.
      imagePullSecrets:
      - name: default-secret
```

- **Added from ConfigMap key**: When creating a workload, you can use a ConfigMap to set environment variables and use the **valueFrom** parameter to reference the key-value pair in the ConfigMap separately.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
```



```
      app: nginx-configmap
  spec:
    containers:
    - name: container-1
      image: nginx:latest
      env:                          # Set the environment variable in the workload.
      - name: SPECIAL_LEVEL         # Name of the environment variable in the workload.
        valueFrom:                  # Specify a ConfigMap to be referenced by the environment variable.
          configMapKeyRef:
            name: cce-configmap     # Name of the referenced ConfigMap.
            key: SPECIAL_LEVEL      # Key in the referenced ConfigMap.
      - name: SPECIAL_TYPE          # Add multiple environment variables to import them at the
same time.
        valueFrom:
          configMapKeyRef:
            name: cce-configmap
            key: SPECIAL_TYPE
    imagePullSecrets:
    - name: default-secret
```

Step 3 Create a workload.

kubectl apply -f nginx-configmap.yaml

Step 4 View the environment variables in the pod.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Expected output:
   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```
2. Run the following command to view the environment variables in the pod:
   ```
   kubectl exec nginx-configmap-*** -- printenv SPECIAL_LEVEL SPECIAL_TYPE
   ```
   Expected output:
   ```
   Hello
   CCE
   ```
   The ConfigMap has been set as environment variables of the workload.

----End

Configuring Command Line Parameters

You can use a ConfigMap as an environment variable to set commands or parameter values for a container by using the environment variable substitution syntax $(VAR_NAME).

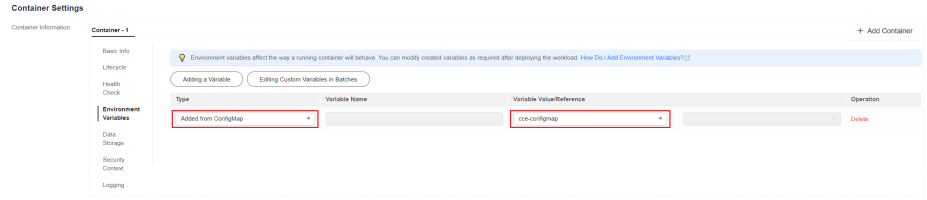
Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**. In this example, select **Added from ConfigMap**.

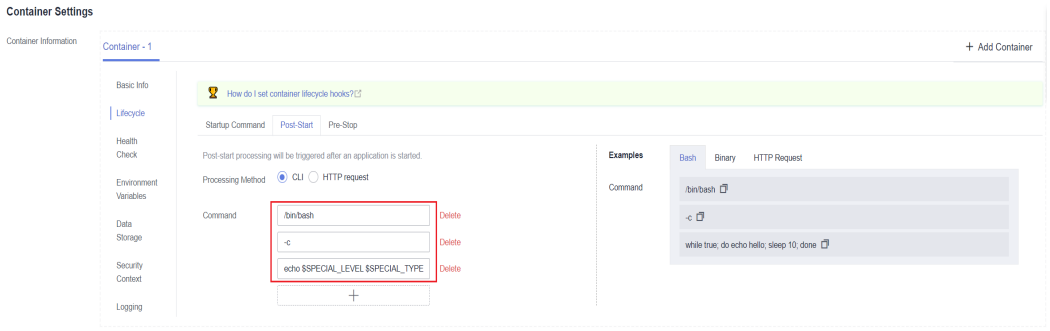
- **Added from ConfigMap**: Select a ConfigMap to import all of its keys as environment variables.



Step 3 Click **Lifecycle** in the **Container Settings** area, click the **Post-Start** tab on the right, and set the following parameters:

- **Processing Method**: **CLI**
- **Command**: Enter the following three command lines. *SPECIAL_LEVEL* and *SPECIAL_TYPE* are the environment variable names in the workload, which are key names in the **cce-configmap** ConfigMap.

```
/bin/bash
-c
echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/index.html
```



Step 4 Configure other workload parameters and click **Create Workload**.

After the workload runs properly, **log in to the container** and run the following statement to check whether the ConfigMap has been set as an environment variable of the workload:

```
cat /usr/share/nginx/html/index.html
```

The example output is as follows:

```
Hello CCE
```

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

In the following example, the **cce-configmap** ConfigMap is imported to the workload. *SPECIAL_LEVEL* and *SPECIAL_TYPE* are the environment variable names in the workload, which are key names in the **cce-configmap** ConfigMap.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
```



```
replicas: 1
selector:
  matchLabels:
    app: nginx-configmap
template:
  metadata:
    labels:
      app: nginx-configmap
  spec:
    containers:
    - name: container-1
      image: nginx:latest
      lifecycle:
        postStart:
          exec:
            command: [ "/bin/sh", "-c", "echo $SPECIAL_LEVEL $SPECIAL_TYPE > /usr/share/nginx/html/index.html" ]
      envFrom:                    # Use envFrom to specify a ConfigMap to be referenced by environment variables.
      - configMapRef:
          name: cce-configmap     # Name of the referenced ConfigMap.
    imagePullSecrets:
      - name: default-secret
```

Step 3 Create a workload.

kubectl apply -f nginx-configmap.yaml

Step 4 Wait until the workload runs properly. Then, data will be added the **/usr/share/nginx/html/index.html** file in the container.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Expected output:
   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```
2. Run the following command to view the environment variables in the pod:
   ```
   kubectl exec nginx-configmap-*** -- cat /usr/share/nginx/html/index.html
   ```
   Expected output:
   ```
   Hello CCE
   ```

----End

Mounting a ConfigMap to the Workload Data Volume

The data stored in a ConfigMap can be referenced in a volume of type ConfigMap. You can mount such a volume to a specified container path. The platform supports the separation of workload codes and configuration files. ConfigMap volumes are used to store workload configuration parameters. Before that, create ConfigMaps in advance. For details, see **Creating a ConfigMap**.

Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

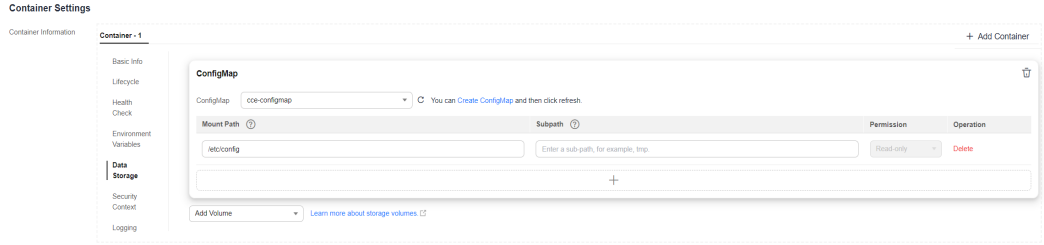
When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **ConfigMap** from the drop-down list.

Step 3 Select parameters for mounting a ConfigMap volume, as shown in **Table 13-4**.

Table 13-4 Mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created beforehand. For details, see **Creating a ConfigMap**.
Mount Path	Enter a mount path. After the ConfigMap volume is mounted, a configuration file with the key as the file name and value as the file content is generated in the mount path of the container. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If the container is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	Enter a subpath of the mount path. • A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path will be used by default. • The subpath can be the key and value of a ConfigMap. If the subpath is a key-value pair that does not exist, the data import does not take effect. • The data imported by specifying a subpath will not be updated along with the ConfigMap updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 13-1 Mounting a ConfigMap to a workload data volume



Step 4 After the configuration, click **Create Workload**.

After the workload runs properly, the **SPECIAL_LEVEL** and **SPECIAL_TYPE** files will be generated in the **/etc/config** directory in this example. The contents of the files are **Hello** and **CCE**, respectively.

Access the container and run the following statement to view the **SPECIAL_LEVEL** or **SPECIAL_TYPE** file in the container:

```
cat /etc/config/SPECIAL_LEVEL
```

Expected output:

```
Hello
```

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

As shown in the following example, after the ConfigMap volume is mounted, a configuration file with the key as the file name and value as the file content is generated in the **/etc/config** directory of the container.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-configmap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-configmap
  template:
    metadata:
      labels:
        app: nginx-configmap
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
        - name: config-volume
          mountPath: /etc/config          # Mount to the /etc/config directory.
          readOnly: true
      volumes:
      - name: config-volume
        configMap:
          name: cce-configmap             # Name of the referenced ConfigMap.
```

Step 3 Create a workload.

kubectl apply -f nginx-configmap.yaml

Step 4 After the workload runs properly, the **SPECIAL_LEVEL** and **SPECIAL_TYPE** files will be generated in the **/etc/config** directory. The contents of the files are **Hello** and **CCE**, respectively.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep nginx-configmap
   ```
   Expected output:
   ```
   nginx-configmap-***   1/1     Running   0          2m18s
   ```
2. Run the following command to view the **SPECIAL_LEVEL** or **SPECIAL_TYPE** file in the pod:
   ```
   kubectl exec nginx-configmap-*** -- cat /etc/config/SPECIAL_LEVEL
   ```


Expected output:

```
Hello
```

----End

13.3 Creating a Secret

Scenario

A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined. After creating secrets, you can use them as files or environment variables in a containerized workload.

Notes and Constraints

Secrets cannot be used in **static pods**.

Procedure

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **ConfigMaps and Secrets** in the navigation pane, click the **Secrets** tab, and click **Create Secret** in the upper right corner.

Step 3 Configure parameters.

Table 13-5 Parameters for creating a secret

Parameter	Description
Name	Name of the secret you create, which must be unique.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of a secret.
Type	Type of the secret you create. • Opaque: common secret. • kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository. • **kubernetes.io/tls**: Kubernetes TLS secret, which is used to store the certificate required by layer-7 load balancing Services. For details about examples of the kubernetes.io/tls secret and its description, see **TLS secrets**. • **IngressTLS**: TLS secret provided by CCE to store the certificate required by layer-7 load balancing Services. • Other: another type of secret, which is specified manually.

Parameter	Description
Secret Data	Workload secret data can be used in containers. • If **Secret Type** is **Opaque**, click +. In the dialog box displayed, enter a key-value pair and select **Auto Base64 Encoding**. • If **Secret Type** is **kubernetes.io/dockerconfigjson**, enter the account and password for logging in to the private image repository. • If **Secret Type** is **kubernetes.io/tls** or **IngressTLS**, upload the certificate file and private key file. **NOTE** – A certificate is a self-signed or CA-signed credential used for identity authentication. – A certificate request is a request for a signature with a private key.
Secret Label	Label of the secret. Enter a key-value pair and click **Confirm**.

Step 4 Click **OK**.

The new secret is displayed in the key list.

----End

Secret Resource File Configuration Example

This section describes configuration examples of secret resource description files.

- Opaque type

  The **secret.yaml** file is defined as shown below. The **data** field is filled in as a key-value pair, and the **value** field must be encoded using Base64. For details about the Base64 encoding method, see **Base64 Encoding**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be encoded using Base64.
type: Opaque
```

- kubernetes.io/dockerconfigjson type

  The **secret.yaml** file is defined as shown below. The value of **.dockerconfigjson** must be encoded using Base64. For details, see **Base64 Encoding**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  .dockerconfigjson: eyJh*****    # Content encoded using Base64.
type: kubernetes.io/dockerconfigjson
```


To obtain the **.dockerconfigjson** content, perform the following steps:

a. Obtain the following login information of the image repository.

- Image repository address: The section uses *address* as an example. Replace it with the actual address.
- Username: The section uses *username* as an example. Replace it with the actual username.
- Password: The section uses *password* as an example. Replace it with the actual password.

b. Use Base64 to encode the key-value pair *username:password* and fill the encoded content in **3**.

```
echo -n "username:password" | base64
```

Command output:

```
dXNlcm5hbWU6cGFzc3dvcmQ=
```

c. Use Base64 to encode the following JSON content:

```
echo -n '{"auths":{"address":
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}'
| base64
```

Command output:

```
eyJhdXRocyI6eyJhZGRyZXNzIjp7InVzZXJuYW1lIjoidXNlcm5hbWUiLCJwYXNzd29yZCI6InBhc3N3b3JkIiwiYXV0aCI6ImRYTmxjbTVoYldVNmNHRnpjM2R2Y21RPSJ9fX0=
```

The encoded content is the **.dockerconfigjson** content.

- kubernetes.io/tls type

  The value of **tls.crt** and **tls.key** must be encoded using Base64. For details, see **Base64 Encoding**.

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  # Certificate content, which must be encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  # Private key content, which must be encoded using Base64.
type: kubernetes.io/tls
```

- IngressTLS type

  The value of **tls.crt** and **tls.key** must be encoded using Base64. For details, see **Base64 Encoding**.

```
kind: Secret
apiVersion: v1
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  tls.crt: LS0tLS1CRU*****FURS0tLS0t  # Certificate content, which must be encoded using Base64.
  tls.key: LS0tLS1CRU*****VZLS0tLS0=  # Private key content, which must be encoded using Base64.
type: IngressTLS
```

Creating a Secret Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the Base64-encoded **cce-secret.yaml** file.

```
# echo -n "content to be encoded" | base64
******
```


vi cce-secret.yaml

The following YAML file uses the Opaque type as an example. For details about other types, see **Secret Resource File Configuration Example**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be encoded using Base64.
```

Step 3 Create a secret.

kubectl create -f cce-secret.yaml

You can query the secret after creation.

kubectl get secret -n default

----End

Related Operations

After creating a secret, you can update or delete it as described in **Table 13-6**.

NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

Table 13-6 Related Operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target secret resides to edit its YAML file.
Updating a secret	1. Select the name of the secret to be updated and click **Update**. 2. Modify the secret data. For more information, see **Table 13-5**. 3. Click **OK**.
Deleting a secret	Select the secret you want to delete and click **Delete**. Follow the prompts to delete the secret.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete** above the secret list. 3. Follow the prompts to delete the secrets.

Base64 Encoding

To Base64-encode a string, run the **echo -n *content to be encoded* | base64** command. The following is an example:


```
root@ubuntu:~# echo -n "content to be encoded" | base64
******
```

13.4 Using a Secret

After secrets are created, they can be mounted as data volumes or be exposed as environment variables to be used by a container in a pod.

NOTICE

Do not perform any operation on the following secrets. For details, see **Cluster Secrets**.

- Do not operate secrets under kube-system.
- Do not operate default-secret and paas.elb in any of the namespaces. The default-secret is used to pull the private image of SWR, and the paas.elb is used to connect the service in the namespace to the ELB service.

- **Configuring Environment Variables of a Workload**
- **Configuring the Data Volume of a Workload**

The following example shows how to use a secret.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: ******  #The value must be Base64-encoded.
  password: ******  #The value must be encoded using Base64.
```

NOTICE

- When a secret is used in a pod, the pod and secret must be in the same cluster and namespace.
- When a secret is updated, Kubernetes updates the data in the data volume at the same time.

  However, when a secret data volume mounted in **subPath** mode is updated, Kubernetes cannot automatically update the data in the data volume.

Configuring Environment Variables of a Workload

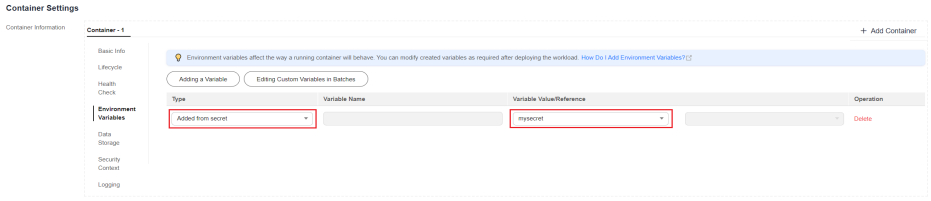
Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the dialog box displayed, click **Create Workload** in the upper right corner.

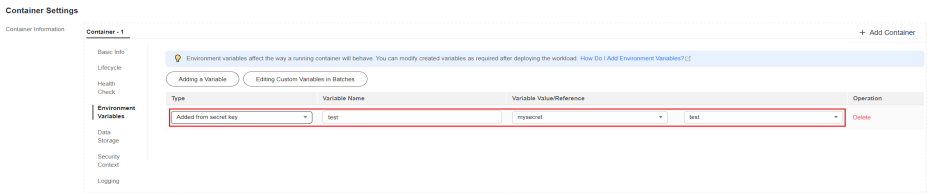
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **Added from secret**: Select a secret and import all keys in the secret as environment variables.



- **Added from secret key**: Import the value of a key in a secret as the value of an environment variable.
  - **Variable Name**: name of an environment variable in the workload. The name can be customized and is set to the key name selected in the secret by default.
  - **Variable Value/Reference**: Select a secret and the key to be imported. The corresponding value is imported as a workload environment variable.

  For example, after you import the value of **username** in secret **mysecret** as the value of workload environment variable **username**, an environment variable named **username** exists in the container.



Step 3 Configure other workload parameters and click **Create Workload**.

After the workload runs properly, **log in to the container** and run the following statement to check whether the secret has been set as an environment variable of the workload:

```
printenv username
```

If the output is the same as the content in the secret, the secret has been set as an environment variable of the workload.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-secret.yaml** and edit it.

vi nginx-secret.yaml

Content of the YAML file:

- **Added from secret**: To add all data in a secret to environment variables, use the **envFrom** parameter. The keys in the secret will become names of environment variables in a workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
```



```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        envFrom:              # Use envFrom to specify a secret to be referenced by environment variables.
          - secretRef:
              name: mysecret      # Name of the referenced secret.
      imagePullSecrets:
      - name: default-secret
```

- **Added from secret key**: When creating a workload, you can use a secret to set environment variables and use the **valueFrom** parameter to reference the key-value pair in the secret separately.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        env:                          # Set the environment variable in the workload.
        - name: SECRET_USERNAME          # Name of the environment variable in the workload.
          valueFrom:                  # Use valueFrom to specify a secret to be referenced by environment variables.
            secretKeyRef:
              name: mysecret      # Name of the referenced secret.
              key: username       # Key in the referenced secret.
        - name: SECRET_PASSWORD           # Add multiple environment variables to import them at the same time.
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
      imagePullSecrets:
      - name: default-secret
```

Step 3 Create a workload.

kubectl apply -f nginx-secret.yaml

Step 4 View the environment variables in the pod.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep nginx-secret
   ```

   Expected output:

   ```
   nginx-secret-***  1/1     Running   0          2m18s
   ```

2. Run the following command to view the environment variables in the pod:

   ```
   kubectl exec nginx-secret-*** -- printenv SPECIAL_USERNAME SPECIAL_PASSWORD
   ```


If the output is the same as the content in the secret, the secret has been set as an environment variable of the workload.

----End

Configuring the Data Volume of a Workload

You can mount a secret as a volume to the specified container path. Contents in a secret are user-defined. Before that, create a secret. For details, see **Creating a Secret**.

Using the CCE console

Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab. Click **Create Workload** in the upper right corner.

When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **Secret** from the drop-down list.

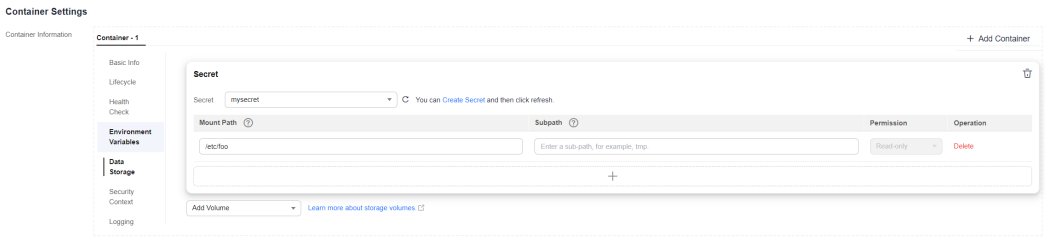
Step 3 Select parameters for mounting a secret volume, as shown in **Table 13-7**.

Table 13-7 Mounting a secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created beforehand. For details, see **Creating a Secret**.
Mount Path	Enter a mount path. After the secret volume is mounted, a secret file with the key as the file name and value as the file content is generated in the mount path of the container. This parameter specifies a container path to which a data volume will be mounted. Do not mount the volume to a system directory such as **/** or **/var/run**. This may lead to container errors. Mount the volume to an empty directory. If the directory is not empty, ensure that there are no files that affect container startup. Otherwise, the files will be replaced, leading to container startup failures or workload creation failures. **NOTICE** If the container is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

Parameter	Description
Subpath	Enter a subpath of the mount path. • A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path will be used by default. • The subpath can be the key and value of a secret. If the subpath is a key-value pair that does not exist, the data import does not take effect. • The data imported by specifying a subpath will not be updated along with the secret updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 13-2 Mounting a secret to a workload data volume



Step 4 After the configuration, click **Create Workload**.

After the workload runs properly, the **username** and **password** files will be generated in the **/etc/foo** directory in this example. The contents of the files are secret values.

Access the container and run the following statement to view the **username** or **password** file in the container:

```
cat /etc/foo/username
```

The expected output is the same as the content in the secret.

----End

Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-secret.yaml** and edit it.

vi nginx-secret.yaml

In the following example, the username and password in the **mysecret** secret are saved in the **/etc/foo** directory as files.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
```



```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
       - name: foo
          mountPath: /etc/foo          # Mount to the /etc/foo directory.
          readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # Name of the referenced secret.
```

You can also use the **items** field to control the mapping path of secret keys. For example, store username in the **/etc/foo/my-group/my-username** directory in the container.

NOTE

- If you use the **items** field to specify the mapping path of the secret keys, the keys that are not specified will not be created as files. For example, if the **password** key in the following example is not specified, the file will not be created.
- If you want to use all keys in a secret, you must list all keys in the **items** field.
- All keys listed in the **items** field must exist in the corresponding secret. Otherwise, the volume is not created.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-secret
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-secret
  template:
    metadata:
      labels:
        app: nginx-secret
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        volumeMounts:
       - name: foo
          mountPath: /etc/foo          # Mount to the /etc/foo directory.
          readOnly: true
    volumes:
    - name: foo
      secret:
        secretName: mysecret      # Name of the referenced secret.
        items:
        - key: username      # Name of the referenced key.
          path: my-group/my-username    # Mapping path of the secret key
```

Step 3 Create a workload.

kubectl apply -f nginx-secret.yaml

Step 4 After the workload runs properly, the **username** and **password** files will be generated in the **/etc/foo** directory.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep nginx-secret
   ```
   Expected output:
   ```
   nginx-secret-***  1/1     Running   0          2m18s
   ```
2. Run the following command to view the **username** or **password** file in the pod:
   ```
   kubectl exec nginx-secret-*** -- cat /etc/foo/username
   ```
   The expected output is the same as the content in the secret.

----End

13.5 Cluster Secrets

By default, CCE creates the following secrets in each namespace:

- default-secret
- paas.elb
- default-token-*xxxxx* (*xxxxx* is a random number.)

The functions of these secrets are described as follows.

default-secret

The type of **default-secret** is **kubernetes.io/dockerconfigjson**. The data is the credential for logging in to the SWR image repository and is used to pull images from SWR. To pull an image from SWR when creating a workload on CCE, set **imagePullSecrets** to **default-secret**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
  imagePullSecrets:
  - name: default-secret
```

The data of **default-secret** is updated periodically, and the current data will expire after a certain period of time. You can run the **describe** command to view the expiration time in default-secret.

NOTICE

Use default-secret directly instead of copying the secret content to create a new one. The credential in the copied secret will expire and the image cannot be pulled.

```
$ kubectl describe secret default-secret
Name:         default-secret
Namespace:    default
Labels:       secret-generated-by=cce
Annotations:  temporary-ak-sk-expires-at: 2021-11-26 20:55:31.380909 +0000 UTC

Type:  kubernetes.io/dockerconfigjson

Data
====
.dockerconfigjson:  347 bytes
```

paas.elb

The **paas.elb** data stores a temporary AK/SK that is used when a node is created or a load balancer is automatically created. The **paas.elb** data is updated periodically and has a specific time limit before it expires.

In practice, you will not directly use **paas.elb**. Do not delete it, as doing so will result in the failure of creating a node or load balancer.

default-token-xxxxx

By default, Kubernetes creates a service account named **default** for each namespace. **default-token-*xxxxx*** is the key of the service account, and ***xxxxx*** is a random number.

NOTE

In clusters v1.25 or later, a secret is not created automatically for each ServiceAccount. For details, see **Service Account Token Security Improvement**.

```
$ kubectl get sa
NAME     SECRETS   AGE
default  1         30d
$ kubectl describe sa default
Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   default-token-xxxxx
Tokens:              default-token-xxxxx
Events:              <none>
```


14 Add-ons

14.1 Overview

CCE provides multiple types of add-ons to extend cluster functions and meet feature requirements. You can install add-ons as required.

NOTICE

CCE uses Helm charts to deploy add-ons. To modify or upgrade an add-on, perform operations on the **Add-ons** page or use open add-on management APIs. Do not directly modify resources related to add-ons in the background. Otherwise, add-on exceptions or other unexpected problems may occur.

Add-on pods are prioritized over service pods. When cluster resources are limited, the add-on pods can use resources that would otherwise be allocated to service pods. This may result in the eviction of service pods.

Scheduling and Elasticity Add-ons

Add-on Name	Description
Volcano Scheduler	This add-on is a scheduler for general-purpose, high-performance computing such as job scheduling, heterogeneous chip management, and job running management, serving end users through computing frameworks for different industries such as AI, big data, gene sequencing, and rendering.
CCE Cluster Autoscaler	This add-on scales in or out the workload nodes in a cluster based on pod scheduling status and resource usage.
CCE Advanced HPA	This add-on is developed by CCE. It can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.

Add-on Name	Description
CCE Cloud Bursting Engine for CCI	This add-on is an implementation of the open-source Virtual Kubelet project. When momentary traffic spikes occur in a CCE cluster, it scales pods of Deployments, StatefulSets, jobs, and cron jobs to CCI, eliminating the overhead of the cluster.

Cloud Native Observability Add-ons

Add-on Name	Description
Cloud Native Cluster Monitoring	This add-on includes the Prometheus-operator and Prometheus components and provides easy-to-use, end-to-end Kubernetes cluster monitoring. This add-on allows the monitoring data to be interconnected with Monitoring Center so that you can view monitoring data and configure alarms there.
Cloud Native Log Collection	This add-on is developed based on Fluent Bit and OpenTelemetry for collecting logs. It supports CRD-based log collection policies, and collects and forwards standard output logs, container file logs, node logs, and Kubernetes events in a cluster.
CCE Node Problem Detector	This add-on monitors abnormal events of cluster nodes and connects to a third-party monitoring platform. It is a daemon running on each node. It collects node issues from different daemons and reports them to the API server. It can run as a DaemonSet or a daemon.
CCE Network Metrics Exporter	This add-on monitors and manages container network traffic. It collects how many IPv4 packets and bytes are received and sent (including those sent to the Internet) and allows you to obtain pod labels. It supports multiple monitoring tasks, allows you to select monitoring metrics, and uses a PodSelector to select monitoring backends. The monitoring information has been adapted to Prometheus. You can call the Prometheus API to view monitoring data.
Kubernetes Metrics Server	This add-on is an aggregator for monitoring data of core cluster resources.
Grafana	This add-on is an open source visualized data monitoring platform. It provides you with various charts and panels for real-time monitoring, analysis, and visualization of various metrics and data sources.
Prometheus (EOM)	This add-on is an open-source system monitoring and alerting framework. CCE allows you to quickly install Prometheus as an add-on.

Cloud Native Heterogeneous Computing Add-ons

Add-on Name	Description
CCE AI Suite (NVIDIA GPU)	This add-on supports and manages GPUs in containers. Only NVIDIA drivers are supported.
CCE AI Suite (Ascend NPU)	This add-on supports and manages NPUs in containers.

Container Network Add-ons

Add-on Name	Description
CoreDNS	This add-on is a DNS server that provides domain name resolution for Kubernetes clusters through a chain add-on.
NGINX Ingress Controller	This add-on forwards application data such as the data of virtual hosts, load balancers, SSL proxy, and HTTP routing for Services that can be directly accessed outside a cluster.
NodeLocal DNSCache	This add-on functions as a DaemonSet to run the DNS cache proxy on each cluster node to improve cluster DNS performance.

Container Storage Add-on

Add-on Name	Description
CCE Container Storage (Everest)	This add-on is a cloud native container storage system, which enables clusters of Kubernetes v1.15.6 or later to use cloud storage through the Container Storage Interface (CSI).

Container Security Add-ons

Add-on Name	Description
CCE Secrets Manager for DEW	This add-on is used to interconnect with Data Encryption Workshop (DEW), which allows you to mount secrets stored outside a cluster (DEW for storing sensitive information) to pods. In this way, sensitive information can be decoupled from the cluster environment, which prevents information leakage caused by program hardcoding or plaintext configuration.
Container Image Signature Verification	This add-on allows you to verify the digital signature of an image file, ensuring its integrity and authenticity. It effectively prevents tampering or malicious code implantation, ensuring user security.

Other Add-ons

Add-on Name	Description
Kubernetes Dashboard	This add-on is a general-purpose, web-based UI for Kubernetes clusters and integrates all commands that can be used in the command-line interface (CLI). It allows users to manage applications running in a cluster and troubleshoot faults, as well as manage the cluster itself.
OpenKruise	This add-on is a Kubernetes-based extension suite that automates cloud native application tasks like deployment, release, O&M, and availability protection.
Gatekeeper	This add-on is a customizable cloud native policy controller based on Open Policy Agent (OPA). It helps enhance policy execution and governance and provides more security policy rules that comply with Kubernetes application scenarios in clusters.
Kubernetes Web Terminal (EOM)	This add-on allows you to use kubectl on a web UI. It can connect to Linux by using WebSocket through a browser and provides APIs for integration into independent systems. It can be directly used as a service to obtain information through the configuration management database (CMDB) and log in to the server.

Add-on Lifecycle

An add-on lifecycle involves all the statuses of the add-on from installation to uninstallation.

Table 14-1 Add-on statuses

Status	Attribute	Description
Running	Stable state	The add-on is running properly, all add-on instances are deployed properly, and the add-on can be used properly.
Partially ready	Stable state	The add-on is running properly, but some add-on instances are not properly deployed. In this state, the add-on functions may be unavailable.
Unavailable	Stable state	The add-on malfunctions, and all add-on instances are not properly deployed.

Status	Attribute	Description
Installing	Intermediate state	The add-on is being deployed. If all instances cannot be scheduled due to incorrect add-on configuration or insufficient resources, the system sets the add-on status to **Unavailable** 10 minutes later.
Installation failed	Stable state	Install add-on failed. Uninstall it and try again.
Upgrading	Intermediate state	The add-on is being upgraded.
Upgrade failed	Stable state	Upgrade add-on failed. Upgrade it again, or uninstall it and try again.
Rolling back	Intermediate state	The add-on is rolling back.
Rollback failed	Stable state	The add-on rollback failed. Retry the rollback, or uninstall it and try again.
Deleting	Intermediate state	The add-on is being deleted. If this state stays for a long time, an exception occurred.
Deletion failed	Stable state	Delete add-on failed. Try again.
Unknown	Stable state	No add-on chart found.

NOTE

When an add-on is in an intermediate state such as **Installing** or **Deleting**, you are not allowed to edit or uninstall the add-on.

If the add-on status is unknown and the returned **status.Reason** is "don't install the addon in this cluster", the secret associated with the Helm release of the add-on in the cluster is typically deleted by mistake. In this case, uninstall the add-on and reinstall it with the same configurations.

Related Operations

You can perform the operations listed in **Table 14-2** on the **Add-ons** page.

Table 14-2 Related operations

Operation	Description	Procedure
Install	Install a specified add-on.	1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**. 2. Click **Install** under the target add-on. Each add-on has different configuration parameters. For details, see the corresponding chapter. 3. Click **OK**.
Upgrade	Upgrade an add-on to the new version.	1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**. 2. If an add-on can be upgraded, the **Upgrade** button is displayed under it. Click **Upgrade**. Each add-on has different configuration parameters. For details, see the corresponding chapter. 3. Click **OK**.
Edit	Edit add-on parameters.	1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**. 2. Click **Edit** under the target add-on. Each add-on has different configuration parameters. For details, see the corresponding chapter. 3. Click **OK**.
Uninstall	Uninstall an add-on from the cluster.	1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**. 2. Click **Uninstall** under the target add-on. 3. In the displayed dialog box, click **Yes**. This operation cannot be undone.

Operation	Description	Procedure
Roll back	Roll back an add-on to the source version. **NOTE** • This function is used to roll back an upgraded add-on to the source version, not to undo the editing of add-on parameters. • An add-on cannot be rolled back repeatedly.	1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**. 2. If an add-on can be rolled back, the **Roll Back** button is displayed under it. Click **Roll Back**. 3. In the displayed dialog box, click **Yes**.

NOTE

Rollback is supported by the following add-ons of certain versions:

- CoreDNS: 1.25.11 and later versions
- CCE Container Storage (Everest): 2.1.19 and later versions
- CCE Cluster Autoscaler
  - v1.21 clusters: v1.21.22 and later versions
  - v1.23 clusters: v1.23.24 and later versions
  - v1.25 clusters: v1.25.14 and later versions
- Volcano Scheduler: 1.11.4 and later versions
- CCE Node Problem Detector: 1.18.22 and later versions

14.2 Scheduling and Elasticity Add-ons

14.2.1 Volcano Scheduler

Introduction

Volcano is a batch processing platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities.

Volcano provides general computing capabilities such as high-performance job scheduling, heterogeneous chip management, and job running management. It accesses the computing frameworks for various industries such as AI, big data, gene, and rendering and schedules up to 1000 pods per second for end users, greatly improving scheduling efficiency and resource utilization.

Volcano provides job scheduling, job management, and queue management for computing applications. Its main features are as follows:

- Diverse computing frameworks, such as TensorFlow, MPI, and Spark, can run on Kubernetes in containers. Common APIs for batch computing jobs through CRD, various plugins, and advanced job lifecycle management are provided.
- Advanced scheduling capabilities are provided for batch computing and high-performance computing scenarios, including group scheduling, preemptive priority scheduling, packing, resource reservation, and task topology.
- Queues can be effectively managed for scheduling jobs. Complex job scheduling capabilities such as queue priority and multi-level queues are supported.

Volcano has been open-sourced in GitHub at **https://github.com/volcano-sh/volcano**.

Install and configure the Volcano add-on in CCE clusters. For details, see **Volcano Scheduling**.

NOTE

When using Volcano as a scheduler, use it to schedule all workloads in the cluster. This prevents resource scheduling conflicts caused by simultaneous working of multiple schedulers.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Volcano Scheduler** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, the system will configure the number of pods and resource quotas for the add-on based on the preset specifications. You can see the configurations on the console.
- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

  The resource quotas of the volcano-admission component are irrelevant to the number of cluster nodes and pods. You can retain the default value. The resource quotas of volcano-controller and volcano-scheduler are related to the number of cluster nodes and pods. The recommended values are as follows:

  - If the number of nodes is less than 100, retain the default configuration. The requested vCPUs are 500m, and the limit is 2000m. The requested memory is 500 MiB, and the limit is 2000 MiB.
  - If the number of nodes is greater than 100, increase the requested vCPUs by 500m and the requested memory by 1000 MiB each time 100 nodes (10,000 pods) are added. Increase the vCPU limit by 1500m and the memory limit by 1000 MiB.

NOTE

Recommended formula for calculating the requested value:

- Requested vCPUs: Calculate the number of target nodes multiplied by the number of target pods, perform interpolation search based on the number of nodes in the cluster multiplied by the number of target pods in **Table 14-3**, and round up the request value and limit value that are closest to the specifications.

  For example, for 2000 nodes and 20,000 pods, Number of target nodes x Number of target pods = 40 million, which is close to the specification of 700/70,000 (Number of cluster nodes x Number of pods = 49 million). According to the following table, set the requested vCPUs to 4000m and the limit value to 5500m.

- Requested memory: It is recommended that 2.4 GiB memory be allocated to every 1000 nodes and 1 GiB memory be allocated to every 10,000 pods. The requested memory is the sum of these two values. (The obtained value may be different from the recommended value in **Table 14-3**. You can use either of them.)

  Requested memory = Number of target nodes/1000 x 2.4 GiB + Number of target pods/10,000 x 1 GiB

  For example, for 2000 nodes and 20,000 pods, the requested memory is 6.8 GiB (2000/1000 x 2.4 GiB + 20,000/10,000 x 1 GiB).

Table 14-3 Recommended requested resources and resource limits for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
50/5000	500	2000	500	2000
100/10000	1000	2500	1500	2500
200/20000	1500	3000	2500	3500
300/30000	2000	3500	3500	4500
400/40000	2500	4000	4500	5500
500/50000	3000	4500	5500	6500
600/60000	3500	5000	6500	7500
700/70000	4000	5500	7500	8500

Step 3 Configure the extended functions supported by the add-on.

- **Descheduling**: After this function is enabled, the volcano-descheduler component is automatically deployed. The scheduler will evict and reschedule pods that do not meet your policy configuration requirements. This helps to balance cluster load and reduce resource fragmentation. For details, see **Descheduling**.
- **Hybrid Service Deployment**: After this function is enabled, the volcano-agent component is automatically deployed in the node pool with the hybrid

deployment capability enabled, which improves resource utilization by ensuring node QoS, enabling CPU bursts, and allowing for dynamic resource oversubscription. This helps to reduce resource usage costs. For details, see **Enabling Cloud Native Hybrid Deployment**.

- **NUMA Topology Scheduling**: After this function is enabled, the resource-exporter component is automatically deployed. The scheduler will schedule pods in NUMA affinity mode, which enhances the performance of high-performance training jobs. For details, see **NUMA Affinity Scheduling**.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-4 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.

Parameter	Description
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

After the add-on is installed, you can choose **Settings** in the navigation pane, switch to the **Scheduling** tab, select **Volcano scheduler**, and find the corresponding expert mode. You can customize advanced scheduling policies based on actual service scenarios. The following is an example:

```
colocation_enable: ''
default_scheduler_conf:
  actions: 'allocate, backfill, preempt'
  tiers:
    - plugins:
        - name: 'priority'
        - name: 'gang'
        - name: 'conformance'
        - name: 'lifecycle'
          arguments:
            lifecycle.MaxGrade: 10
            lifecycle.MaxScore: 200.0
            lifecycle.SaturatedTresh: 1.0
            lifecycle.WindowSize: 10
    - plugins:
        - name: 'drf'
        - name: 'predicates'
        - name: 'nodeorder'
    - plugins:
        - name: 'cce-gpu-topology-predicate'
        - name: 'cce-gpu-topology-priority'
        - name: 'cce-gpu'
    - plugins:
        - name: 'nodelocalvolume'
        - name: 'nodeemptydirvolume'
        - name: 'nodeCSIscheduling'
        - name: 'networkresource'
tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 60
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 60
```


Table 14-5 Advanced Volcano configuration parameters

Plugin	Function	Description	Demonstration
colocation_enable	Whether to enable hybrid deployment.	Value: • **true**: hybrid enabled • **false**: hybrid disabled	None
default_scheduler_conf	Used to schedule pods. It consists of a series of actions and plugins and features high scalability. You can specify and implement actions and plugins based on your requirements.	It consists of actions and tiers. • **actions**: defines the types and sequence of actions to be executed by the scheduler. • **tiers**: configures the plugin list.	None

Plugin	Function	Description	Demonstration
actions	Actions to be executed in each scheduling phase. The configured action sequence is the scheduler execution sequence. For details, see **Actions**. The scheduler traverses all jobs to be scheduled and performs actions such as enqueue, allocate, preempt, and backfill in the configured sequence to find the most appropriate node for each job.	The following options are supported: • **enqueue**: uses a series of filtering algorithms to filter out tasks to be scheduled and sends them to the queue to wait for scheduling. After this action, the task status changes from **pending** to **inqueue**. • **allocate**: selects the most suitable node based on a series of pre-selection and selection algorithms. • **preempt**: performs preemption scheduling for tasks with higher priorities in the same queue based on priority rules. • **backfill**: schedules pending tasks as much as possible to maximize the utilization of node resources.	`actions: 'allocate, backfill, preempt'` **NOTE** When configuring **actions**, use either **preempt** or **enqueue**.
plugins	Implementation details of algorithms in actions based on different scenarios. For details, see **Plugins**.	For details, see **Table 14-6**.	None

tolerations	Tolerance of the add-on to node taints.	By default, the add-on can run on nodes with the **node.kubernetes.io/not-ready** or **node.kubernetes.io/unreachable** taint and the taint effect value is **NoExecute**, but it'll be evicted in 60 seconds.	`tolerations:`
			`- effect: NoExecute`
			`key: node.kubernetes.io/not-ready`
			`operator: Exists`
			`tolerationSeconds: 60`
			`- effect: NoExecute`
			`key: node.kubernetes.io/unreachable`
			`operator: Exists`
			`tolerationSeconds: 60`

Table 14-6 Supported plugins

Plugin	Function	Description	Demonstration
binpack	Schedule pods to nodes with high resource usage (not allocating pods to light-loaded nodes) to reduce resource fragments.	**arguments**: • **binpack.weight**: weight of the binpack plugin. • **binpack.cpu**: ratio of CPUs to all resources. The parameter value defaults to **1**. • **binpack.memory**: ratio of memory resources to all resources. The parameter value defaults to **1**. • **binpack.resources**: other custom resource types requested by the pod, for example, **nvidia.com/gpu**. Multiple types can be configured and be separated by commas (,). • **binpack.resources.***<your_resource>*: weight of your custom resource in all resources. Multiple types of resources can be added. *<your_resource>* indicates the resource type defined in **binpack.resources**, for example, **binpack.resources.nvidia.com/gpu**.	- plugins: - **name: binpack** arguments: binpack.weight: 10 binpack.cpu: 1 binpack.memory: 1 binpack.resources: nvidia.com/gpu, example.com/foo binpack.resources.nvidia.com/gpu: 2 binpack.resources.example.com/foo: 3
conformance	Prevent key pods, such as the pods in the **kube-system** namespace from being preempted.	None	- plugins: - name: 'priority' - name: 'gang' enablePreemptable: false - **name: 'conformance'**

Plugin	Function	Description	Demonstration
lifecycle	By collecting statistics on service scaling rules, pods with similar lifecycles are preferentially scheduled to the same node. With the horizontal scaling capability of the Autoscaler, resources can be quickly scaled in and released, reducing costs and improving resource utilization. 1. Collects statistics on the lifecycle of pods in the service load and schedules pods with similar lifecycles to the same node. 2. For a cluster configured with an automatic scaling policy, adjust the scale-in annotation of the node to preferentially scale in the node with low usage.	**arguments**: • **lifecycle.WindowSize** : The value is an integer greater than or equal to 1 and defaults to **10**. Record the number of times that the number of replicas changes. If the load changes regularly and periodically, decrease the value. If the load changes irregularly and the number of replicas changes frequently, increase the value. If the value is too large, the learning period is prolonged and too many events are recorded. • **lifecycle.MaxGrade**: The value is an integer greater than or equal to 3 and defaults to **3**. It indicates levels of replicas. For example, if the value is set to **3**, the replicas are classified into three levels. If the load changes regularly and periodically, decrease the value. If the load changes irregularly, increase the value. Setting an excessively small value may result in inaccurate lifecycle forecasts. • **lifecycle.MaxScore**: float64 floating point number. The value must be greater than or equal to 50.0. The default value is **200.0**.	- plugins: - name: priority - name: gang enablePreemptable: false - name: conformance - **name: lifecycle** arguments: lifecycle.MaxGrade: 3 lifecycle.MaxScore: 200.0 lifecycle.SaturatedTresh: 0.8 lifecycle.WindowSize: 10 **NOTE** • For nodes that do not want to be scaled in, manually mark them as long-period nodes and add the annotation **volcano.sh/long-lifecycle-node: true** to them. For an unmarked node, the lifecycle plugin automatically marks the node based on the lifecycle of the load on the node. • The default value of **MaxScore** is **200.0**, which is twice the weight of other plugins. When the lifecycle plugin does not have obvious effect or conflicts with other plugins, disable other plugins or increase the value of **MaxScore**. • After the scheduler is restarted, the lifecycle plugin needs to re-record the load change. The optimal scheduling effect can be achieved only after several periods of statistics are collected.

Plugin	Function	Description	Demonstration
		Maximum score (equivalent to the weight) of the lifecycle plugin. • **lifecycle.SaturatedTresh**: float64 floating point number. If the value is less than 0.5, use **0.5**. If the value is greater than 1, use **1**. The default value is **0.8**. Threshold for determining whether the node usage is too high. If the node usage exceeds the threshold, the scheduler preferentially schedules jobs to other nodes.	

Plugin	Function	Description	Demonstration
Gang	Consider a group of pods as a whole for resource allocation. This plugin checks whether the number of scheduled pods in a job meets the minimum requirements for running the job. If yes, all pods in the job will be scheduled. If no, the pods will not be scheduled. **NOTE** If a gang scheduling policy is used, if the remaining resources in the cluster are greater than or equal to half of the minimum number of resources for running a job but less than the minimum of resources for running the job, Autoscaler scale-outs will not be triggered.	• **enablePreemptable**: – **true**: Preemption enabled – **false**: Preemption not enabled • **enableJobStarving**: – **true**: Resources are preempted based on the **minAvailable** setting of jobs. – **false**: Resources are preempted based on job replicas. **NOTE** – The default value of **minAvailable** for Kubernetes-native workloads (such as Deployments) is **1**. It is a good practice to set **enableJobStarving** to **false**. – In AI and big data scenarios, you can specify the **minAvailable** value when creating a vcjob. It is a good practice to set **enableJobStarving** to **true**. – In Volcano versions earlier than v1.11.5, **enableJobStarving** is set to **true** by default. In Volcano versions later than v1.11.5, **enableJobStarving** is set to **false** by default.	- plugins: - name: priority **- name: gang** **enablePreemptable: false** **enableJobStarving: false** - name: conformance
priority	Schedule based on custom load priorities.	None	- plugins: **- name: priority** - name: gang enablePreemptable: false - name: conformance

Plugin	Function	Description	Demonstration

overcommit	Resources in a cluster are scheduled after being accumulated in a certain multiple to improve the workload enqueuing efficiency. If all workloads are Deployments, remove this plugin or set the raising factor to **2.0**.
	NOTE
	This plugin is supported in Volcano 1.6.5 and later versions.
		arguments:
		• **overcommit-factor**: inflation factor, which defaults to **1.2**.
			- plugins:
			  - **name: overcommit**
			  arguments:
			  overcommit-factor: 2.0

drf	The Dominant Resource Fairness (DRF) scheduling algorithm, which schedules jobs based on their dominant resource share. Jobs with a smaller resource share will be scheduled with a higher priority.	None	- plugins:
			  - **name: 'drf'**
			  - name: 'predicates'
			  - name: 'nodeorder'

Plugin	Function	Description	Demonstration

predicates	Determine whether a task is bound to a node by using a series of evaluation algorithms, such as node/pod affinity, taint tolerance, node repetition, volume limits, and volume zone matching.	None	- plugins:
			  - name: 'drf'
			  - **name: 'predicates'**
			  - name: 'nodeorder'

Plugin	Function	Description	Demonstration

nodeorder	A common algorithm for selecting nodes. Nodes are scored in simulated resource allocation to find the most suitable node for the current job.	Scoring parameters:
		• **nodeaffinity.weight**: Pods are scheduled based on node affinity. This parameter defaults to **2**.
		• **podaffinity.weight**: Pods are scheduled based on pod affinity. This parameter defaults to **2**.
		• **leastrequested.weight**: Pods are scheduled to the node with the least requested resources. This parameter defaults to **1**.
		• **balancedresource.weight**: Pods are scheduled to the node with balanced resource allocation. This parameter defaults to **1**.
		• **mostrequested.weight**: Pods are scheduled to the node with the most requested resources. This parameter defaults to **0**.
		• **tainttoleration.weight**: Pods are scheduled to the node with a high taint tolerance. This parameter defaults to **3**.
		• **imagelocality.weight**: Pods are scheduled to the node where the required images exist. This parameter defaults to **1**.
		• **podtopologyspread.weight**: Pods are
			- plugins:
			  - **name: nodeorder**
			  arguments:
			  leastrequested.weight: 1
			  mostrequested.weight: 0
			  nodeaffinity.weight: 2
			  podaffinity.weight: 2
			  balancedresource.weight: 1
			  tainttoleration.weight: 3
			  imagelocality.weight: 1
			  podtopologyspread.weight: 2

Plugin	Function	Description	Demonstration
		scheduled based on the pod topology. This parameter defaults to **2**.	

cce-gpu-topology-predicate	GPU-topology scheduling preselection algorithm	None	- plugins:
			  - **name: 'cce-gpu-topology-predicate'**
			  - name: 'cce-gpu-topology-priority'
			  - name: 'cce-gpu'

cce-gpu-topology-priority	GPU-topology scheduling priority algorithm	None	- plugins:
			  - name: 'cce-gpu-topology-predicate'
			  - **name: 'cce-gpu-topology-priority'**
			  - name: 'cce-gpu'

Plugin	Function	Description	Demonstration

cce-gpu	GPU resource allocation that supports decimal GPU configurations by working with the gpu add-on.
	NOTE
	• The plugin of version 1.10.5 or later does not support this add-on. Use xGPU instead.
	• The prerequisite for configuring decimal GPUs is that the GPU nodes in the cluster are in shared mode. For details about how to check whether GPU sharing is disabled in the cluster, see the **enable-gpu-share** parameter in **Modifying Cluster Configurations**.
		None	- plugins:
			  - name: 'cce-gpu-topology-predicate'
			  - name: 'cce-gpu-topology-priority'
			  - **name: 'cce-gpu'**

numa-aware	NUMA affinity scheduling. For details, see **NUMA Affinity Scheduling**.	**arguments**:
		• **weight**: weight of the numa-aware plugin
			- plugins:
			  - name: 'nodelocalvolume'
			  - name: 'nodeemptydirvolume'
			  - name: 'nodeCSIscheduling'
			  - name: 'networkresource'
			  arguments:
			  NetworkType: vpc-router
			  - **name: numa-aware**
			  arguments:
			  weight: 10

Plugin	Function	Description	Demonstration

network resource	The ENI requirement node can be preselected and filtered. The parameters are transferred by CCE and do not need to be manually configured.	**arguments**:
		• **NetworkType**: network type (**eni** or **vpc-router**)
			- plugins:
			  - name: 'nodelocalvolume'
			  - name: 'nodeemptydirvolume'
			  - name: 'nodeCSIscheduling'
			  - **name: 'networkresource'**
			  arguments:
			  NetworkType: vpc-router

nodelocalvolume	Filter out nodes that do not meet local volume requirements.	None	- plugins:
			  - **name: 'nodelocalvolume'**
			  - name: 'nodeemptydirvolume'
			  - name: 'nodeCSIscheduling'
			  - name: 'networkresource'

nodeemptydirvolume	Filter out nodes that do not meet the emptyDir requirements.	None	- plugins:
			  - name: 'nodelocalvolume'
			  - **name: 'nodeemptydirvolume'**
			  - name: 'nodeCSIscheduling'
			  - name: 'networkresource'

nodeCSIscheduling	Filter out nodes with malfunctional Everest.	None	- plugins:
			  - name: 'nodelocalvolume'
			  - name: 'nodeemptydirvolume'
			  - **name: 'nodeCSIscheduling'**
			  - name: 'networkresource'

----End

Components

Table 14-7 Add-on components

Component	Description	Resource Type
volcano-scheduler	Schedule pods.	Deployment
volcano-controller	Synchronize CRDs.	Deployment
volcano-admission	Webhook server, which verifies and modifies resources such as pods and jobs	Deployment
volcano-agent	Cloud native hybrid agent, which is used for node QoS assurance, CPU burst, and dynamic resource oversubscription	DaemonSet

Component	Description	Resource Type
resource-exporter	Report the NUMA topology information of nodes.	DaemonSet
volcano-descheduler	Reschedule pods in a cluster. After the rescheduling capability is enabled, pods will be automatically deployed on nodes.	Deployment
volcano-recommender	Generate recommendations for CPU and memory requests based on the historical CPU and memory usage of a container.	Deployment
volcano-recommender-prometheus-adapter	Collect historical CPU and memory metrics of containers from Prometheus.	Deployment

Modifying the volcano-scheduler Configurations Using the Console

volcano-scheduler is the component responsible for pod scheduling. It consists of a series of actions and plugins. Actions should be executed in every step. Plugins provide the action algorithm details in different scenarios. volcano-scheduler is highly scalable. You can specify and implement actions and plugins based on your requirements.

After the add-on is installed, you can choose **Settings** in the navigation pane, switch to the **Scheduling** tab, and configure the basic scheduling capabilities. You can also use the expert mode of the Volcano scheduler to customize advanced scheduling policies based on service scenarios.

This section describes how to configure volcano-scheduler.

NOTE

Only Volcano of v1.7.1 and later support this function.

Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Settings** and click the **Scheduling** tab. In the **Select Cluster Scheduler** area, select **Volcano scheduler**, find the expert mode, and click **Refresh**.

- Using **resource_exporter**:

```
...
  "default_scheduler_conf": {
```



```
    "actions": "allocate, backfill, preempt",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          },
          {
            "name": "numa-aware" # add this also enable resource_exporter
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
      }
    ]
  },
...
```

After this function is enabled, you can use the functions of both numa-aware and resource_exporter.

Collecting Prometheus Metrics

volcano-scheduler exposes Prometheus metrics through port 8080. You can build a Prometheus collector to identify and obtain volcano-scheduler scheduling metrics from **http://{{*volcano-schedulerPodIP*}}:{{*volcano-schedulerPodPort*}}/metrics**.

NOTE

Prometheus metrics can be exposed only by the Volcano add-on of version 1.8.5 or later.

Table 14-8 Key metrics

Metric	Type	Description	Label
e2e_scheduling_latency_milliseconds	Histogram	E2E scheduling latency (ms) (scheduling algorithm + binding)	None
e2e_job_scheduling_latency_milliseconds	Histogram	E2E job scheduling latency (ms)	None
e2e_job_scheduling_duration	Gauge	E2E job scheduling duration	labels=["job_name", "queue", "job_namespace"]
plugin_scheduling_latency_microseconds	Histogram	Add-on scheduling latency (μs)	labels=["plugin", "OnSession"]
action_scheduling_latency_microseconds	Histogram	Action scheduling latency (μs)	labels=["action"]
task_scheduling_latency_milliseconds	Histogram	Task scheduling latency (ms)	None
schedule_attempts_total	Counter	Number of pod scheduling attempts. **unschedulable** indicates that the pods cannot be scheduled, and **error** indicates that the internal scheduler is faulty.	labels=["result"]
pod_preemption_victims	Gauge	Number of selected preemption victims	None
total_preemption_attempts	Counter	Total number of preemption attempts in a cluster	None
unschedule_task_count	Gauge	Number of unschedulable tasks	labels=["job_id"]
unschedule_job_count	Gauge	Number of unschedulable jobs	None

Metric	Type	Description	Label
job_retry_counts	Counter	Number of job retries	labels=["job_id"]

Uninstalling the Volcano Add-on

After the add-on is uninstalled, all custom Volcano resources (**Table 14-9**) will be deleted, including the created resources. Reinstalling the add-on will not inherit or restore the tasks before the uninstallation. It is a good practice to uninstall the Volcano add-on only when no custom Volcano resources are being used in the cluster.

Table 14-9 Custom Volcano resources

Item	API Group	API Version	Resource Level
Command	bus.volcano.sh	v1alpha1	Namespaced
Job	batch.volcano.sh	v1alpha1	Namespaced
Numatopology	nodeinfo.volcano.sh	v1alpha1	Cluster
PodGroup	scheduling.volcano.sh	v1beta1	Namespaced
Queue	scheduling.volcano.sh	v1beta1	Cluster
BalancerPolicyTemplate	autoscaling.volcano.sh	v1alpha1	Cluster
Balancer	autoscaling.volcano.sh	v1alpha1	Cluster

NOTE

BalancerPolicyTemplate and Balancer resources are created only after the application scaling priority policies are enabled. For details, see **Application Scaling Priority Policies**.

Related Operations

- **Dynamic Resource Oversubscription**
- **NUMA Affinity Scheduling**

Change History

NOTICE

It is a good practice to upgrade Volcano to the latest version that is supported by the cluster.

Table 14-10 Release history

Add-on Version	Supported Cluster Version	New Feature
1.16.8	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	• CCE clusters 1.31 are supported. • Optimized the resource scheduling capability of supernodes.
1.15.10	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• Supported tor packing and scheduling. • Optimized the NPU dual-die topology scheduling.
1.15.8	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Supported the NPU dual-die affinity scheduling.
1.15.6	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Resources can be oversubscribed based on pod profiling.

Add-on Version	Supported Cluster Version	New Feature
1.14.11	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• The supernode resource scheduling model (HyperJob) is available. • Supported supernode affinity scheduling. • Supported Kubernetes 1.30.
1.13.7	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• ENIs can be pre-bound for scheduling. • The resource oversubscription ratio can be customized.
1.13.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• Supported scale-in of customized resources based on node priorities. • Optimized the association between preemption and node scale-out.
1.13.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Optimized scheduler memory usage.
1.12.18	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• CCE clusters 1.29 are supported. • The preemption function is enabled by default.
1.12.1	v1.19.16 v1.21 v1.23 v1.25 v1.27 v1.28	Optimized application auto scaling performance.

Add-on Version	Supported Cluster Version	New Feature
1.11.21	v1.19.16 v1.21 v1.23 v1.25 v1.27 v1.28	• Supported Kubernetes 1.28. • Supported load-aware scheduling. • Updated image OS to HCE 2.0. • Optimized CSI resource preemption. • Optimized load-aware rescheduling. • Optimized preemption in hybrid deployment scenarios.
1.11.6	v1.19.16 v1.21 v1.23 v1.25 v1.27	• Supported Kubernetes 1.27. • Supported rescheduling. • Supported affinity scheduling of nodes in the node pool. • Optimized the scheduling performance.
1.10.7	v1.19.16 v1.21 v1.23 v1.25	Fixes the issue that the local PV add-on fails to calculate the number of pods pre-bound to the node.
1.10.5	v1.19.16 v1.21 v1.23 v1.25	• The volcano agent supports resource oversubscription. • Adds the verification admission for GPUs. The value of **nvidia.com/gpu** must be less than 1 or a positive integer, and the value of **volcano.sh/gpu-core.percentage** must be less than 100 and a multiple of 5. • Fixes the issue that pod scheduling is slow after PVC binding fails. • Fixes the issue that newly added pods cannot run when there are terminating pods on a node for a long time. • Fixes the issue that volcano restarts when creating or mounting PVCs to pods.

Add-on Version	Supported Cluster Version	New Feature
1.9.1	v1.19.16 v1.21 v1.23 v1.25	• Fixes the issue that the counting pipeline pod of the networkresource add-on occupies supplementary network interfaces (Sub-ENI). • Fixes the issue where the binpack add-on scores nodes with insufficient resources. • Fixes the issue of processing resources in the pod with unknown end status. • Optimizes event output. • Supports HA deployment by default.
1.7.2	v1.19.16 v1.21 v1.23 v1.25	• Adapts to clusters 1.25. • Improves scheduling performance of volcano.
1.7.1	v1.19.16 v1.21 v1.23 v1.25	Adapts to clusters 1.25.
1.4.7	v1.15 v1.17 v1.19 v1.21	Deletes the pod status Undetermined to adapt to cluster Autoscaler.
1.4.5	v1.17 v1.19 v1.21	Changes the deployment mode of volcano-scheduler from **statefulset** to **deployment**, and fixes the issue that pods cannot be automatically migrated when the node is abnormal.
1.4.2	v1.15 v1.17 v1.19 v1.21	• Resolves the issue that cross-GPU allocation fails. • Supports the updated EAS API.

Add-on Version	Supported Cluster Version	New Feature
1.3.7	v1.15 v1.17 v1.19 v1.21	• Supports hybrid deployment of online and offline jobs and resource oversubscription. • Optimizes the scheduling throughput for clusters. • Fixes the issue where the scheduler panics in certain scenarios. • Fixes the issue that the volumes.secret verification of the volcano job in the CCE clusters 1.15 fails. • Fixes the issue that jobs fail to be scheduled when volumes are mounted.
1.3.3	v1.15 v1.17 v1.19 v1.21	Fixes the scheduler crash caused by GPU exceptions and the privileged init container admission failure.
1.3.1	v1.15 v1.17 v1.19	• Upgrades the volcano framework to the latest version. • Supported Kubernetes 1.19. • Adds the numa-aware add-on. • Fixes the deployment scaling issue in the multi-queue scenario. • Adjusts the algorithm add-on enabled by default.
1.2.5	v1.15 v1.17 v1.19	• Fixes the OutOfcpu issue in some scenarios. • Fixes the issue that pods cannot be scheduled when some capabilities are set for a queue. • Makes the log time of the volcano component consistent with the system time. • Fixes the issue of preemption between multiple queues. • Fixes the issue that the result of the ioaware add-on does not meet the expectation in some extreme scenarios. • Supports hybrid clusters.

Add-on Version	Supported Cluster Version	New Feature
1.2.3	v1.15 v1.17 v1.19	• Fixes the training task OOM issue caused by insufficient precision. • Fixes the GPU scheduling issue in CCE v1.15 and later versions. Rolling upgrade of CCE versions during task distribution is not supported. • Fixes the issue where the queue status is unknown in certain scenarios. • Fixes the issue where a panic occurs when a PVC is mounted to a job in a specific scenario. • Fixes the issue that decimals cannot be configured for GPU jobs. • Adds the ioaware add-on. • Adds the ring controller.

14.2.2 CCE Cluster Autoscaler

Introduction

The CCE Cluster Autoscaler add-on is built on the Autoscaler component of the community. It can automatically adjust the number of cluster nodes based on the resource needs of applications, optimizing resource utilization and performance. Autoscaler is the main controller in Kubernetes. It can automatically scale nodes in or out based on resource requirements. When there are not enough node resources to schedule pods in a cluster, Autoscaler adds more nodes with additional resources for those pods. Furthermore, if the resource utilization of the added nodes is low, Autoscaler will automatically remove them. For details about how to implement node auto scaling, see **Creating a Node Scaling Policy**.

Open source community: **https://github.com/kubernetes/autoscaler**

How the Add-on Works

Autoscaler controls auto scale-out and scale-in.

- **Auto scale-out**

  You can choose either of the following methods:

  – If a pod cannot be scheduled due to insufficient resources of worker nodes, CCE will add more nodes to the cluster. The new nodes have the same resource quotas as those configured for the node pools that the new nodes are in.

    Auto scale-out will be performed when:

    - Node resources are insufficient.
    - No node affinity policy is set in the scheduling configurations of the pod. If the pod is configured affinity for a node, the system will not automatically add more nodes in the cluster. For details about how to configure node affinity policies, see **Configuring Node Affinity Scheduling (nodeAffinity)**.
  - When the cluster meets the node scaling policy, cluster scale-out is also triggered. For details, see **Creating a Node Scaling Policy**.

  NOTE

  The add-on follows the "No Less, No More" policy. For example, if three cores are required for creating a pod and the system supports four-core and eight-core nodes, Autoscaler will preferentially create a four-core node.

- **Auto scale-in**

  When a cluster node is idle for a period of time (10 minutes by default), cluster scale-in is triggered, and the node is automatically deleted. However, a node cannot be deleted from a cluster if the following pods exist:

  - Pods that do not meet specific requirements set in Pod Disruption Budgets (**PodDisruptionBudget**)
  - Pods that cannot be scheduled to other nodes due to constraints such as affinity and anti-affinity policies
  - Pods that have the **cluster-autoscaler.kubernetes.io/safe-to-evict: 'false'** annotation
  - Pods (except those created by DaemonSets in the kube-system namespace) that exist in the kube-system namespace on the node
  - Pods that are not created by the controller (Deployment/ReplicaSet/job/StatefulSet)

  NOTE

  When a node meets the scale-in conditions, Autoscaler adds the **DeletionCandidateOfClusterAutoscaler** taint to the node in advance to prevent pods from being scheduled to the node. After the Autoscaler add-on is uninstalled, if the taint still exists on the node, manually delete it.

Notes and Constraints

- There must be enough resources in the cluster during the add-on installation.
- This add-on supports **VMs** only.
- The default node pool does not support auto scaling. For details, see **Description of DefaultPool**.
- Node scale-in will cause PVC/PV data loss for the **local PVs** associated with the node. These PVCs and PVs cannot be restored or used again. In a node scale-in, a pod that uses the local PV will be evicted from the node. A new pod will be created, but it remains in a pending state because the label of the PVC bound to it conflicts with the node label.
- When CCE Cluster Autoscaler is used, some taints or annotations may affect auto scaling. Therefore, do not use the following taints or annotations in clusters:

- **ignore-taint.cluster-autoscaler.kubernetes.io**: The taint works on nodes. Kubernetes-native Autoscaler supports protection against abnormal scale-outs and periodically evaluates the proportion of available nodes in the cluster. When the proportion of non-ready nodes exceeds 45%, protection will be triggered. In this case, all nodes with the **ignore-taint.cluster-autoscaler.kubernetes.io** taint in the cluster are filtered out from the Autoscaler template and recorded as non-ready nodes, which affect cluster scaling.
- **cluster-autoscaler.kubernetes.io/enable-ds-eviction**: The annotation works on pods, which determines whether DaemonSet pods can be evicted by Autoscaler. For details, see **Well-Known Labels, Annotations and Taints**.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Cluster Autoscaler** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

There are three types of **preset specifications** based on the cluster scale. You can select one as required. The system will configure the number of pods and resource quotas for the add-on based on the selected preset specifications. You can see the configurations on the console.

Step 3 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-11 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 4 After the configuration is complete, click **Install**.

----End

Components

Table 14-12 Add-on components

Component	Description	Resource Type
Autoscaler	Auto scaling for Kubernetes clusters	Deployment

Change History

Table 14-13 Release history for the add-on adapted to clusters 1.31

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.31.8	v1.31	CCE clusters 1.31 are supported.	1.31.1

Table 14-14 Release history for the add-on adapted to clusters 1.30

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.30.46	v1.30	Fixed some issues.	1.30.1
1.30.19	v1.30	Fixed some issues.	1.30.1
1.30.18	v1.30	Fixed some issues.	1.30.1
1.30.15	v1.30	• Clusters 1.30 are supported. • Included the name of the target node pool to the reported event.	1.30.1

Table 14-15 Release history for the add-on adapted to clusters 1.29

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.29.79	v1.29	Fixed some issues.	1.29.1
1.29.54	v1.29	Fixed some issues.	1.29.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.29.53	v1.29	Fixed some issues.	1.29.1
1.29.50	v1.29	Included the name of the target node pool to the reported event.	1.29.1
1.29.17	v1.29	Optimized events.	1.29.1
1.29.13	v1.29	Clusters 1.29 are supported.	1.29.1

Table 14-16 Release history for the add-on adapted to clusters 1.28

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.28.118	v1.28	Fixed some issues.	1.28.1
1.28.93	v1.28	Fixed some issues.	1.28.1
1.28.92	v1.28	Fixed some issues.	1.28.1
1.28.91	v1.28	Fixed some issues.	1.28.1
1.28.88	v1.28	Included the name of the target node pool to the reported event.	1.28.1
1.28.55	v1.28	Optimized events.	1.28.1
1.28.51	v1.28	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.28.1
1.28.22	v1.28	Fixed some issues.	1.28.1
1.28.20	v1.28	Fixed some issues.	1.28.1
1.28.17	v1.28	Fixed the issue that scale-in cannot be performed when there are custom pod controllers in a cluster.	1.28.1

Table 14-17 Release history for the add-on adapted to clusters 1.27

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.27.149	v1.27	Fixed some issues.	1.27.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.27.124	v1.27	Fixed some issues.	1.27.1
1.27.123	v1.27	Fixed some issues.	1.27.1
1.27.122	v1.27	Fixed some issues.	1.27.1
1.27.119	v1.27	Included the name of the target node pool to the reported event.	1.27.1
1.27.88	v1.27	Optimized events.	1.27.1
1.27.84	v1.27	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.27.1
1.27.55	v1.27	Fixed some issues.	1.27.1
1.27.53	v1.27	Fixed some issues.	1.27.1
1.27.51	v1.27	Fixed some issues.	1.27.1
1.27.14	v1.27	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected **PreferNoSchedule** taint issues.	1.27.1

Table 14-18 Release history for the add-on adapted to clusters 1.25

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.179	v1.25	Fixed some issues.	1.25.0
1.25.154	v1.25	Fixed some issues.	1.25.0
1.25.153	v1.25	Fixed some issues.	1.25.0
1.25.152	v1.25	Included the name of the target node pool to the reported event.	1.25.0
1.25.120	v1.25	Optimized events.	1.25.0
1.25.116	v1.25	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.25.0
1.25.88	v1.25	Fixed some issues.	1.25.0
1.25.86	v1.25	Fixed some issues.	1.25.0
1.25.84	v1.25	Fixed some issues.	1.25.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.46	v1.25	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected **PreferNoSchedule** taint issues.	**1.25.0**
1.25.34	v1.25	• Optimized the method of identifying GPUs and NPUs. • Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.	**1.25.0**
1.25.21	v1.25	• Fixed the issue that the autoscaler's least-waste is disabled by default. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	**1.25.0**
1.25.11	v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.	**1.25.0**

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.7	v1.25	• CCE clusters 1.25 are supported. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.25.0

Table 14-19 Release history for the add-on adapted to clusters 1.23

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.159	v1.23	Fixed some issues.	1.23.0
1.23.157	v1.23	Fixed some issues.	1.23.0
1.23.156	v1.23	Included the name of the target node pool to the reported event.	1.23.0
1.23.125	v1.23	Optimized events.	1.23.0
1.23.121	1.23	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.23.0
1.23.95	v1.23	Fixed some issues.	1.23.0
1.23.93	v1.23	Fixed some issues.	1.23.0
1.23.91	v1.23	Fixed some issues.	1.23.0
1.23.54	v1.23	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected **PreferNoSchedule** taint issues.	1.23.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.44	v1.23	• Optimized the method of identifying GPUs and NPUs. • Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.	1.23.0
1.23.31	v1.23	• Fixed the issue that the autoscaler's least-waste is disabled by default. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	1.23.0
1.23.21	v1.23	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.	1.23.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.17	v1.23	• Supported NPUs and security containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.23.0
1.23.10	v1.23	• Optimized logging. • Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.	1.23.0
1.23.9	v1.23	Added the **nodenetworkconfigs.crd.yangtse.cni** resource object permission.	1.23.0
1.23.8	v1.23	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	1.23.0
1.23.7	v1.23	-	1.23.0
1.23.3	v1.23	CCE clusters 1.23 are supported.	1.23.0

Table 14-20 Release history for the add-on adapted to clusters 1.21

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.114	v1.21	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.21.0
1.21.89	v1.21	Fixed some issues.	1.21.0
1.21.87	v1.21	Fixed some issues.	1.21.0
1.21.86	v1.21	Fixed the issue that the node pool auto scaling cannot meet expectations after AZ topology constraints are configured for nodes.	1.21.0
1.21.51	v1.21	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected **PreferNoSchedule** taint issues.	1.21.0
1.21.43	v1.21	• Optimized the method of identifying GPUs and NPUs. • Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.	1.21.0
1.21.29	v1.21	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	1.21.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.20	v1.21	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.	1.21.0
1.21.16	v1.21	• Supported NPUs and security containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.21.0
1.21.9	v1.21	• Optimized logging. • Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.	1.21.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.8	v1.21	Added the **nodenetworkconfigs.crd.yangtse.cni** resource object permission.	1.21.0
1.21.6	v1.21	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	1.21.0
1.21.4	v1.21	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	1.21.0
1.21.2	v1.21	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	1.21.0
1.21.1	v1.21	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	1.21.0

Table 14-21 Release history for the add-on adapted to clusters 1.19

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.76	v1.19	• Optimized the method of identifying GPUs and NPUs. • Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.	1.19.0
1.19.56	v1.19	Fixed the scale-in failure of nodes of different specifications in the same node pool and unexpected **PreferNoSchedule** taint issues.	1.19.0
1.19.48	v1.19	• Optimized the method of identifying GPUs and NPUs. • Used the remaining node quota of a cluster for the extra nodes that are beyond the cluster scale.	1.19.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.35	v1.19	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	1.19.0
1.19.27	v1.19	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • Fixed the issue that the number of node pools cannot be restored when scaling group resources are insufficient.	1.19.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.22	v1.19	• Supported NPUs and security containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.19.0
1.19.14	v1.19	• Optimized logging. • Supported scale-in waiting so that operations such as data dump can be performed before a node is deleted.	1.19.0
1.19.13	v1.19	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	1.19.0
1.19.12	v1.19	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	1.19.0
1.19.11	v1.19	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	1.19.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.9	v1.19	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	1.19.0
1.19.8	v1.19	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	1.19.0
1.19.7	v1.19	Regular upgrade of add-on dependencies	1.19.0
1.19.6	v1.19	Fixed the issue that repeated scale-out is triggered when taints are asynchronously updated.	1.19.0
1.19.3	v1.19	Supports scheduled scaling policies based on the total number of nodes, CPU limit, and memory limit. Fixes other functional defects.	1.19.0

Table 14-22 Release history for the add-on adapted to clusters 1.17

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.27	v1.17	• Optimized logging. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Fixed the issue that taints on newly added nodes are overwritten. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled.	1.17.0
1.17.22	v1.17	Optimized logging.	1.17.0
1.17.21	v1.17	Fixed the issue that scale-out fails when the number of nodes to be added at a time exceeds the upper limit in periodic scale-outs.	1.17.0
1.17.19	v1.17	Fixed the issue that authentication fails due to incorrect signature in the add-on request retries.	1.17.0
1.17.17	v1.17	Fixed the issue that auto scaling may be blocked due to a failure in deleting an unregistered node.	1.17.0
1.17.16	v1.17	Fixed the issue that the node pool modification in the existing periodic auto scaling rule does not take effect.	1.17.0
1.17.15	v1.17	Unified resource specification configuration unit.	1.17.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.14	v1.17	Fixed the issue that repeated scale-out is triggered when taints are asynchronously updated.	1.17.0
1.17.8	v1.17	Fixed bugs.	1.17.0
1.17.7	v1.17	Added log content and fixed bugs.	1.17.0
1.17.5	v1.17	Supported clusters 1.17 and allowed scaling events to be displayed on the CCE console.	1.17.0
1.17.2	v1.17	Clusters 1.17 are supported.	1.17.0

Change History

Table 14-23 Release history for the add-on adapted to clusters 1.31

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.31.8	v1.31	CCE clusters 1.31 are supported.	1.31.1

Table 14-24 Release history for the add-on adapted to clusters 1.28

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.28.118	v1.28	Included the name of the target node pool to the reported event.	1.28.1
1.28.88	v1.28	The reported event includes the name of the target node pool.	1.28.1
1.28.55	v1.28	None	1.28.1
1.28.20	v1.28	None	1.28.1

Table 14-25 Release history for the add-on adapted to clusters 1.25

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.179	v1.25	Included the name of the target node pool to the reported event.	1.25.0
1.25.152	v1.25	The reported event includes the name of the target node pool.	1.25.0
1.25.120	v1.25	None	1.25.0
1.25.86	v1.25	• Fixed the scale-in failure of nodes of different flavors in the same node pool and unexpected **PreferNoSchedule** taint issues.	1.25.0
1.25.21	v1.25	• CCE clusters 1.25 are supported. • Fixed the issue that the autoscaler's least-waste is disabled by default. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	1.25.0

Table 14-26 Release history for the add-on adapted to clusters 1.23

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.156	v1.23	The reported event includes the name of the target node pool.	1.23.0
1.23.125	v1.23	None	1.23.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.31	v1.23	• Fixed the issue that the autoscaler's least-waste is disabled by default. • Fixed the issue that the node pool cannot be switched to another pool for scaling out after a scale-out failure and the add-on has to restart. • The default taint tolerance duration is changed to 60s. • Fixed the issue that scale-out is still triggered after the scale-out rule is disabled.	1.23.0
1.23.17	v1.23	• CCE clusters 1.23 are supported. • Supported NPUs and secure containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.23.0

Table 14-27 Release history for the add-on adapted to clusters 1.21

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.21.16	v1.21	• Supported NPUs and secure containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.21.0

Table 14-28 Release history for the add-on adapted to clusters 1.19

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.22	v1.19	• Supported NPUs and secure containers. • Supported node scaling policies without a step. • Fixed a bug so that deleted node pools are automatically removed. • Supported scheduling by priority. • Supported the emptyDir scheduling policy. • Fixed a bug so that scale-in can be triggered on the nodes whose capacity is lower than the scale-in threshold when the node scaling policy is disabled. • Modified the memory request and limit of a customized flavor. • Enabled to report an event indicating that scaling cannot be performed in a node pool with auto scaling disabled. • Fixed the bug that NPU node scale-out is triggered again during scale-out.	1.19.0

14.2.3 CCE Advanced HPA

CCE Advanced HPA (formerly cce-hpa-controller) is a CCE-developed add-on, which can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.

After installing this add-on, you can create scheduled CronHPA and CustomedHPA policies. For details, see **Creating a Scheduled CronHPA Policy** and **Creating a CustomedHPA Policy**.

Main Functions

- Scaling can be performed based on the percentage of the current number of pods.
- The minimum scaling step can be set.

- Different scaling operations can be performed based on the actual metric values.

Notes and Constraints

- If the add-on version is earlier than 1.2.11, the **Prometheus** add-on must be installed. If the add-on version is 1.2.11 or later, the add-ons that can provide metrics API must be installed. You can select one of the following add-ons based on your cluster version and requirements.
  - **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
  - **Cloud Native Cluster Monitoring**: available only in clusters of v1.17 or later.
    - Auto scaling based on basic resource metrics: Prometheus needs to be registered as a metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**.
    - Auto scaling based on custom metrics: Custom metrics need to be aggregated to the Kubernetes API server. For details, see **Creating an HPA Policy Using Custom Metrics**.
  - **Prometheus (EOM)**: Prometheus needs to be registered as a metrics API. For details, see **Providing Resource Metrics Through the Metrics API**. This add-on supports only clusters of v1.21 or earlier.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Advanced HPA** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, the add-on specifications will be automatically configured based on the recommended values by CCE. These values are suitable for most scenarios and can be viewed on the console.
- If you selected **Custom**, you can modify the number of replicas, CPUs, and memory of each add-on component as required.

  NOTE

  Replicas: HA is not possible with just one replica, so one replica is used only for verification. In commercial scenarios, you can configure multiple replicas based on the cluster specifications.

  CPU Quota and **Memory Quota**: The resource quotas of a component are affected by how many containers and scaling policies in a cluster. For typical situations, it is recommended that you configure 500m CPU cores and 1,000 MiB of memory for every 5,000 containers in a cluster. As for scaling policies, 100m CPU cores and 500 MiB of memory should be configured for every 1,000 of them.

Step 3 Configure the add-on parameters.

- **AHPA Policy**: After this function is enabled, historical monitoring metrics can be used to predict the required number of replicas and scales accordingly. For details, see **Creating an AHPA Policy**.

To enable AHPA, make sure to install the Cloud Native Cluster Monitoring add-on and enable the function that reports monitoring data to AOM. For details, see **Cloud Native Cluster Monitoring**.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-29 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.

Parameter	Description
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-30 Add-on components

Component	Description	Resource Type
customedhpa-controller	CCE auto scaling component, which scales in or out Deployments based on metrics such as CPU usage and memory usage	Deployment

Change History

Table 14-31 Release history

Add-on Version	Supported Cluster Version	New Feature
1.5.21	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	• CCE clusters 1.31 are supported. • Supported intelligent scalability based on application trend prediction.

Add-on Version	Supported Cluster Version	New Feature
1.5.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	AHPA is available.
1.4.30	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.
1.4.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
1.4.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.
1.3.43	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.
1.3.42	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.3.14	v1.19 v1.21 v1.23 v1.25 v1.27	CCE clusters 1.27 are supported.
1.3.10	v1.19 v1.21 v1.23 v1.25	Periodic scaling is not affected by the cooldown period.
1.3.7	v1.19 v1.21 v1.23 v1.25	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.
1.3.3	v1.19 v1.21 v1.23 v1.25	• CCE clusters 1.25 are supported. • Allowed CronHPA to adjust the number of Deployment pods with the skip scenario supported.
1.3.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.12	v1.15 v1.17 v1.19 v1.21	Optimizes the add-on performance to reduce resource consumption.
1.2.11	v1.15 v1.17 v1.19 v1.21	• Enables the Kubernetes metrics API to obtain resource metrics. • Takes not-ready pods into consideration when calculating resource usage.
1.2.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.2.4	v1.15 v1.17 v1.19	• Regular upgrade of add-on dependencies • Allows custom add-on resource specifications.
1.2.3	v1.15 v1.17 v1.19	Supported ARM64 nodes.
1.2.2	v1.15 v1.17 v1.19	Enhanced the health check function.
1.2.1	v1.15 v1.17 v1.19	• CCE clusters 1.19 are supported. • Updated the add-on to a stable version.
1.1.3	v1.15 v1.17	Supported periodic scaling rules.

14.2.4 CCE Cloud Bursting Engine for CCI

The bursting add-on functions as a virtual kubelet to connect Kubernetes clusters to APIs of other platforms. This add-on is mainly used to extend Kubernetes APIs to serverless container services such as Huawei Cloud CCI.

With this add-on, you can schedule Deployments, StatefulSets, jobs, and CronJobs running in CCE clusters to **CCI** during peak hours. In this way, you can reduce consumption caused by cluster scaling.

Installing the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Install**.
5. Configure the add-on parameters.

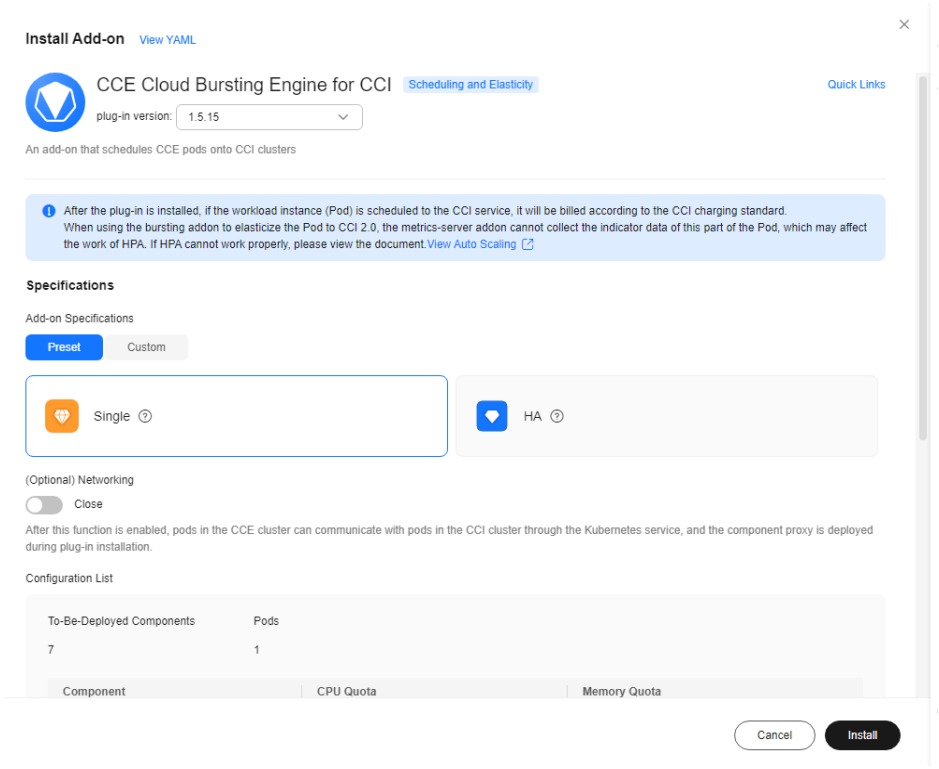


Table 14-32 Add-on parameters

Parameter	Description
Version	Add-on version. There is a mapping between add-on versions and CCE cluster versions. For more details, see "Change History" in **CCE Cloud Bursting Engine for CCI**.

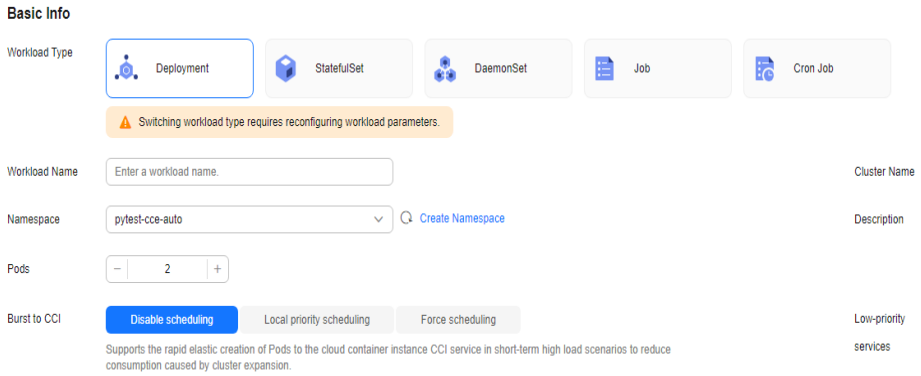
Parameter	Description

Specifications	Number of pods required for running the add-on.
	• If you select **Preset**, you can select **Single** or **HA**.
	• If you select **Custom**, you can modify the number of replicas, vCPUs, and memory of each add-on component as required.
	NOTE
	• The bursting add-on 1.5.2 or later uses more node resources. You need to reserve sufficient pods before upgrading the add-on.
	  • **Single** (only one pod for the add-on): There must be a node that has at least seven schedulable pods. If **Networking** is enabled, eight schedulable pods are required.
	  • **HA** (two pods for the add-on): There must be two nodes, each of which must have at least seven schedulable pods, a total of 14 schedulable pods. If **Networking** is enabled, eight schedulable pods are required on each node, a total of 16 schedulable pods.
	• The resource usage of the add-on varies depending on the workloads scaled to CCI. The pods, secrets, ConfigMaps, PVs, and PVCs requested by the services occupy VM resources. You are advised to evaluate the service usage and apply for VMs based on the following specifications: For 1,000 pods and 1,000 ConfigMaps (300 KB), nodes with 2 vCPUs and 4-GiB memory are recommended. For 2,000 pods and 2,000 ConfigMaps, nodes with 4 vCPUs and 8-GiB memory are recommended. For 4,000 pods and 4,000 ConfigMaps, nodes with 8 vCPUs and 16-GiB memory are recommended.

Networking	If this option is enabled, pods in the CCE cluster can communicate with pods in CCI through Services. The component proxy will be automatically deployed upon add-on installation. For details, see **Networking**.

Creating a Workload

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Workloads**.
4. Click **Create Workload**. For details, see **Creating a Workload**.
5. Specify basic information. Set **Burst to CCI** to **Force scheduling**. For more information about scheduling policies, see **Scheduling Pods to CCI**.



CAUTION

When you schedule a workload in a CCE cluster to CCI, TCP probes cannot be used for health check.

6. Configure the container parameters.
7. Click **Create Workload**.
8. On the **Workloads** page, click the name of the created workload to go to the workload details page.
9. View the node where the workload is running. If the workload is running on a CCI node, it has been scheduled to CCI.

Uninstalling the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

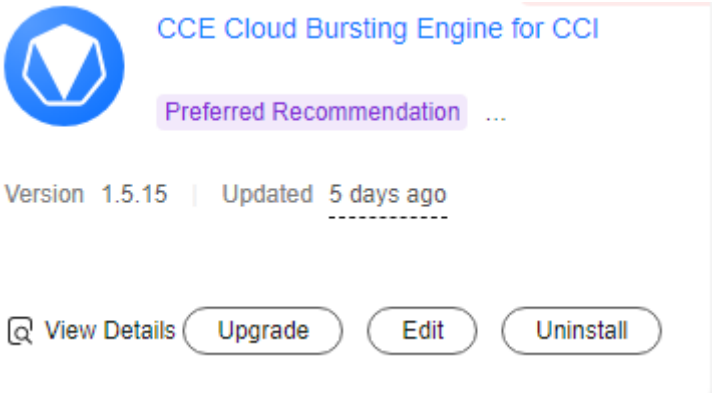


Table 14-33 Special scenarios for uninstalling the add-on

Scenario	Symptom	Description
There are no nodes in the CCE cluster that the bursting add-on needs to be uninstalled from.	Failed to uninstall the bursting add-on.	If the bursting add-on is uninstalled from the cluster, a job for clearing resources will be started in the cluster. To ensure that the job can be started, there is at least one node in the cluster that can be scheduled.
The CCE cluster is deleted, but the bursting add-on is not uninstalled.	There are residual resources in the namespace on CCI. If the resources are not free, additional expenditures will be generated.	The cluster is deleted, but the resource clearing job is not executed. You can manually clear the namespace and residual resources.

For more information about the bursting add-on, see **CCE Cloud Bursting Engine for CCI**.

Change History

Table 14-34 Release history

Add-on Version	Supported Cluster Version	New Feature
1.5.27	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.5.26	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Fixed some issues.
1.5.24	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Optimized some functions.
1.5.16	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Compacted pod CPU and memory resources.
1.5.8	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.
1.3.57	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.
1.3.48	v1.21 v1.23 v1.25 v1.27	• Clusters 1.25 and 1.27 are supported. • Supported JuiceFS.

Add-on Version	Supported Cluster Version	New Feature
1.3.25	v1.17 v1.19 v1.21 v1.23	• Supports Downward API volumes. • Supports Projected volumes. • Supports custom StorageClass.
1.3.19	v1.17 v1.19 v1.21 v1.23	Supports schedule profile.
1.3.7	v1.17 v1.19 v1.21 v1.23	Clusters 1.21 and 1.23 are supported.
1.2.12	v1.13 v1.15 v1.17 v1.19	• Adds some metrics. • Supports HPA and CustomedHPA. • Enables the hostPath in the pod that is scaled to CCI to be converted to other types of storage. • Fixes an issue that the Kubernetes dashboard cannot run on terminals.
1.2.5	v1.13 v1.15 v1.17 v1.19	• Automatically clears CCI resources that are no longer used by pods. • **Requests** and **Limits** can be set to different values. When CCI is scaled, the number of applied resources is subject to **Limits**. • Fixes the issue that the add-on fails to be uninstalled when the CCI namespace does not exist. • Adds the function of intercepting creation requests when the pod specifications exceed the CCI limit.

Add-on Version	Supported Cluster Version	New Feature
1.2.0	v1.13 v1.15 v1.17 v1.19	• Clusters 1.19 are supported. • Supported SFS and SFS Turbo storage. • Supported CronJobs. • Supported envFrom configuration. • Supports automatic logs dumping. • Shields TCPSocket health check. • Supports resource tags (pod-tag). • Improves performance and reliability. • Resolves some known issues.
1.0.5	v1.13 v1.15 v1.17	Clusters 1.17 are supported.

14.2.5 Vertical Pod Autoscaler

The CCE Vertical Pod Autoscaler add-on (VPA) supports vertical pod autoscaling. It automates the adjustment of CPU and memory resource requests for pods based on their historical resource usage.

For details about the open-source Vertical Pod Autoscaler, see **autoscaler**.

Overview

VPA collects and analyzes resource metrics for each container, adjusts the requested resources based on actual usage, and maintains the ratio of resource limit to request before and after the adjustment. VPA can increase or decrease CPU and memory resources as needed.

The rules are as follows:

- VPA generates the CPU and memory resource recommendations using the data collected by the Metrics API.
- VPA, in theory, recommends a minimum of 250 MiB of memory for each pod and 250 MiB divided by the number of containers in the pod for each container. It also recommends a minimum of 25m vCPUs for each pod and 25m divided by the number of containers in the pod for each container.

  When setting up a VPA, you can establish the minimum and maximum number of elastic resources in containers by configuring the **containerPolicies** field.

- If a container has both resource request and limit configured, VPA will provide resource recommendations. It will adjust the requested resources of the container to match the recommendations and generate recommended resource limit based on the ratio of the original resource request to the limit set during the container's initial creation.

  Assume that the requested vCPUs of a container are 100m and the limit is 200m (with a ratio of 1:2). If VPA recommends a requested vCPU of 80m, the container's vCPU limit will be 160m.
- VPA ensures its recommendations align with other resource limits. If the VPA recommendations conflict with a resource limit, they will not be adjusted to fit the limit. This means that the resource configuration suggested by VPA may go beyond other resource limits.

  Assume that the requested memory of a namespace cannot exceed 2 GiB. If VPA recommends a high memory configuration for a pod in that namespace, the total memory requested by the namespace may exceed 2 GiB after the pod's resource configuration is updated. This means the pod will not be scheduled.

Prerequisites

- The cluster version must be v1.25 or later.
- An add-on that provides Metrics API has been installed in the cluster. You can select one of the following add-ons based on your service requirements:
  - **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage.
  - **Cloud Native Cluster Monitoring**: provides basic resource usage metrics using Prometheus. You need to register Prometheus as a service of Metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Vertical Pod Autoscaler** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small**, **Medium**, or **Large** based on the number of pods in the cluster. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.
- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-35 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 4 Click **Install**.

----End

Components

Table 14-36 Add-on components

Component	Description	Resource Type
vpa-admission-controller	Change the resource requests for a pod to the recommendations generated by the VPA when the pod is created.	Deployment
vpa-recommender	Collect the actual CPU and memory metrics of a pod and generate resource recommendations for the requested resources based on the actual resource usage.	Deployment
vpa-updater	Evict a pod whose actual resource requests are different from the VPA recommendations and trigger pod recreation so that the resources recommendations can apply to the new pod.	Deployment

Change History

Table 14-37 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.0.6	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	1.1.2
1.0.4	v1.25 v1.27 v1.28 v1.29 v1.30	Vertical Pod Autoscaler (VPC) is now available.	1.1.2

14.3 Cloud Native Observability Add-ons

14.3.1 Cloud Native Cluster Monitoring

Introduction

Cloud Native Cluster Monitoring (formerly kube-prometheus-stack) uses Prometheus-operator and Prometheus and provides easy-to-use, end-to-end Kubernetes cluster monitoring.

This add-on allows the monitoring data to be interconnected with CIA so that you can view monitoring data and configure alarms there.

Open source community: **https://github.com/prometheus/prometheus**

Notes and Constraints

- By default, the kube-state-metrics component of the add-on does not collect labels and annotations of Kubernetes resources. To collect these labels and annotations, manually enable the collection function in the startup parameters and check whether the corresponding metrics are added to the collection trustlist of ServiceMonitor named **kube-state-metrics**. For details, see **Collecting All Labels and Annotations of a Pod**.
- In 3.8.0 and later versions, component metrics in the kube-system and monitoring namespaces are not collected by default. If you have workloads in the two namespaces, use **Pod Monitor** or **Service Monitor** to collect these metrics.
- In 3.8.0 and later versions, etcd-server, kube-controller, kube-scheduler, autoscaler, fluent-bit, volcano-agent, volcano-scheduler and otel-collector metrics are not collected by default. Enable the collection as required.

  To enable this function, on the **ConfigMaps and Secrets** page, expand the dropdown list of **Namespace**, and select **monitoring**. Locate the row that contains the configuration item named **persistent-user-config**, and click **Edit YAML** in the operation column. Remove the **serviceMonitorDisable** or **podMonitorDisable** configuration in the **customSettings** field as required or set the configuration to an empty array.

  ```
  ...
    customSettings:
      podMonitorDisable: []
      serviceMonitorDisable: []
  ```

- In versions later than 3.9.0, Grafana is removed from Cloud Native Cluster Monitoring and has become an independent Grafana add-on. Therefore, before rolling Cloud Native Cluster Monitoring from a version later than 3.9.0 to a version earlier than 3.9.0, uninstall Grafana from the add-on.

Permissions

The node-exporter component of the Cloud Native Cluster Monitoring add-on needs to read the Docker info data from the **/var/run/docker.sock** directory on the host for monitoring the Docker disk space.

The following permission is required for running node-exporter:

- cap_dac_override: reads the Docker info data.

Installing the Add-on

NOTE

The Cloud Native Cluster Monitoring add-on automatically selects a deployment mode based on **Data Storage Configuration**. This is supported by Cloud Native Cluster Monitoring 3.7.1 or later.

- Original agent mode: Disable **Local data storage** and enable at least one of **Report Monitoring Data to AOM** and **Report Monitoring Data to a Third-Party Platform**.
- Original server mode: Enable **Local data storage** and **Report Monitoring Data to AOM** or **Report Monitoring Data to a Third-Party Platform**.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Cloud Native Cluster Monitoring** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, enable at least one item in the **Data Storage Configuration** area.

- **Reporting Monitoring Data to AOM**: Report Prometheus data to AOM. After this function is enabled, you can select the corresponding AOM instance. The collected basic metrics are free of charge. Custom metrics are charged by AOM. For details, see **Product Pricing Details**. To interconnect with AOM, you must have certain permissions. Only **Huawei Cloud accounts, HUAWEI IDs, and users in the admin user group** can perform this operation.
- **Reporting Monitoring Data to a Third-Party Monitoring Platform**: To report Prometheus data to a third-party monitoring system, you need to enter the address and token of the third-party monitoring system and determine whether to skip certificate authentication.
- **Local data storage**: Select the type and size of a disk for storing monitoring data to store Prometheus data in PVCs in the cluster. **Storage volumes are not deleted along with the add-on.** If local data storage is enabled, all components will be deployed. For details, see **Components**.

  NOTE

  An available PVC named **pvc-prometheus-server** exists in namespace **monitoring** and will be used as the storage source.

Step 3 Configure the add-on **specifications** as needed.

- **Add-on Specifications**:
  - If you selected **Preset**, the system will configure the number of pods and resource quotas for the add-on based on the preset specifications. You can see the configurations on the console.
  - If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.
- **Prometheus HA**: The Prometheus-server, Prometheus-operator, thanos-query, custom-metrics-apiserver, alertmanager, and kube-state-metrics components are deployed in multi-instance mode in the cluster.
- **Number of collected shards** (available after **Local data storage** is disabled): When there is a lot of Prometheus data, you can configure this parameter to spread the data across a specific number of Prometheus instances. This will

help with storage and querying. Increasing the number of shards reduces the data volume carried by each shard. This can increase the upper limit of the metric collection throughput and cause more resources to be consumed. You are advised to increase shards when the cluster scale is large to improve the collection performance. In addition, you need to consider the impact of resource usage and optimize it based on specific scenarios.

- **Install Grafana**: Use Grafana to visualize monitoring data. Grafana creates a 5 GiB storage volume by default. Uninstalling the add-on **will not delete this volume**. The default username and password for the first login are **admin**. You will be asked to change the password immediately after login.

  In versions later than 3.9.0, Grafana is removed from Cloud Native Cluster Monitoring and has become an independent Grafana add-on. So this option is no longer displayed.

Step 4 Configure the add-on parameters.

- **Custom Metric Collection**: Application metrics are automatically collected in the form of service discovery. After this function is enabled, you need to add related configurations to the target application. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.
- **Collection Interval**: Configure the collection interval.
- **Data Retention** (available only after **Local data storage** is enabled): Configure how long the monitoring data can be kept.
- **node-exporter Listening Port**: This port uses the host network to listen to and expose metrics on the node for Prometheus collection. The default port number is 9100, but it can be modified if there is a conflict with an existing port.

Step 5 Click **Install**.

After the add-on is installed, you may need to perform the following operations:

- To use custom metrics to create an auto scaling policy, ensure that local data storage is enabled for the add-on and then take the following steps:
  a. Collect custom metrics reported by applications to Prometheus. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.
  b. Aggregate these custom metrics collected by Prometheus to the API server for the HPA policy to use. For details, see **Creating an HPA Policy Using Custom Metrics**.
- To provide system resource metrics (such as CPU and memory usage) for workload auto scaling using this add-on, ensure that local data storage is enabled for the add-on and then enable the Metrics API. For details, see **Providing Basic Resource Metrics Through the Metrics API**. After the configuration, you can use Prometheus to collect system resource metrics. (This operation is not recommended because it may conflict with the Kubernetes Metric Server add-on.)

----End

Components

All Kubernetes resources created during Cloud Native Cluster Monitoring add-on installation are created in the namespace named **monitoring**.

Table 14-38 Add-on components

Component	Description	Supported Deployment Mode	Resource Type
prometheusOperator (workload name: prometheus-operator)	Deploys and manages the Prometheus Server based on CustomResourceDefinitions (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.	All	Deployment
prometheus (workload name: prometheus-server)	A Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.	All	StatefulSet
alertmanager (workload name: alertmanager-alertmanager)	Alarm center of the add-on. It receives alarms sent by Prometheus and manages alarm information by deduplicating, grouping, and distributing.	Local data storage enabled	StatefulSet
thanosSidecar	Available only in HA mode. Runs with prometheus-server in the same pod to implement persistent storage of Prometheus metric data.	Local data storage enabled	Container
thanosQuery	Available only in HA mode. Entry for PromQL query when Prometheus is in HA scenarios. It can delete duplicate metrics from Store or Prometheus.	Local data storage enabled	Deployment
adapter (workload name: custom-metrics-apiserver)	Aggregates custom metrics to the native Kubernetes API Server.	Local data storage enabled	Deployment

Component	Description	Supported Deployment Mode	Resource Type
kubeStateMetrics (workload name: kube-state-metrics)	Converts the Prometheus metric data into a format that can be identified by Kubernetes APIs. By default, the kube-state-metrics component does not collect all labels and annotations of Kubernetes resources. To collect all labels and annotations, see **Collecting All Labels and Annotations of a Pod**. **NOTE** If the components run in multiple pods, only one pod provides metrics.	All	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	All	DaemonSet
grafana (workload name: grafana)	Visualizes monitoring data. Grafana creates a 5 GiB storage volume by default. Uninstalling the add-on will not delete this volume.	All	Deployment
clusterProblemDetector (workload name: cluster-problem-detector)	Monitors cluster exceptions.	Local data storage enabled	Deployment

Providing Basic Resource Metrics Through the Metrics API

NOTE

If local data storage is enabled for the Cloud Native Cluster Monitoring add-on, basic resource metrics can only be provided through Metrics API.

Resource metrics of containers and nodes, such as CPU and memory usage, can be obtained through the Kubernetes Metrics API. Resource metrics can be directly accessed, for example, by using the **kubectl top** command, or used by HPA or CustomedHPA policies for auto scaling.

The add-on can provide the Kubernetes Metrics API that is disabled by default. To enable the Metrics API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
```



```
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If information similar to the following is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                       CPU(cores)   MEMORY(bytes)
......
custom-metrics-apiserver-d4f556ff9-l2j2m               38m          44Mi
......
```

NOTICE

To uninstall the add-on, run the following kubectl command and delete the APIService object. Otherwise, residual APIService resources may prevent the installation of the Kubernetes Metrics Server add-on.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Creating an HPA Policy Using Custom Metrics

HPA policies can only be used when Cloud Native Cluster Monitoring is deployed with local data storage enabled. You can configure custom metrics required by HPA policies in the **user-adapter-config** ConfigMap.

NOTICE

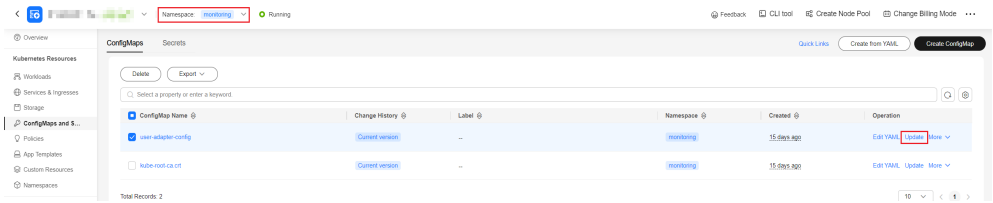
To use Prometheus to monitor custom metrics, the application needs to provide a metric monitoring API. For details, see **Prometheus Monitoring Data Collection**.

In this section, the nginx metric (nginx_connections_accepted) in **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring** is used as an example.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **ConfigMaps and Secrets**.

Step 2 Click the **ConfigMaps** tab, select the **monitoring** namespace, locate the row containing **user-adapter-config** (or **adapter-config**), and click **Update**.

Figure 14-1 Updating a ConfigMap



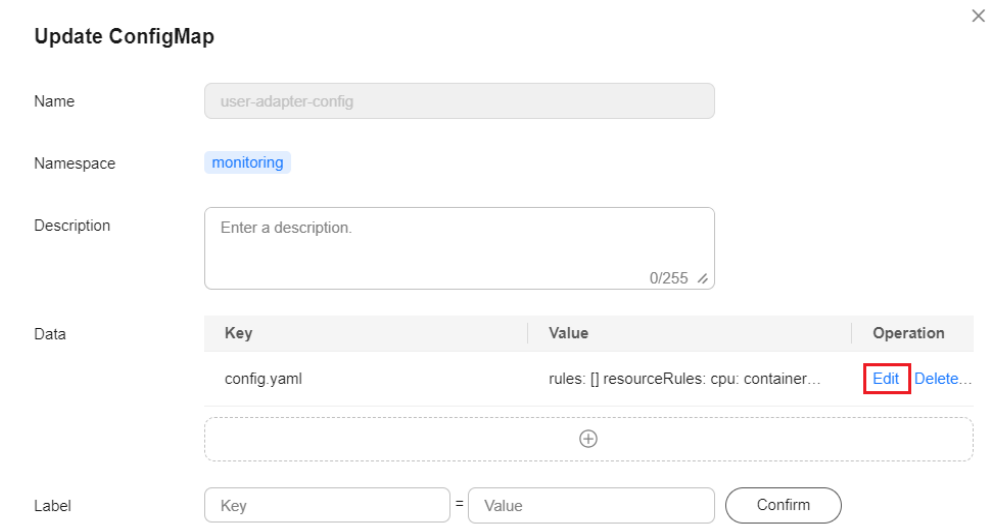
Step 3 In **Data**, click **Edit** for the **config.yaml** file to add a custom metric collection rule under the **rules** field. Click **OK**.

You can add multiple collection rules by adding multiple configurations under the **rules** field. For details, see **Metrics Discovery and Presentation Configuration**.

Example custom metric rule:

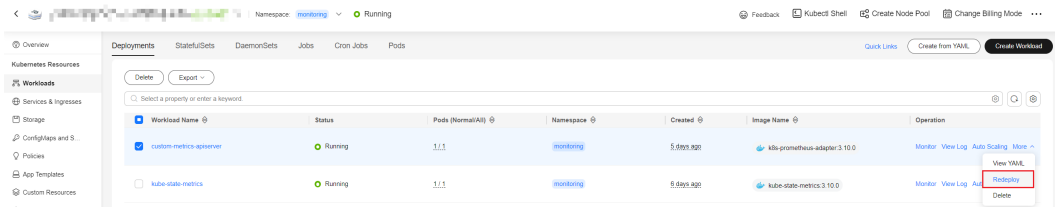
```
rules:
# Match the metric whose name is nginx_connections_accepted. The metric name must be confirmed. Otherwise, the HPA controller cannot get the metric.
- seriesQuery: '{__name__=~"nginx_connections_accepted",container!="POD",namespace!="",pod!=""}'
  resources:
    # Specify pod and namespace resources.
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  name:
    #Use nginx_connections_accepted"
    matches: "nginx_connections_accepted"
    #Use nginx_connections_accepted_per_second to represent the metric. The name is the custom metric name in a custom HPA policy.
    as: "nginx_connections_accepted_per_second"
    # Calculate rate(nginx_connections_accepted[2m]) to specify the number of requests received per second.
  metricsQuery: 'rate(<<.Series>>{<<.LabelMatchers>>,container!="POD"}[2m])'
```

Figure 14-2 Modifying ConfigMap data



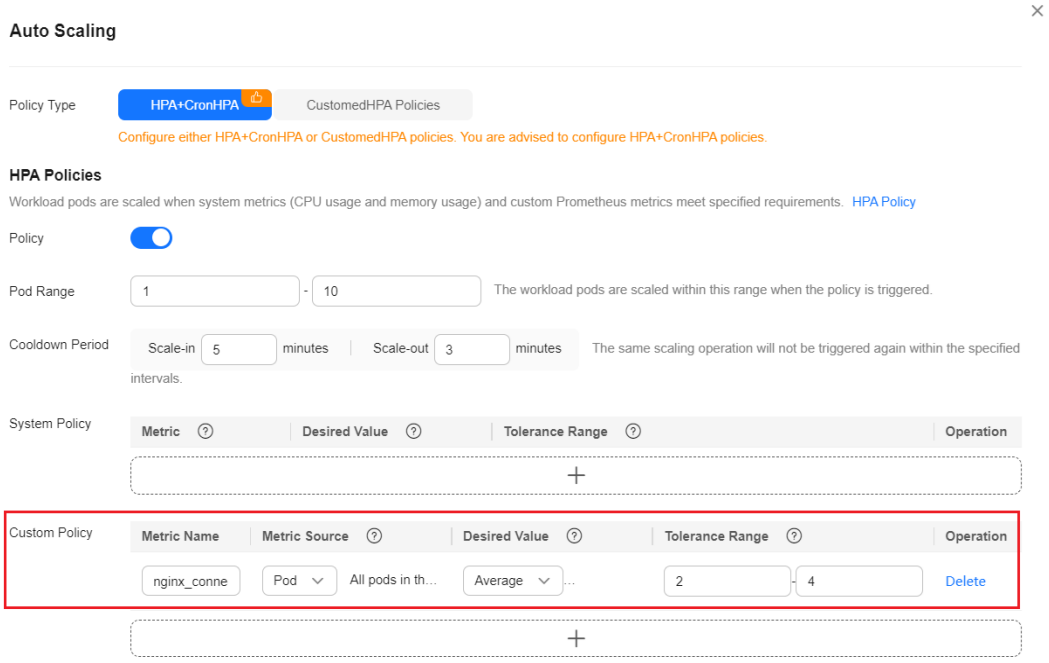
Step 4 Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

Figure 14-3 Redeploying custom-metrics-apiserver



Step 5 In the navigation pane, choose **Workloads**. Locate the workload for which you want to create an HPA policy and choose **More** > **Auto Scaling**. In the **Custom Policy** area, you can select the preceding parameters to create an auto scaling policy.

Figure 14-4 Creating an HPA policy



----End

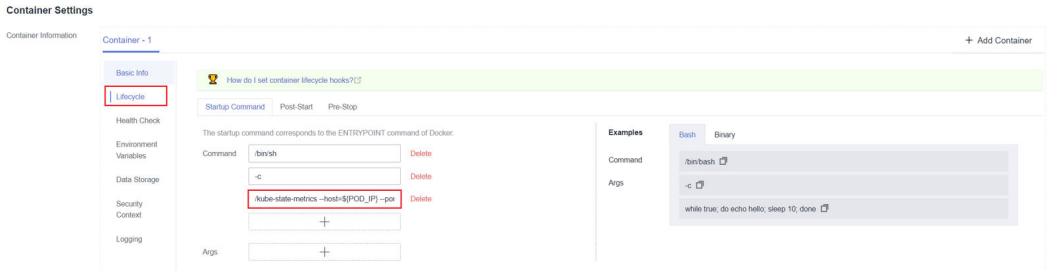
Collecting All Labels and Annotations of a Pod

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**.

Step 2 Switch to the **monitoring** namespace, click the **Deployments** tab, and click the name of the **kube-state-metrics** workload. On the page displayed, click the **Containers** tab and click **Edit** on the right.

Step 3 In the **Lifecycle** area of the container settings, edit the startup command.

Figure 14-5 Editing the startup command



To collect labels, add the following information to the end of the original **kube-state-metrics** startup parameter:

```
--metric-labels-allowlist=pods=[*],nodes=[node,failure-domain.beta.kubernetes.io/
zone,topology.kubernetes.io/zone]
```

To collect annotations, add parameters in the startup parameters in the same way.

```
--metric-annotations-allowlist=pods=[*],nodes=[node,failure-domain.beta.kubernetes.io/
zone,topology.kubernetes.io/zone]
```

NOTICE

When editing the startup command, do not modify other original startup parameters. Otherwise, the component may be abnormal.

Step 4 **kube-state-metrics** starts to collect the labels/annotations of pods and nodes and checks whether **kube_pod_labels/kube_pod_annotations** is in the collection task of Prometheus.

```
kubectl get servicemonitor kube-state-metrics -nmonitoring -oyaml | grep kube_pod_labels
```

----End

For more kube-state-metrics startup parameters, see **kube-state-metrics/cli-arguments**.

Change History

Table 14-39 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.12.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	• CCE clusters 1.31 are supported. • Upgraded Prometheus.	2.53.2

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.11.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	2.37.8
3.10.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	The NodeExporter component is upgraded to 1.8.0.	2.37.8
3.10.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Clusters 1.29 are supported.	2.37.8
3.9.5	v1.21 v1.23 v1.25 v1.27 v1.28	• Added the option for collecting custom metrics. This function is enabled by default. • Removed support for clusters 1.17 and 1.19. • Removed Grafana. Grafana is split into an independent add-on. • Collected only free metrics and custom service discovery metrics by default. • Upgraded open source components.	2.37.8

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.8.2	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27	Fixed some issues.	**2.35.0**
3.7.3	v1.17 v1.19 v1.21 v1.23 v1.25	None	**2.35.0**
3.7.2	v1.17 v1.19 v1.21 v1.23 v1.25	Supported collection of Virtual Kubelet pod metrics.	**2.35.0**
3.7.1	v1.17 v1.19 v1.21 v1.23 v1.25	Supports PrometheusAgent.	**2.35.0**
3.6.6	v1.17 v1.19 v1.21 v1.23 v1.25	• Grafana is upgraded to 7.5.17. • Supported containerd nodes.	**2.35.0**
3.5.1	v1.17 v1.19 v1.21 v1.23	None	**2.35.0**
3.5.0	v1.17 v1.19 v1.21 v1.23	Updated the add-on to its community version v2.35.0.	**2.35.0**

14.3.2 Cloud Native Log Collection

Introduction

The Cloud Native Log Collection add-on (formerly log-agent) is developed based on Fluent Bit and OpenTelemetry for collecting logs and Kubernetes events. This add-on supports CRD-based log collection policies. It collects and forwards standard output logs, container file logs, node logs, and Kubernetes event logs in a cluster based on configured policies. It also reports Kubernetes events to AOM for configuring event alarms. By default, all abnormal events and some normal events are reported.

NOTE

In 1.3.2 and later versions, Cloud Native Log Collection reports all warning events and some normal events to AOM by default. The reported events can be used to configure alarms. If the cluster version is 1.19.16, 1.21.11, 1.23.9, 1.25.4, or later, after Cloud Native Log Collection is installed, events are reported to AOM by this add-on instead of the control plane component. After Cloud Native Log Collection is uninstalled, events will not be reported to AOM.

Notes and Constraints

This add-on is available only in clusters v1.17 or later.

Add-on Performance

Item	Description	Remarks
Size of a log	Each individual log must not exceed 512 KB in size. In the case of multi-line logs, the length of each line will be calculated separately.	None
Maximum number of collected files	On a single node, the total number of files that can be listened by all log collection rules is limited to 4,095.	None
Log collection rate	• If the add-on version is earlier than 1.5.0, in each cluster, no more than 10,000 single-line logs can be collected per second, and no more than 2000 multi-line logs can be collected per second. • If the add-on version is 1.5.0 or later, on each node, no more than 20,000 logs or 10 MB of logs can be collected per second.	Service quality cannot be ensured if any of these limits is exceeded.
Configuration update	Configuration updates take effect in 1 to 3 minutes.	None

Permissions

The fluent-bit component of the add-on reads and collects the stdout logs on each node, file logs in pods, and node logs based on the collection configuration.

The following permissions are required for running the fluent-bit component:

- CAP_DAC_OVERRIDE: ignores the discretionary access control (DAC) restrictions on files.
- CAP_FOWNER: ignores the restrictions that the file owner ID must match the process user ID.
- DAC_READ_SEARCH: ignores the DAC restrictions on file reading and catalog research.
- SYS_PTRACE: allows all processes to be traced.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Cloud Native Log Collection** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** depending on the number of logs on nodes. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  You can select the **Small** option for clusters where the logs of a single node are less than 5000/s or 5 MB/s, and select the **Large** option for clusters where the logs on a single node are less than 10000/s or 10 MB/s.
- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure add-on parameters, which is supported by the add-on of v1.6.1 or later. For details, see **Tail**.

Parameter	Description	Default Value	
		Large Specification	Other Specification
Initial Buffer Size (Buffer_Chunk_Size)	The size of the initial buffer used to read files.	256k	128k

Parameter	Description	Default Value	
		Large Specification	Other Specification
Maximum Buffer Size (Buffer_Max_Size)	Limit on the buffer size for each monitored file. When a buffer is added, the number of memory buffers that can be added is restricted by this value. If the limit is exceeded, the file will be removed from the monitored file list.	1024k	512k
Memory Buffer Limit (Mem_Buf_Limit)	Memory limit when data is appended to the engine. When the limit is reached, the add-on will temporarily stop reading log file data. The reading will resume after the data is refreshed.	300mb	40mb

NOTE

The unit can be case-insensitive k, kb, m, mb, g or gb. If no unit is specified, the default unit of byte will be used.

Step 4 Configure deployment policies for the add-on pods.

NOTE

Scheduling policies do not take effect on add-on pods of the DaemonSet type.

Table 14-40 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.

Step 5 Click **Install**.

----End

Components

Table 14-41 Add-on components

Component	Description	Resource Type
fluent-bit	Lightweight log collector and forwarder deployed on each node to collect logs	DaemonSet
cop-logs	Used to generate soft links for collected files and run in the same pod as fluent-bit	DaemonSet
log-operator	Used to generate internal configuration files	Deployment
otel-collector	Used to collect logs from applications and services and report the logs to LTS	Deployment

Add-on Usage

This add-on can collect container standard output logs, container file logs, node logs, and Kubernetes events. You can use LTS or AOM to store the collected logs. These services support different types of logs. For details, see **Table 14-42**.

Table 14-42 Log storage description

Log Storage Location	Supported Log Types	How to Use
LTS	• Container standard output logs • Container file logs • Node logs • Kubernetes events	Go to **Logging** to create a policy. For details, see **Collecting Container Logs Using Cloud Native Log Collection**.
AOM	Kubernetes events	If the cluster version is 1.19.16, 1.21.11, 1.23.9, 1.25.4, or later, all abnormal events and some normal events will be reported by default. For details, see **Reporting Kubernetes Events to AOM**.

Change History

Table 14-43 Release history

Add-on Version	Supported Cluster Version	New Feature
1.7.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Fixed some issues.
1.7.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Clusters 1.31 are supported.
1.6.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• LTS log streams can be automatically created. • The **Buffer_Chunk_Size**, **Buffer_Max_Size**, and **Mem_Buf_Limit** parameters can be configured. • The **pod_ip** field has been added to the events reported to AOM.
1.6.0	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• Clusters 1.30 are supported. • Security hardening: The permissions of the add-on for accessing secrets are limited to the **monitoring** namespace.

Add-on Version	Supported Cluster Version	New Feature
1.5.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	The index function is added when the default log stream of container logs is created.
1.4.5	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.
1.4.2	v1.21 v1.23 v1.25 v1.27 v1.28	• Clusters 1.28 are supported. • Supported logging from on-premises clusters. • Supported special processing of AOM-related fields in GPU event reporting.
1.3.2	v1.17 v1.19 v1.21 v1.23 v1.25	Supports reporting Kubernetes events to AOM.
1.3.0	v1.17 v1.19 v1.21 v1.23 v1.25	Clusters 1.25 are supported.
1.2.3	v1.17 v1.19 v1.21 v1.23	None
1.2.2	v1.17 v1.19 v1.21 v1.23	log-agent is a cloud native log collection add-on built on open source Fluent Bit and OpenTelemetry and supports CRD-based log collection policies. It collects and forwards standard container output logs, container file logs, node logs, and Kubernetes event logs in a cluster following your rules.

14.3.3 CCE Node Problem Detector

Introduction

The CCE Node Problem Detector (formerly NPD) add-on monitors abnormal events of cluster nodes and can connect to a third-party monitoring platform. It is a daemon running on each node. It collects node issues from different daemons and reports them to the API server. It can run as a DaemonSet or a daemon.

The CCE Node Problem Detector add-on is developed based on the open-source project node-problem-detector. For details, see **node-problem-detector**.

Notes and Constraints

- When using this add-on, do not format or partition node disks.
- Each NPD process occupies 30 m CPU and 100 MiB of memory.
- If the NPD version is 1.18.45 or later, the EulerOS version of the host machine must be 2.5 or later.

Permissions

To monitor kernel logs, the NPD add-on needs to read the host **/dev/kmsg**. Therefore, the privileged mode must be enabled. For details, see **privileged**.

In addition, CCE mitigates risks according to the least privilege principle. Only the following privileges are available for NPD running:

- cap_dac_read_search: permission to access **/run/log/journal**.
- cap_sys_admin: permission to access **/dev/kmsg**.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Node Problem Detector** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

You can adjust the number of add-on pods and resource quotas as required. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure the add-on parameters.

Maximum Number of Isolated Nodes in a Fault: specifies the maximum number of nodes that can be isolated to prevent avalanches in case of a fault occurring on multiple nodes. You can configure this parameter either by percentage or quantity.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-44 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.

Parameter	Description
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-45 Add-on components

Component	Description	Resource Type
node-problem-controller	Isolate faults basically based on fault detection results.	Deployment
node-problem-detector	Detect node faults.	DaemonSet

NPD Check Items

NOTE

Check items are supported only in 1.16.0 and later versions.

Check items cover events and statuses.

- Event-related

  For event-related check items, when a problem occurs, NPD reports an event to the API server. The event type can be **Normal** (normal event) or **Warning** (abnormal event).

Table 14-46 Event-related check items

Check Item	Function	Description
OOMKilling	Listen to the kernel logs and check whether OOM events occur and are reported. Typical scenario: When the memory usage of a process in a container exceeds the limit, OOM is triggered and the process is terminated.	Warning event Listening object: **/dev/kmsg** Matching rule: "Killed process \\d+ (.+) total-vm:\\d+kB, anon-rss:\\d+kB, file-rss:\\d+kB.*"
TaskHung	Listen to the kernel logs and check whether taskHung events occur and are reported. Typical scenario: Disk I/O suspension causes process suspension.	Warning event Listening object: **/dev/kmsg** Matching rule: "task \ \S+:\\w+ blocked for more than \\w+ seconds\\."
Readonly Filesystem	Check whether the **Remount root filesystem read-only** error occurs in the system kernel by listening to the kernel logs. Typical scenario: A user detaches a data disk from a node by mistake on the ECS, and applications continuously write data to the mount point of the data disk. As a result, an I/O error occurs in the kernel and the disk is remounted as a read-only disk. **NOTE** If the rootfs of node pods is of the device mapper type, an error will occur in the thin pool if a data disk is detached. This will affect NPD and NPD will not be able to detect node faults.	Warning event Listening object: **/dev/kmsg** Matching rule: **Remounting filesystem read-only**

- Status-related

  For status-related check items, when a problem occurs, NPD reports an event to the API server and changes the node status synchronously. This function can be used together with **Node-problem-controller fault isolation** to isolate nodes.

  If the check period is not specified in the following check items, the default period is 30 seconds.

Table 14-47 Checking system components

Check Item	Function	Description
Container network component error CNIProblem	Check the status of the CNI components (container network components).	None
Container runtime component error CRIProblem	Check the status of Docker and containerd of the CRI components (container runtime components).	Check object: Docker or containerd
Frequent restarts of Kubelet FrequentKubeletRestart	Periodically backtrack system logs to check whether the key component Kubelet restarts frequently.	• Default threshold: 10 restarts within 10 minutes If Kubelet restarts for 10 times within 10 minutes, it indicates that the system restarts frequently and a fault alarm is generated. • Listening object: logs in the **/run/log/journal** directory **NOTE** The Ubuntu and HCE 2.0 OSs do not support the preceding check items due to incompatible log formats.
Frequent restarts of Docker FrequentDockerRestart	Periodically backtrack system logs to check whether the container runtime Docker restarts frequently.	
Frequent restarts of containerd FrequentContainerdRestart	Periodically backtrack system logs to check whether the container runtime containerd restarts frequently.	
kubelet error KubeletProblem	Check the status of the key component Kubelet.	None
kube-proxy error KubeProxyProblem	Check the status of the key component kube-proxy.	None

Table 14-48 Checking system metrics

Check Item	Function	Description
Conntrack table full ConntrackFullProblem	Check whether the conntrack table is full.	• Default threshold: 90% • Usage: **nf_conntrack_count** • Maximum value: **nf_conntrack_max**

Check Item	Function	Description
Insufficient disk resources DiskProblem	Check the usage of the system disk and CCE data disks (including the CRI logical disk and kubelet logical disk) on the node.	• Default threshold: 90% • Source: `df -h` Currently, additional data disks are not supported.
Insufficient file handles FDProblem	Check if the FD file handles are used up.	• Default threshold: 90% • Usage: the first value in **/proc/sys/fs/file-nr** • Maximum value: the third value in **/proc/sys/fs/file-nr**
Insufficient node memory MemoryProblem	Check whether memory is used up.	• Default threshold: 80% • Usage: **MemTotal-MemAvailable** in **/proc/meminfo** • Maximum value: **MemTotal** in **/proc/meminfo**
Insufficient process resources PIDProblem	Check whether PID process resources are exhausted.	• Default threshold: 90% • Usage: **nr_threads in /proc/loadavg** • Maximum value: smaller value between **/proc/sys/kernel/pid_max** and **/proc/sys/kernel/threads-max**.

Table 14-49 Checking the storage

Check Item	Function	Description
Disk read-only DiskReadonly	Periodically perform write tests on the system disk and CCE data disks (including the CRI logical disk and Kubelet logical disk) of the node to check the availability of key disks.	Detection paths: • /mnt/paas/kubernetes/kubelet/ • /var/lib/docker/ • /var/lib/containerd/ • /var/paas/sys/log/cceaddon-npd/ The temporary file **npd-disk-write-ping** is generated in the detection path. Currently, additional data disks are not supported.

Check Item	Function	Description
emptyDir storage pool error EmptyDirVolumeGroupStatusError	Check whether the ephemeral volume group on the node is normal. Impact: Pods that depend on the storage pool cannot write data to the temporary volume. The temporary volume is remounted as a read-only file system by the kernel due to an I/O error. Typical scenario: When creating a node, a user configures two data disks as an ephemeral volume storage pool. Some data disks are deleted by mistake. As a result, the storage pool becomes abnormal.	• Detection period: 30s • Source: `vgs -o vg_name, vg_attr` • Principle: Check whether the VG (storage pool) is in the P state. If yes, some PVs (data disks) are lost. • Joint scheduling: The scheduler can automatically identify a PV storage pool error and prevent pods that depend on the storage pool from being scheduled to the node. • Exceptional scenario: The NPD add-on cannot detect the loss of all PVs (data disks), resulting in the loss of VGs (storage pools). In this case, kubelet automatically isolates the node, detects the loss of VGs (storage pools), and updates the corresponding resources in **nodestatus.allocatable** to **0**. This prevents pods that depend on the storage pool from being scheduled to the node. The damage of a single PV cannot be detected by this check item, but by the ReadonlyFilesystem check item.
PV storage pool error LocalPvVolumeGroupStatusError	Check the PV group on the node. Impact: Pods that depend on the storage pool cannot write data to the persistent volume. The persistent volume is remounted as a read-only file system by the kernel due to an I/O error. Typical scenario: When creating a node, a user configures two data disks as a persistent volume storage pool. Some data disks are deleted by mistake.	

Check Item	Function	Description
Mount point error MountPointProblem	Check the mount point on the node. Exceptional definition: You cannot access the mount point by running the **cd** command. Typical scenario: Network File System (NFS), for example, obsfs and s3fs is mounted to a node. When the connection is abnormal due to network or peer NFS server exceptions, all processes that access the mount point are suspended. For example, during a cluster upgrade, a kubelet is restarted, and all mount points are scanned. If the abnormal mount point is detected, the upgrade fails.	Alternatively, you can run the following command: `for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir; done && echo "ok"`

Check Item	Function	Description
Suspended disk I/O DiskHung	Check whether I/O suspension occurs on all disks on the node, that is, whether I/O read and write operations are not responded. Definition of I/O suspension: The system does not respond to disk I/O requests, and some processes are in the D state. Typical scenario: Disks cannot respond due to abnormal OS hard disk drivers or severe faults on the underlying network.	• Check object: all data disks • Source: /proc/diskstat Alternatively, you can run the following command: `iostat -xmt 1` • Thresholds: (All following conditions must be met.) – Average usage (**ioutil**) ≥ 0.99 – Average I/O queue length (**avgqu-sz**) ≥ 1 – Average I/O transfer volume ≤ 1 Average I/O transfer volume = Number of writes completed per second (**iops**, unit: w/s) + Amount of data written per second (**ioth**, unit: wMB/s) **NOTE** In some OSs, no data changes during I/O. In this case, calculate the CPU I/O time usage. The value of **iowait** should be greater than 0.8.

Check Item	Function	Description
Slow disk I/O DiskSlow	Check whether all disks on the node have slow I/Os, that is, whether I/Os respond slowly. Typical scenario: EVS disks have slow I/Os due to network fluctuation.	• Check object: all data disks • Source: /proc/diskstat Alternatively, you can run the following command: `iostat -xmt 1` • Default threshold: Average I/O latency (**await**) ≥ 5000 ms **NOTE** If I/O requests are not responded and the **await** data is not updated, this check item is invalid.

Table 14-50 Other check items

Check Item	Function	Description
Abnormal NTP NTPProblem	Check whether the node clock synchronization service ntpd or chronyd is running properly and whether a system time drift is caused.	Default clock offset threshold: 8000 ms
Process D error ProcessD	Check whether there is a process D on the node.	Default threshold: 10 abnormal processes detected for three consecutive times Source: • /proc/{PID}/stat • Alternately, you can run the **ps aux** command. Exceptional scenario: The ProcessD check item ignores the resident D processes (heartbeat and update) on which the SDI driver on the BMS node depends.
Process Z error ProcessZ	Check whether the node has processes in Z state.	

Check Item	Function	Description
ResolvConf error ResolvConfFileProblem	Check whether the ResolvConf file is lost. Check whether the ResolvConf file is normal. Exceptional definition: No upstream domain name resolution server (nameserver) is included.	Object: **/etc/resolv.conf**
Existing scheduled event ScheduledEvent	Check whether scheduled live migration events exist on the node. A live migration plan event is usually triggered by a hardware fault and is an automatic fault rectification method at the IaaS layer. Typical scenario: The host is faulty. For example, the fan is damaged or the disk has bad sectors. As a result, live migration is triggered for VMs.	Source: • http://169.254.169.254/meta-data/latest/events/scheduled This check item is an Alpha feature and is disabled by default.
The spot price node is being reclaimed. SpotPriceNodeReclaimNotification	Check whether the reclaiming of a spot price node is interrupted due to preemption.	Default check interval: 120 seconds Default fault handling policy: evicts node loads.

The kubelet component has the following default check items, which have bugs or defects. You can fix them by upgrading the cluster or using NPD.

Table 14-51 Default kubelet check items

Check Item	Function	Description
Insufficient PID resources PIDPressure	Check whether PIDs are sufficient.	• Interval: 10 seconds • Threshold: 90% • Defect: In community version 1.23.1 and earlier versions, this check item becomes invalid when over 65535 PIDs are used. For details, see **issue 107107**. In community version 1.24 and earlier versions, thread-max is not considered in this check item.
Insufficient memory MemoryPressure	Check whether the allocable memory for the containers is sufficient.	• Interval: 10 seconds • Threshold: max. 100 MiB • Allocable = Total memory of a node – Reserved memory of a node • Defect: This check item checks only the memory consumed by containers, and does not consider that consumed by other elements on the node.
Insufficient disk resources DiskPressure	Check the disk usage and inodes usage of the kubelet and Docker disks.	• Interval: 10 seconds • Threshold: 90%

Node-problem-controller Fault Isolation

NOTE

Fault isolation is supported only by add-ons of 1.16.0 and later versions.

By default, if multiple nodes become faulty, NPC adds taints to up to 10% of the nodes. You can set **npc.maxTaintedNode** to increase the threshold.

The open source NPD plugin provides fault detection but not fault isolation. CCE enhances the node-problem-controller (NPC) based on the open source NPD. This component is implemented based on the Kubernetes **node controller**. For faults reported by NPD, NPC automatically adds taints to nodes for node fault isolation.

Table 14-52 Parameters

Parameter	Description	Default
npc.enable	Whether to enable NPC This parameter is not supported in 1.18.0 or later versions.	true
npc.maxTaintedNode	The maximum number of nodes that NPC can add taints to when an individual fault occurs on multiple nodes for minimizing impact. The value can be in int or percentage format.	10% Value range: • The value is in int format and ranges from 1 to infinity. • The value ranges from 1% to 100%, in percentage. The minimum value of this parameter multiplied by the number of cluster nodes is 1.
npc.nodeAffinity	Node affinity of the controller	N/A

Viewing NPD Events

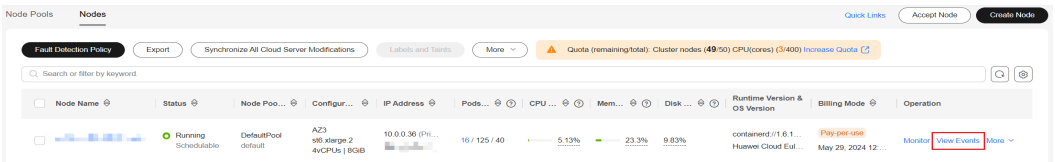
Events reported by the NPD add-on can be queried on the **Nodes** page.

Step 1 Log in to the CCE console.

Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane.

Step 3 Locate the row that contains the target node, and click **View Events**.

Figure 14-6 Viewing node events



----End

Configuring NPD Metric Alarms

For NPD status check items, you can configure alarm rules to notify you of exceptions by SMS message or email. For details about how to create a custom alarm rule, see **Configuring Alarms in Alarm Center**.

NOTE

To use NPD check items to configure alarm rules, you need to install the Cloud Native Cluster Monitoring add-on in the cluster and the add-on has been interconnected with an AOM instance.

Collecting Prometheus Metrics

The NPD daemon pod exposes Prometheus metric data on port 19901. By default, the NPD pod is added with the annotation **metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"prometheus","path":"/metrics","port":"19901","names":""}]'**. You can build a Prometheus collector to identify and obtain NPD metrics from **http://{{NpdPodIP}}:{{NpdPodPort}}/metrics**.

NOTE

If the NPD add-on version is earlier than 1.16.5, the exposed port of Prometheus metrics is **20257**.

Currently, the metric data includes **problem_counter** and **problem_gauge**, as shown below.

```
# HELP problem_counter Number of times a specific type of problem have occurred.
# TYPE problem_counter counter
problem_counter{reason="DockerHung"} 0
problem_counter{reason="DockerStart"} 0
problem_counter{reason="EmptyDirVolumeGroupStatusError"} 0
...
# HELP problem_gauge Whether a specific type of problem is affecting the node or not.
# TYPE problem_gauge gauge
problem_gauge{reason="CNIIsDown",type="CNIProblem"} 0
problem_gauge{reason="CNIIsUp",type="CNIProblem"} 0
problem_gauge{reason="CRIIsDown",type="CRIProblem"} 0
problem_gauge{reason="CRIIsUp",type="CRIProblem"} 0
..
```

Change History

Table 14-53 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.16	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.19.11	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Fixed some issues.	0.8.10
1.19.8	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• Compatible with a single system disk. • Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Allowed adding a taint to a spot ECS before its release for the node to evict pods. • Synchronized time zones used by the add-on and the node. • CCE clusters 1.30 are supported.	0.8.10
1.19.1	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.	0.8.10
1.19.0	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	0.8.10
1.18.48	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.18.46	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	0.8.10
1.18.22	v1.19 v1.21 v1.23 v1.25 v1.27	None	0.8.10
1.18.14	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Allowed adding a taint to a spot ECS before its release for the node to evict pods. • Synchronized time zones used by the add-on and the node.	0.8.10
1.18.10	v1.19 v1.21 v1.23 v1.25	• Optimized the configuration page. • Added threshold configuration to the DiskSlow check item. • Added threshold configuration to the NTPProblem check item. • Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Supported interruption detection for spot ECSs and evicted pods on the nodes before the interruption.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.17.4	v1.17 v1.19 v1.21 v1.23 v1.25	Optimized DiskHung check item.	0.8.10
1.17.3	v1.17 v1.19 v1.21 v1.23 v1.25	• The maximum number of taint nodes that can be added to the NPC can be configured by percentage. • Added the ProcessZ check item. • Added the time deviation detection to the NTPProblem check item. • Fixed the processes consistently in the D state (exist in the BMS node).	0.8.10
1.17.2	v1.17 v1.19 v1.21 v1.23 v1.25	• Added the DiskHung check item for disk I/O. • Added the DiskSlow check item for disk I/O. • Added the ProcessD check item. • Added MountPointProblem to check the health of mount points. • To avoid conflicts with the service port range, the default health check listening port is changed to **19900**, and the default Prometheus metric exposure port is changed to **19901**. • Clusters 1.25 are supported.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.16.4	v1.17 v1.19 v1.21 v1.23	• Added the beta check item ScheduledEvent to detect cold and live VM migration events caused by host machine exceptions using the metadata API. This check item is disabled by default.	0.8.10
1.16.3	v1.17 v1.19 v1.21 v1.23	Added the function of checking the ResolvConf configuration file.	0.8.10
1.16.1	v1.17 v1.19 v1.21 v1.23	• Added node-problem-controller. Supported basic fault isolation. • Added the PID, FD, disk, memory, temporary volume pool, and PV pool check items.	0.8.10
1.15.0	v1.17 v1.19 v1.21 v1.23	• Hardened check items comprehensively to avoid false positives. • Supported kernel check. Supported reporting of OOMKilled and TaskHung events.	0.8.10
1.14.11	v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	0.7.1
1.14.5	v1.17 v1.19	Fixed the issue that monitoring metrics cannot be obtained.	0.7.1
1.14.4	v1.17 v1.19	• Supported containerd nodes.	0.7.1
1.14.2	v1.17 v1.19	• CCE clusters 1.19 are supported. • Supported Ubuntu OS and Kata containers.	0.7.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.13.8	v1.15.11 v1.17	• Fixed the CNI health check issue on the container tunnel network. • Adjusted resource quotas.	0.7.1
1.13.6	v1.15.11 v1.17	Fixed the issue that zombie processes are not reclaimed.	0.7.1
1.13.5	v1.15.11 v1.17	Added taint tolerance configuration.	0.7.1
1.13.2	v1.15.11 v1.17	Added resource limits and enhanced the detection capability of the cni add-on.	0.7.1

14.3.4 CCE Network Metrics Exporter

Introduction

CCE Network Metrics Exporter enhances the observability of container networks by providing monitoring data related to container networks in CCE Turbo clusters, so you can observe network traffic and detect and locate network problems faster. The add-on pods can only be deployed on HCE 2.0 nodes running x86/Arm, or EulerOS nodes running x86.

The IP, UDP, and TCP traffic can be monitored by pod or flow. You can use podSelector to select the monitoring backend. Multiple monitoring tasks can be created and monitoring metrics can be selected as required. The label information of pods can also be obtained. The monitoring data has been adapted to Prometheus. You can call the Prometheus API to view monitoring data.

Notes and Constraints

- This add-on can be installed only in CCE Turbo clusters 1.19 or later. The add-on pods can run only on nodes running HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later) or nodes running EulerOS on x86.
- This add-on can be installed on nodes that use the containerd or Docker container engine. In containerd nodes, it can trace pod updates in real time. In Docker nodes, it can query pod updates in polling mode.
- Only traffic statistics of secure containers using the Kata container runtimes and common containers (using the runC container runtimes) in a CCE Turbo cluster can be collected.
- By default, traffic monitoring is not carried out after the installation of the add-on. To set up a traffic monitoring task, you must create a MonitorPolicy in the kube-system namespace.

- Pods using the host network mode cannot be monitored.
- Ensure that there are enough resources on a node for installing the add-on.
- The source of monitoring labels and user labels must be already available before a pod is created.
- You can specify a maximum of five labels (a maximum of 10 labels in versions later than 1.3.4). You cannot specify the labels used by the system. Labels used by the system include **pod**, **task**, **ipfamily**, **srcip**, **dstip**, **srcport**, **dstport**, and **protocol**.
- Nodes running HCE 2.0 on x86 do not support the **tcp_drop** monitoring item.

Installing the Add-on

Step 1 Log in to the CCE console and click the CCE Turbo cluster name to access the cluster. In the navigation pane, choose **Add-ons**, locate **CCE Network Metrics Exporter** on the right, and click **Install**.

Step 2 On the Install Add-on page, check the add-on configuration.

No parameter can be configured for the current add-on.

Step 3 Click **Install**.

After the add-on is installed, select the cluster and choose **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 14-54 Add-on components

Component	Description	Resource Type
dolphin	Monitor the container network traffic in CCE Turbo clusters.	Daemon Set

Monitoring Metrics of dolphin

You can deliver a monitoring task by creating a MonitorPolicy. A MonitorPolicy can be created by calling an API or using the **kubectl apply** command after logging in to a worker node. A MonitorPolicy represents a monitoring task and provides optional parameters such as **selector** and **podLabel**. The following table describes the supported monitoring metrics.

Table 14-55 Supported monitoring metrics

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of IPv4 packets sent to the Internet	dolphin_ip4_send_pkt_internet	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of IPv4 bytes sent to the Internet	dolphin_ip4_send_byte_internet	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IPv4 packets	dolphin_ip4_rcv_pkt	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of received IPv4 bytes	dolphin_ip4_rcv_byte	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IPv4 packets	dolphin_ip4_send_pkt	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86
Number of sent IPv4 bytes	dolphin_ip4_send_byte	Pod	runc/ kata	v1.19 or later	1.1.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Health status of the latest health check	dolphin_health_check_status	Pod	runc/ kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Total number of successful health checks	dolphin_health_check_successful_counter	Pod	runc/ kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Total number of failed health checks	dolphin_health_check_failed_counter	Pod	runc/kata	v1.19 or later	1.2.2	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of received IP packets	dolphin_ip_receive_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received IP bytes	dolphin_ip_receive_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of sent IP packets	dolphin_ip_send_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent IP bytes	dolphin_ip_send_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of received TCP packets	dolphin_tcp_receive_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received TCP bytes	dolphin_tcp_receive_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of sent TCP packets	dolphin_tcp_send_pkt	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent TCP bytes	dolphin_tcp_send_byte	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of retransmitted TCP packets	dolphin_tcp_retrans	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of new TCP connections	dolphin_tcp_connection	Pod	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of received IP packets	dolphin_flow_ip_receive_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received IP bytes	dolphin_flow_ip_receive_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of sent IP packets	dolphin_flow_ip_send_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent IP bytes	dolphin_flow_ip_send_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of received TCP packets	dolphin_flow_tcp_receive_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of received TCP bytes	dolphin_flow_tcp_receive_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of sent TCP packets	dolphin_flow_tcp_send_pkt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent TCP bytes	dolphin_flow_tcp_send_byte	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of retransmitted TCP packets	dolphin_flow_tcp_retrans	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
TCP smoothed round trip	dolphin_flow_tcp_srtt	Flow	runC	v1.23 or later	1.3.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm (supported by CCE Network Metrics Exporter 1.4.5 or later)
Number of lost TCP packets	dolphin_tcp_drop	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of lost TCP packets	dolphin_flow_tcp_drop	Flow	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
TCP connection setup failures	dolphin_tcp_connection_failure	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of received UDP packets	dolphin_udp_receive_pkt	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of received UDP bytes	dolphin_udp_receive_byte	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of sent UDP packets	dolphin_udp_send_pkt	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent UDP bytes	dolphin_udp_send_byte	Pod	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of received UDP packets	dolphin_flow_udp_receive_pkt	Flow	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of received UDP bytes	dolphin_flow_udp_receive_byte	Flow	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm
Number of sent UDP packets	dolphin_flow_udp_send_pkt	Flow	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
Number of sent UDP bytes	dolphin_flow_udp_send_byte	Flow	runC	v1.23 or later	1.4.5	EulerOS 2.9 on x86 EulerOS 2.10 on x86 HCE 2.0 on x86 or Arm

Delivering a Monitoring Task

The template for creating a MonitorPolicy is as follows:

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
   name: example-task           # Monitoring task name.
   namespace: kube-system       # The value must be kube-system. This field is mandatory.
spec:
   selector:                    # (Optional) Backend monitored by the add-on. The value has the same format as the labelSelector. By default, all pods on the node are monitored.
   matchLabels:
     app: nginx
   matchExpressions:
     - key: app
       operator: In
       values:
         - nginx
 podLabel: [app]              # (Optional) Pod label.
 ip4Tx:                      # (Optional) Indicates whether to collect statistics about the number of sent IPv4 packets and the number of sent IPv4 bytes. This function is disabled by default.
   enable: true
   ip4Rx:                      # (Optional) Indicates whether to collect statistics about the number of received IPv4 packets and the number of received IPv4 bytes. This function is disabled by default.
   enable: true
   ip4TxInternet:              # (Optional) Indicates whether to collect statistics about the number of sent IPv4 packets and the number of sent IPv4  bytes. This function is disabled by default.
   enable: true
 healthCheck:                # (Optional) Whether to collect statistics about whether the latest health check result is healthy and the total number of healthy times and unhealthy times in the pod health checks of the local node. This function is disabled by default.
   enable: true              # true false
   failureThreshold: 3       # (Optional) Number of failures that determine the health check is unhealthy. One check failure is considered as unhealthy by default.
   periodSeconds: 5          # (Optional) Interval between health checks, in seconds. The default value is 60.
   command: ""               # (Optional) Health check command. The value can be ping (default), arping, or curl.
   ipFamilies: [""]          # (Optional) Health check IP address family. The value is IPv4 by default.
   port: 80                  # (Optional) Port number, which is mandatory when curl is used.
   path: ""                  # (Optional) HTTP API path, which is mandatory when curl is used.
 monitor:
   ip:
     ipReceive:
       aggregateType: flow     # (Optional). The value can be pod (monitored by pod) or flow (monitored by flow).
```



```
    ipSend:
      aggregateType: flow      # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
  tcp:
    tcpReceive:
      aggregateType: flow      # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpSend:
      aggregateType: flow      # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpRetrans:
      aggregateType: flow      # (Optional). The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpRtt:
      aggregateType: flow      # (Optional). The value can be flow (monitored by flow). The unit is μs.
    tcpNewConnection:
      aggregateType: pod       # (Optional). The value can be pod (monitored by pod).
    tcpDrop:
      aggregateType: flow      # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
    tcpConnectionFailure:
      aggregateType: pod       # (Optional) The value can be pod (monitored by pod).
  udp:
    udpReceive:
      aggregateType: flow      # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
    udpSend:
      aggregateType: flow      # (Optional) The value can be pod (monitored by pod) or flow (monitored
by flow).
```

PodLabel: You can enter the labels of multiple pods and separate them with commas (,), for example, [app, version].

The labels must comply with the following rules, and the corresponding regular expression is (^[a-zA-Z0-9_][a-zA-Z0-9\-\._/]{0,254}$).

- A maximum of five labels can be entered (a maximum of 10 labels in versions later than 1.3.4). A label can contain a maximum of 255 characters.
- All characters except letters and digits are replaced with underscores (_).

You can create, modify, and delete monitoring tasks in the preceding format. A maximum of 10 monitoring tasks can be created. When multiple monitoring tasks match the same monitoring backend, each monitoring backend generates the monitoring metrics specific to the number of monitoring tasks.

NOTE

- If you modify or delete a monitoring task, monitoring data collected by the monitoring task will be lost. Therefore, exercise caution when performing this operation.
- If the add-on is uninstalled, the MonitorPolicy of the monitoring task will be removed together with the add-on.

Example application scenarios:

1. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the three health check metrics. By default, the **ping** command is used to detect local pods. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

   ```
   apiVersion: crd.dolphin.io/v1
   kind: MonitorPolicy
   metadata:
     name: example-task
   ```



```
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  healthCheck:
    enable: true
    failureThreshold: 3
    periodSeconds: 5
```

2. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the three health check metrics. Customized **curl** command is used, which considers only the network connectivity. That is, no matter what the HTTP code is returned by the program, the pod is considered healthy as long as the network is connected. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  healthCheck:
    enable: true
    failureThreshold: 3
    periodSeconds: 5
    command: "curl"
    port: 80
    path: "healthz"
```

3. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates monitoring data by pod, including the number of sent IP packets, received IP packets, sent IP bytes, received IP bytes, sent UDP packets, received UDP packets, sent UDP bytes, received UDP bytes, lost TCP packets, TCP connection setup failures, sent TCP packets, received TCP packets, sent TCP bytes, received TCP bytes, retransmitted TCP packets, and new TCP connections. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  monitor:
    ip:
      ipReceive:
        aggregateType: pod
      ipSend:
        aggregateType: pod
    tcp:
      tcpReceive:
```



```
      aggregateType: pod
    tcpSend:
      aggregateType: pod
    tcpRetrans:
      aggregateType: pod
    tcpNewConnection:
      aggregateType: pod
    tcpDrop:
      aggregateType: pod
    tcpConnectionFailure:
      aggregateType: pod
  udp:
    udpReceive:
      aggregateType: pod
    udpSend:
      aggregateType: pod
```

4. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates monitoring data by flow, including the number of sent IP packets, received IP packets, sent IP bytes, received IP bytes, sent TCP packets, received TCP packets, sent TCP bytes, received TCP bytes, retransmitted TCP packets, lost TCP packets, sent UDP packets, received UDP packets, sent UDP bytes, received UDP bytes, and TCP round-trip time (μs). If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**. Flow-based monitoring helps you learn about detailed container traffic information. It generates a large amount of data that occupies more CPU and memory resources. Use flow-based monitoring based on your needs.

   A flow-based IP monitoring task (one or more IP monitoring items enabled in a MonitorPolicy) occupies 2.6 MiB of kernel memory. A flow-based TCP monitoring task (one or more TCP monitoring items enabled in a MonitorPolicy) occupies 14 MiB of kernel memory.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  monitor:
    ip:
      ipReceive:
        aggregateType: flow
      ipSend:
        aggregateType: flow
    tcp:
      tcpReceive:
        aggregateType: flow
      tcpSend:
        aggregateType: flow
      tcpRetrans:
        aggregateType: flow
      tcpRtt:
        aggregateType: flow
      tcpDrop:
        aggregateType: flow
    udp:
      udpReceive:
        aggregateType: flow
      udpSend:
        aggregateType: flow
```


NOTE

If the data generated by flow-based monitoring exceeds a certain limit, excess flow statistics will be lost. The restrictions are as follows:

- A maximum of 50,000 TCP flows (per monitoring task) can be collected in kernel mode within 10 seconds.
- A maximum of 10,000 IP flows (per monitoring task) can be collected in kernel mode within 10 seconds.
- A maximum of 60,000 flow statistical records (all monitoring tasks) can be cached at the interval between two Prometheus data fetches.
- If Prometheus does not obtain monitoring data for a long time, only the monitoring data generated within the latest hour will be cached.

5. The example below monitors all pods on a node and generates the number of sent IPv4 packets and the number of sent IPv4 bytes. If the monitored container contains the **app** label, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  podLabel: [app]
  ip4Tx:
    enable: true
```

6. The example below monitors all pods with label **app=nginx** selected by the **labelselector** on a node and generates the number of sent IPv4 packets, received IPv4 packets, sent IPv4 bytes, received IPv4 bytes, IPv4 packets sent to the public network, and IPv4 bytes sent to the public network. If the monitored container contains the **test** and **app** labels, the key-value information of the corresponding label is carried in the monitoring metrics. Otherwise, the value of the corresponding label is **not found**.

```
apiVersion: crd.dolphin.io/v1
kind: MonitorPolicy
metadata:
  name: example-task
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: nginx
  podLabel: [test, app]
  ip4Tx:
    enable: true
  ip4Rx:
    enable: true
  ip4TxInternet:
    enable: true
```

Checking Traffic Statistics

The CCE Network Metrics Exporter add-on outputs monitoring information in the Prometheus exporter format. You can obtain this information in either of the following ways:

- Directly access the service port 10001 provided by CCE Network Metrics Exporter, for example, **http://*{POD_IP}*:10001/metrics**.

Note that if you access the add-on service port on a node, allow access from the security group of the node and pod.

Examples of the monitored information:

- Example 1 (number of IPv4 packets sent to the Internet):
  ```
  dolphin_ip4_send_pkt_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 241
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod to the public network is **241**.
- Example 2 (number of IPv4 bytes sent to the Internet):
  ```
  dolphin_ip4_send_byte_internet{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task" } 23618
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes sent by the pod to the public network is **23618**.
- Example 3 (number of sent IPv4 packets):
  ```
  dolphin_ip4_send_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 379
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod is **379**.
- Example 4 (number of sent IPv4 bytes):
  ```
  dolphin_ip4_send_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 33129
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes sent by the pod is **33129**.
- Example 5 (number of received IPv4 packets):
  ```
  dolphin_ip4_rcv_pkt{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 464
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets received by the pod is **464**.
- Example 6 (number of received IPv4 bytes):
  ```
  dolphin_ip4_rcv_byte{app="nginx",pod="default/nginx-66c9c65dbf-zjg24",task="kube-system/example-task "} 34654
  ```
  In the preceding example, the namespace of the pod is **default**, the pod name is **nginx-66c9c65dbf-zjg24**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 bytes received by the pod is **34654**.
- Example 7 (health check status)
  ```
  dolphin_health_check_status{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
  ```
  In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**,

and the network health status of the pod is **0** (healthy). If the network status is unhealthy, the value will be **1**.

- Example 8 (number of successful health checks)

  ```
  dolphin_health_check_successful_counter{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 5
  ```

  In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of successful network health checks for the pod is **5**.

- Example 9 (number of failed health check failures)

  ```
  dolphin_health_check_failed_counter{app="nginx",pod="default/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
  ```

  In the preceding example, the namespace of the pod is **kube-system**, the pod name is **default/nginx-deployment-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of failed network health checks for the pod is **0**.

- Example 10 (flow-based monitoring result):

  ```
  dolphin_flow_tcp_send_byte{app="nginx",dstip="192.168.0.89",dstport="80",ipfamily="ipv4",pod="kube-system/nginx-b74766f5f-7582p",srcip="192.168.1.67",srcport="12973",task="kube-system/example-task"} 1725 1700538280914
  ```

  In the preceding example, the namespace of the pod is **kube-system**, the pod name is **nginx-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 TCP bytes sent from **192.168.1.67:12973** to **192.168.0.89:80** is **1725**. The timestamp is **1700538280914**.

- Example 11 (pod-based monitoring result):

  ```
  dolphin_tcp_send_pkt{app="nginx",ipfamily="ipv4",pod="kube-system/nginx-b74766f5f-7582p",task="kube-system/example-task"} 14
  dolphin_tcp_send_pkt{app="nginx",ipfamily="ipv6",pod="kube-system/nginx-b74766f5f-7582p",task="kube-system/example-task"} 0
  ```

  In the preceding example, the namespace of the pod is **kube-system**, the pod name is **nginx-b74766f5f-7582p**, the label is **app**, and the value is **nginx**. This metric is created by monitoring task **example-task**, and the number of IPv4 packets sent by the pod is **14**. **0** IPv6 packets were sent by this pod.

NOTE

If the container does not contain the specified label, the label value in the response body is **not found**. The format is as follows:

```
dolphin_ip4_send_byte_internet{test="not found", pod="default/nginx-66c9c65dbf-zjg24",task="default" } 23618
```


Change History

Table 14-56 Release history

Add-on Version	Supported Cluster Version	New Feature
1.4.18	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE Turbo clusters 1.31 are supported.
1.4.15	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE Turbo clusters 1.30 are supported.
1.4.7	v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
1.4.5	v1.23 v1.25 v1.27 v1.28 v1.29	• Supported pod-based UDP, TCP drop, and TCP connect fail monitoring for common containers. • Supported flow-based UDP and TCP drop monitoring of common containers. • Supported Huawei Cloud EulerOS 2.0 on x86 or Arm. • CCE clusters 1.29 are supported.
1.3.10	v1.23 v1.25 v1.27 v1.28	Fixed some issues.

Add-on Version	Supported Cluster Version	New Feature
1.3.8	v1.23 v1.25 v1.27 v1.28	• Supported pod-based IP address and TCP monitoring of containers. • Supported flow-based IP address and TCP monitoring of containers. • CCE clusters 1.27 are supported. • CCE clusters 1.28 are supported.
1.2.27	v1.19 v1.21 v1.23 v1.25	None
1.2.4	v1.19 v1.21 v1.23 v1.25	• Added the description that only EulerOS is supported.
1.2.2	v1.19 v1.21 v1.23 v1.25	• Supported VPC network health checks in a local pod.
1.1.8	v1.19 v1.21 v1.23 v1.25	• CCE clusters 1.25 are supported.
1.1.6	v1.19 v1.21 v1.23	None
1.1.5	v1.19 v1.21 v1.23	• Optimizes liveness health check.
1.1.2	v1.19 v1.21 v1.23	• Supports wide matching of operating system types.

Add-on Version	Supported Cluster Version	New Feature
1.0.1	v1.19 v1.21	• Supports traffic statistics persistence and local socket communications.

14.3.5 Kubernetes Metrics Server

From version 1.8 onwards, Kubernetes provides resource usage metrics, such as the container CPU and memory usage, through the Metrics API. These metrics can be directly accessed by users (for example, by using the **kubectl top** command) or used by controllers (for example, Horizontal Pod Autoscaler) in a cluster for decision-making. The specific component is metrics-server, which is used to substitute for heapster for providing the similar functions. heapster has been gradually abandoned since v1.11.

metrics-server is an aggregator for monitoring data of core cluster resources. You can quickly install this add-on on the CCE console.

After installing this add-on, you can create HPA policies. For details, see **Creating an HPA Policy**.

The official community project and documentation are available at **https://github.com/kubernetes-sigs/metrics-server**.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Kubernetes Metrics Server** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  The **smaller specification** lacks HA capabilities, while the **large specification** has them. However, deploying multiple pods requires more compute resources.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-57 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 4 Click **Install**.

----End

Components

Table 14-58 Add-on components

Component	Description	Resource Type
metrics-server	Aggregator for the monitored data of cluster core resources, which is used to collect and aggregate resource usage metrics obtained through the Metrics API in the cluster	Deployment

Change History

Table 14-59 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.90	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	**0.6.2**
1.3.68	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	**0.6.2**
1.3.60	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	**0.6.2**

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.39	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	0.6.2
1.3.37	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	0.6.2
1.3.12	v1.19 v1.21 v1.23 v1.25 v1.27	None	0.6.2
1.3.8	v1.19 v1.21 v1.23 v1.25	Synchronized time zones used by the add-on and the node.	0.6.2
1.3.6	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • The default taint tolerance duration is changed to 60s.	0.6.2
1.3.3	v1.19 v1.21 v1.23 v1.25	• CCE clusters 1.25 are supported. • Allowed CronHPA to adjust the number of Deployment pods with the skip scenario supported.	0.6.2
1.3.2	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	0.6.2
1.2.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.	0.4.4

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.1.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	0.4.4
1.1.4	v1.15 v1.17 v1.19	Unified resource specification configuration unit.	0.4.4
1.1.2	v1.15 v1.17 v1.19	Updated the add-on to its community version v0.4.4.	0.4.4
1.1.1	v1.13 v1.15 v1.17 v1.19	Allows you to change the maximum number of invalid pods to 1.	0.3.7
1.1.0	v1.13 v1.15 v1.17 v1.19	CCE clusters 1.19 are supported.	0.3.7
1.0.5	v1.13 v1.15 v1.17	Updated the add-on to its community version v0.3.7.	0.3.7

14.3.6 Grafana

Introduction

Grafana is an open-source visualized data monitoring platform. It provides you with various charts and panels for real-time monitoring, analysis, and visualization of various metrics and data sources.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Grafana** on the right, and click **Install**.

Step 2 Configure **Specifications** of the add-on pods. You can adjust their CPU and memory quotas as needed.

Step 3 Configure the add-on parameters.

Table 14-60 Grafana parameters

Parameter	Description
PVC Type	To install Grafana, create a storage volume to store local data. When Grafana is uninstalled, this storage volume will not be deleted. • If you selected **EVS**, you need to select an EVS disk type. The EVS disk types supported by different regions may vary. You can select one disk type on the management console. EVS disks are charged by storage capacity and occupy your EVS disk quotas. • If you selected **DSS**, you need to select a DSS pool. The created storage volume will be in the corresponding storage pool.
Capacity (GiB)	Expand the disk capacity after the disk is created. The EVS disk size is 5 GiB by default. For details, see **Related Operations**.
Interconnecting Data Sources with AOM	Enable this function to report Prometheus data to AOM. After this function is enabled, you can select the corresponding AOM instance. The collected basic metrics are free of charge. Custom metrics are charged by AOM. For details, see **Product Pricing Details**. To interconnect with AOM, you must have certain permissions. Only **Huawei Cloud accounts, HUAWEI IDs, and users in the admin user group** can perform this operation.
Public Access	This option is available in Grafana 1.2.1 or later versions. After it is enabled, you need to select a load balancer as the Grafana service entry. Only load balancers in the VPC to which the cluster belongs can be selected. If a dedicated load balancer is used, you must configure network specifications for it. **NOTICE** If you allow public access, Grafana services will be exposed to the public network. It is recommended that you assess the security risks and establish access control policies.

Step 4 Configure deployment policies for the add-on pods.

Table 14-61 Configurations for add-on scheduling

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

After the add-on is installed, select the cluster and choose **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 14-62 Add-on components

Component	Description	Resource Type
grafana	This component provides the data visualization capability for Grafana.	Deployment

How to Use

To access Grafana charts through a public network, you need to bind a LoadBalancer service to the Grafana container.

Step 1 Log in to the CCE console, and click the name of the cluster with the Grafana add-on installed to access the cluster console. On the displayed page, choose **Services & Ingresses** in the navigation pane.

Step 2 Click **Create from YAML** in the upper right corner to create a public network LoadBalancer Service for Grafana.

```
apiVersion: v1
kind: Service
metadata:
  name: grafana-lb     # Service name, which is customizable
  namespace: monitoring
  labels:
    app: grafana
  annotations:
    kubernetes.io/elb.id: 038ff***     # Replace it with the ID of the public network load balancer in the VPC that the cluster belongs to.
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80     #Service port, which can be customized.
      targetPort: 3000     # Default Grafana port. Retain the default value.
  selector:
    app: grafana
  type: LoadBalancer
```

Step 3 After the creation, visit ***load balancer public IP:Service port*** to access Grafana and select a proper dashboard to view the aggregated data.

Figure 14-7 Grafana panel

----End

Change History

Table 14-63 Release history

Add-on Version	Supported Cluster Version	New Feature
1.3.1	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Clusters 1.31 are supported.
1.3.0	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Clusters 1.30 are supported.
1.2.1	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Supported association with the LoadBalancer Services.

Add-on Version	Supported Cluster Version	New Feature
1.2.0	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Clusters 1.29 are supported.
1.1.0	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28	Supported the open-source Grafana.

14.3.7 Prometheus (EOM)

Introduction

Prometheus is an open-source system monitoring and alerting framework. It is derived from Google's borgmon monitoring system, which was created by former Google employees working at SoundCloud in 2012. Prometheus was developed as an open-source community project and officially released in 2015. In 2016, Prometheus officially joined the Cloud Native Computing Foundation, after Kubernetes.

CCE allows you to quickly install Prometheus as an add-on.

Official website of Prometheus: **https://prometheus.io/**

Open source community: **https://github.com/prometheus/prometheus**

Notes and Constraints

The Prometheus add-on is supported only in clusters of v1.21 and earlier.

Features

As a next-generation monitoring framework, Prometheus has the following features:

- Powerful multi-dimensional data model

    a. Time series data is identified by metric name and key-value pair.
    b. Multi-dimensional labels can be set for all metrics.
    c. Data models do not require dot-separated character strings.
    d. Data models can be aggregated, cut, and sliced.
    e. The double floating-point format is supported. Labels can all be set to unicode.
- Flexible and powerful query statement (PromQL): One query statement supports addition, multiplication, and connection for multiple metrics.
- Easy to manage: The Prometheus server is a separate binary file that can work locally. It does not depend on distributed storage.
- Efficient: Each sampling point occupies only 3.5 bytes, and one Prometheus server can process millions of metrics.
- The pull mode is used to collect time series data, which facilitates local tests and prevents faulty servers from pushing bad metrics.
- Time series data can be pushed to the Prometheus server in push gateway mode.
- Users can obtain the monitored targets through service discovery or static configuration.
- Multiple visual GUIs are available.
- Easy to scale

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Prometheus** on the right, and click **Install**.

Step 2 In the **Configuration** step, set the following parameters:

Table 14-64 Prometheus add-on parameters

Parameter	Description
Add-on Specifications	Select an add-on specification based on service requirements. The options are as follows: • **Demo(<= 100 containers)**: The specification type applies to the experience and function demonstration environment. In this specification, Prometheus occupies few resources but has limited processing capabilities. You are advised to use this specification when the number of containers in the cluster does not exceed 100. • **Small(<= 2000 containers)**: You are advised to use this specification when the number of containers in the cluster does not exceed 2,000. • **Medium(<= 5000 containers)**: You are advised to use this specification when the number of containers in the cluster does not exceed 5000. • **Large(> 5000 containers)**: You are advised to use this specification when the number of containers in the cluster exceeds 5,000.
Pods	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Data Retention (days)	Number of days for storing customized monitoring data. The default value is 15 days.
Storage	Cloud hard disks can be used as storage. Set the following parameters as prompted: • **AZ**: Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. • **Disk Type**: Common I/O, high I/O, and ultra-high I/O are supported. • **Capacity**: Enter the storage capacity based on service requirements. The default value is 10 GB. **NOTE** If a PVC already exists in the namespace monitoring, the configured storage will be used as the storage source.

Step 3 Click **Install**. After the installation, the add-on deploys the following instances in the cluster.

- prometheus-operator: deploys and manages the Prometheus Server based on CustomResourceDefinitions (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.
- prometheus (server): a Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.
- prometheus-kube-state-metrics: converts the Prometheus metric data into a format that can be identified by Kubernetes APIs.
- custom-metrics-apiserver: aggregates custom metrics to the native Kubernetes API server.
- prometheus-node-exporter: deployed on each node to collect node monitoring data.
- grafana: visualizes monitoring data.

----End

Providing Resource Metrics Through the Metrics API

Resource metrics of containers and nodes, such as CPU and memory usage, can be obtained through the Kubernetes Metrics API. Resource metrics can be directly accessed, for example, by using the **kubectl top** command, or used by HPA or CustomedHPA policies for auto scaling.

The add-on can provide the Kubernetes Metrics API that is disabled by default. To enable the API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If information similar to the following is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                        CPU(cores)   MEMORY(bytes)
......
custom-metrics-apiserver-d4f556ff9-l2j2m                38m          44Mi
......
```


NOTICE

To uninstall the add-on, run the following kubectl command and delete the APIService object. Otherwise, the metrics-server add-on cannot be installed due to residual APIService resources.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Reference

- For details about the Prometheus concepts and configurations, see the **Prometheus Official Documentation**.
- For details about how to install Node Exporter, see the **node_exporter GitHub**.

Change History

Table 14-65 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.23.32	v1.17 v1.19 v1.21	None	**2.10.0**
2.23.31	v1.15	• CCE clusters 1.15 are supported.	**2.10.0**
2.23.30	v1.17 v1.19 v1.21	• CCE clusters 1.21 are supported.	**2.10.0**
2.21.14	v1.17 v1.19 v1.21	• CCE clusters 1.21 are supported.	**2.10.0**
2.21.12	v1.15	• CCE clusters 1.15 are supported.	**2.10.0**
2.21.11	v1.17 v1.19	• CCE clusters 1.19 are supported.	**2.10.0**
1.15.1	v1.15 v1.17	• The add-on is a monitoring system and time series library.	**2.10.0**

14.4 Cloud Native Heterogeneous Computing Add-ons

14.4.1 CCE AI Suite (NVIDIA GPU)

Introduction

NVIDIA GPU is a device management add-on that supports GPUs in containers. To use GPU nodes in a cluster, this add-on must be installed.

Notes and Constraints

- The driver to be downloaded must be a **.run** file.
- Only NVIDIA Tesla drivers are supported, not GRID drivers.
- When installing or reinstalling the add-on, ensure that the driver download address is correct and accessible. CCE does not verify the address validity.
- The gpu-beta add-on only enables you to download the driver and execute the installation script. The add-on status only indicates that how the add-on is running, not whether the driver is successfully installed.
- CCE does not guarantee the compatibility between the GPU driver version and the CUDA library version of your application. You need to check the compatibility by yourself.
- If a custom OS image has had a a GPU driver installed, CCE cannot ensure that the GPU driver is compatible with other GPU components such as the monitoring components used in CCE.
- If the version of the GPU driver you used is not included in the **Supported GPU Drivers**, the GPU driver may be incompatible with the OS, instance type, or container runtime. As a result, the driver installation may fail or the GPU add-on may be abnormal. If you use a customized GPU driver, verify its availability.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE AI Suite (NVIDIA GPU)** on the right, and click **Install**.

Step 2 Determine whether to enable **Use DCGM-Exporter to Observe DCGM Metrics**. After this function is enabled, DCGM-Exporter is deployed on the GPU node.

After DCGM-Exporter is enabled, if the collected GPU monitoring data needs to be reported to AOM, you need to install the Cloud Native Cluster Monitoring add-on and enable the function of reporting monitoring data to AOM. Then, you can choose **Settings** in the navigation pane, click the **Monitoring** tab, and enable the **ServiceMonitor** of DCGM-Exporter. GPU metrics reported to AOM are custom metrics and will be billed on a pay-per-use basis. For details, see **Price Calculator**.

NOTICE

If the add-on version is 2.7.40 or later, DCGM-Exporter can be deployed. DCGM-Exporter maintains the community capability and does not support the sharing mode or GPU virtualization.

Step 3 Configure the add-on parameters.

Table 14-66 Add-on parameters

Parameter	Description
Default Cluster Driver	All GPU nodes in a cluster use the same driver. You can select a proper GPU driver version or customize the driver link and enter the download link of the NVIDIA driver. **NOTICE** • If the download link is a public network address, for example, **https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run**, bind an EIP to each GPU node. For details about how to obtain the driver link, see **Obtaining the Driver Link from Public Network**. • If the download link is an OBS URL, you do not need to bind an EIP to GPU nodes. For details about how to obtain the driver link, see **Obtaining the Driver Link from OBS**. • Ensure that the NVIDIA driver version matches the GPU node. For details about the version mapping, see **Supported GPU Drivers**. • If the driver version is changed, restart the node to apply the change. • Use driver version 470 or later for Huawei Cloud EulerOS 2.0 on which Linux Kernel 5.x is built, and driver 515 or later for Ubuntu 22.04.

NOTE

After the add-on is installed, you can configure GPU virtualization and node pool drivers on the **Heterogeneous Resources** tab in **Settings**.

Step 4 Click **Install**.

NOTE

If the add-on is uninstalled, GPU pods newly scheduled to the nodes cannot run properly, but GPU pods already running on the nodes will not be affected.

----End

Verifying the Add-on

After the add-on is installed, run the **nvidia-smi** command on the GPU node and the container that schedules GPU resources to verify the availability of the GPU device and driver.

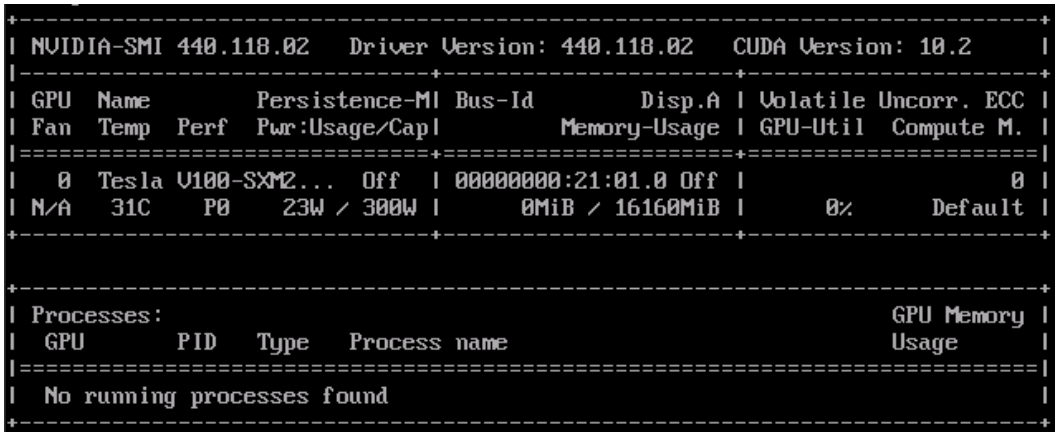
- GPU node:

```
# If the add-on version is earlier than 2.0.0, run the following command:
cd /opt/cloud/cce/nvidia/bin && ./nvidia-smi

# If the add-on version is 2.0.0 or later and the driver installation path is changed, run the following command:
cd /usr/local/nvidia/bin && ./nvidia-smi
```

- Container:


```
cd /usr/local/nvidia/bin && ./nvidia-smi
```

If GPU information is returned, the device is available and the add-on has been installed.



Supported GPU Drivers

NOTICE

- The list of supported GPU drivers applies only to GPU add-ons of 1.2.28 and later versions.
- If you want to use the latest GPU driver, upgrade your GPU add-on to the latest version.

Table 14-67 Supported GPU drivers

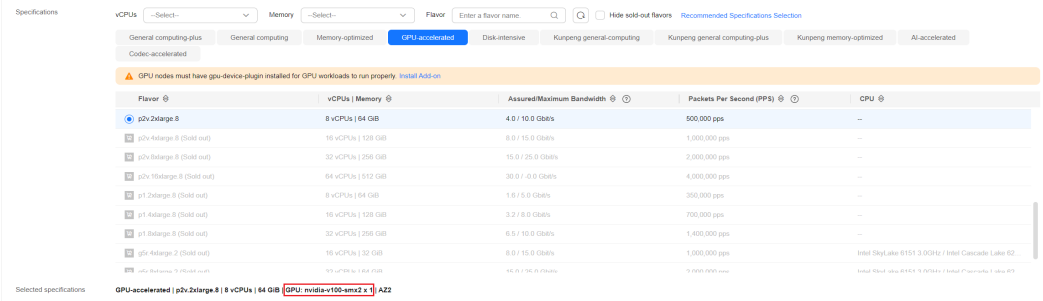
GPU Model	Supported Cluster Type	Specification	OS							
			Huawei Cloud EulerOS 2.0 (GPU Virtualization Supported)	Ubuntu 22.04.4	Ubuntu 22.04.3	CentOS Linux release 7.6	EulerOS release 2.9	EulerOS release 2.5	Ubuntu 18.04 (end of maintenance)	EulerOS release 2.3 (end of maintenance)
Tesla T4	CCE standard cluster	g6 pi2	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03
Volta V100	CCE standard cluster	p2s p2vs p2v	535.54.03 510.47.03 470.57.02	535.161.08	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03

Obtaining the Driver Link from Public Network

Step 1 Log in to the CCE console.

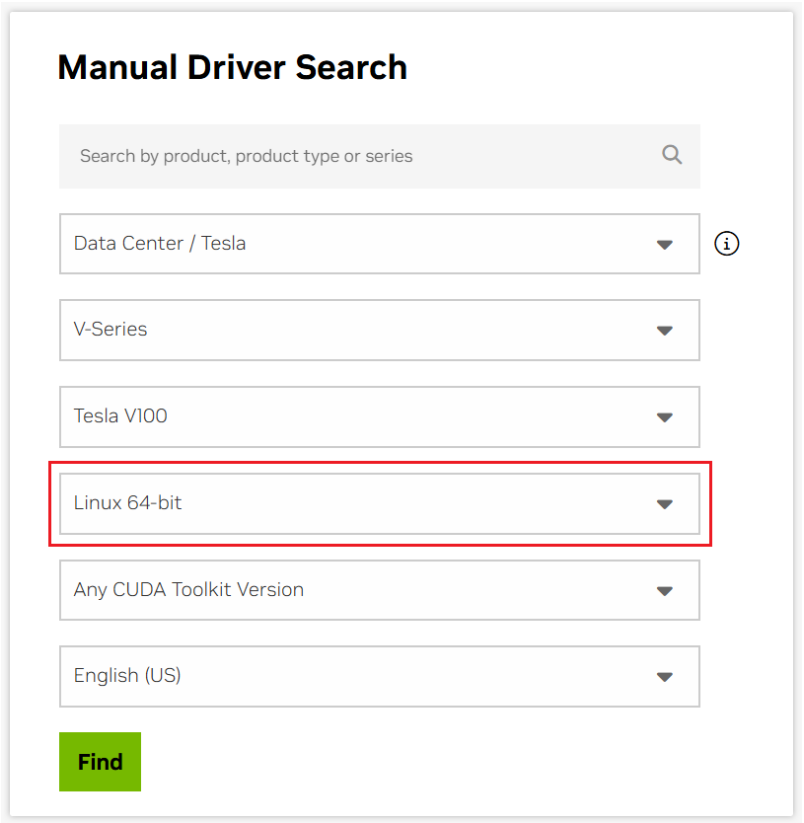
Step 2 Create a node. In the **Specifications** area, select the GPU node flavor. The GPU card models are displayed in the lower part of the area.

Figure 14-8 Viewing the GPU card model



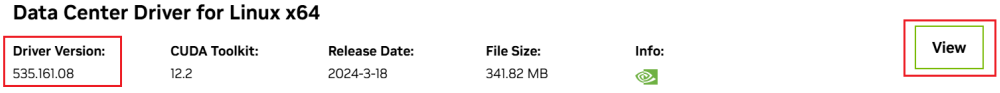
Step 3 Log in to the **NVIDIA driver download page** and search for the driver information. The OS must be **Linux 64-bit**.

Figure 14-9 Selecting parameters



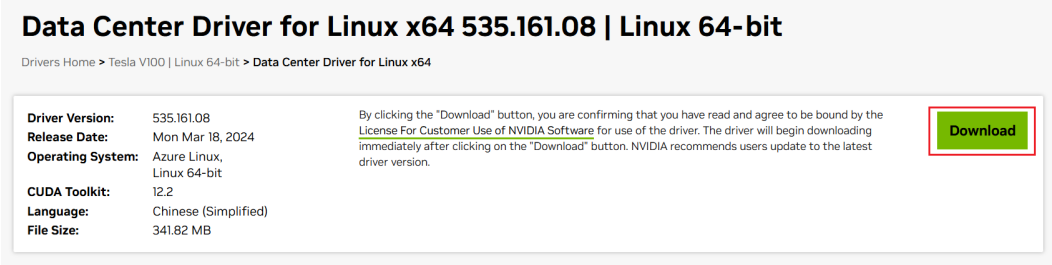
Step 4 After confirming the driver information, click **Find**. On the displayed page, find the driver to be downloaded and click **View**.

Figure 14-10 Viewing the driver information



Step 5 Click **Download** and copy the download link.

Figure 14-11 Obtaining the link



----End

Obtaining the Driver Link from OBS

Step 1 Upload the driver to OBS and set the driver file to public read. For details, see **Uploading an Object**.

NOTE

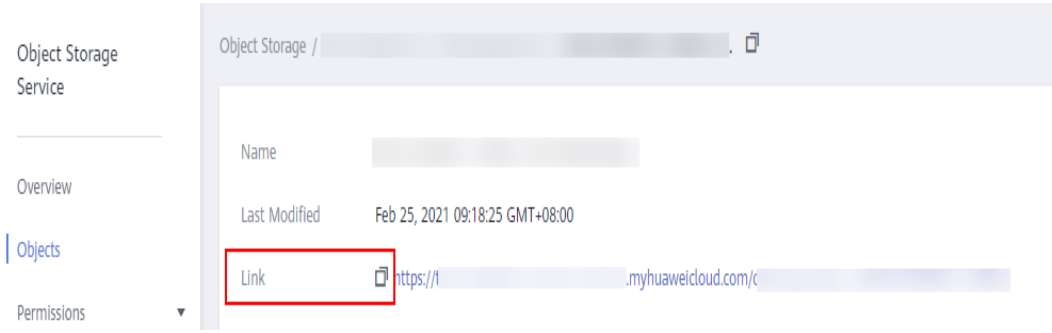
When the node is restarted, the driver will be downloaded and installed again. Ensure that the OBS bucket link of the driver is valid.

Step 2 In the bucket list, click a bucket name, and then the **Overview** page of the bucket is displayed.

Step 3 In the navigation pane, choose **Objects**.

Step 4 Select the name of the target object and copy the driver link on the object details page.

Figure 14-12 Copying an OBS link



----End

Components

Table 14-68 Add-on components

Component	Description	Resource Type
nvidia-driver-installer	A workload for installing the NVIDIA GPU driver on a node, which only uses resources during the installation process (Once the installation is finished, no resources are used.)	DaemonSet

Component	Description	Resource Type
nvidia-gpu-device-plugin	A Kubernetes device plugin that provides NVIDIA GPU heterogeneous compute for containers	DaemonSet
nvidia-operator	A component that provides NVIDIA GPU node management capabilities for clusters	Deployment

GPU Metrics

For more details, see **GPU Metrics**.

Helpful Links

- **How Do I Rectify Failures When the NVIDIA Driver Is Used to Start Containers on GPU Nodes?**
- **What Should I Do If GPU Node Exceptions Occur?**
- **GPU Scheduling**

Change History

Table 14-69 Release history

Add-on Version	Supported Cluster Version	New Feature
2.7.42	v1.28 v1.29 v1.30 v1.31	The NVIDIA 535.216.03 driver is added to support xGPUs.
2.7.41	v1.28 v1.29 v1.30 v1.31	The NVIDIA 535.216.03 driver is added to support xGPUs.
2.7.40	v1.28 v1.29 v1.30 v1.31	Integrated with DCGM-Exporter to observe the DCGM metrics of NVIDIA GPU nodes in clusters.
2.7.19	v1.28 v1.29 v1.30	Fixed the nvidia-container-toolkit CVE-2024-0132 container escape vulnerability.

Add-on Version	Supported Cluster Version	New Feature
2.7.13	v1.28 v1.29 v1.30	• Supported xGPU configuration by node pool. • Supported GPU rendering. • Clusters 1.30 are supported.
2.6.4	v1.28 v1.29	Updated the isolation logic of GPU cards.
2.6.1	v1.28 v1.29	Upgraded the base images of the add-on.
2.5.6	v1.28	Fixed an issue that occurred during the installation of the driver.
2.5.4	v1.28	Clusters 1.28 are supported.
2.1.24	v1.21 v1.23 v1.25 v1.27	Added xGPU data to GPU basic metrics.
2.1.14	v1.21 v1.23 v1.25 v1.27	Fixed the nvidia-container-toolkit CVE-2024-0132 container escape vulnerability.
2.1.8	v1.21 v1.23 v1.25 v1.27	Fixed some issues.
2.0.69	v1.21 v1.23 v1.25 v1.27	Upgraded the base images of the add-on.
2.0.46	v1.21 v1.23 v1.25 v1.27	• Supported Nvidia driver 535. • Non-root users can use xGPUs. • Optimized startup logic.
2.0.18	v1.21 v1.23 v1.25 v1.27	Supported Huawei Cloud EulerOS 2.0.

Add-on Version	Supported Cluster Version	New Feature
1.2.28	v1.19 v1.21 v1.23 v1.25	• Adapted to Ubuntu 22.04. • Optimized the automatic mounting of the GPU driver directory.
1.2.24	v1.19 v1.21 v1.23 v1.25	• Enabled a node pool to configure GPU driver versions. • Supported GPU metric collection.
1.2.20	v1.19 v1.21 v1.23 v1.25	Set the add-on alias to **gpu**.
1.2.17	v1.15 v1.17 v1.19 v1.21 v1.23	Added the nvidia-driver-install pod limits configuration.
1.2.15	v1.15 v1.17 v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.11	v1.15 v1.17 v1.19 v1.21	Supported EulerOS 2.10.
1.2.10	v1.15 v1.17 v1.19 v1.21	CentOS supports the GPU driver of the new version.
1.2.9	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.2.2	v1.15 v1.17 v1.19	Supported the new EulerOS kernel.
1.2.1	v1.15 v1.17 v1.19	• CCE clusters 1.19 are supported. • Added taint tolerance configuration.
1.1.13	v1.13 v1.15 v1.17	Supported kernel-3.10.0-1127.19.1.el7.x86_64 for CentOS 7.6.
1.1.11	v1.15 v1.17	• Allowed users to customize driver addresses to download drivers. • Clusters 1.15 and 1.17 are supported.

14.4.2 CCE AI Suite (Ascend NPU)

Introduction

CCE AI Suite (Ascend NPU) supports and manages NPUs in containers.

After this add-on is installed, you can create Ascend-accelerated nodes to quickly and efficiently process inference and image recognition.

Notes and Constraints

- To use Ascend-accelerated nodes in a cluster, the Ascend NPU add-on must be installed.
- After an AI-accelerated node is migrated, the node will be reset. Manually reinstall the NPU driver.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE AI Suite (Ascend NPU)** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed. You can adjust the number of add-on instances and resource quotas as required.

Step 3 Determine whether to enable **Auto Driver Installation** (supported only when the add-on version is 1.2.5 or later).

- Enabled: You can specify the driver version based on the NPU model for easier driver maintenance.

After the driver is enabled, the add-on automatically installs the driver based on the specified driver version. By default, the recommended driver is used. You can also select **Path to a custom driver** from the drop-down list and enter a driver address.

NOTE

- The add-on installs the driver based on the driver version selected for the specified model. Such installation is only for nodes with no NPU driver installed. Nodes with an NPU driver installed remain unchanged. If you change the driver version when upgrading the add-on or updating add-on parameters, such change takes effect only on the nodes with no NPU driver installed.
- After the driver is successfully installed, restart the node for the driver to take effect. For details about how to check whether the driver is successfully installed, see **How to Check Whether the NPU Driver Has Been Installed on a Node**.
- Uninstalling the add-on does not automatically delete the installed NPU driver. For details about how to uninstall the NPU driver, see **Uninstalling the NPU Driver**.

- Disabled: Driver versions are decided by the system, and the drivers cannot be maintained using the add-on. When you add an NPU node on the console, the system adds the command to install an NPU driver (version and type decided by the system) and automatically restarts the node after the drive installation is complete. Adding an NPU node in another way, such as using an API, requires you to add the driver installation command to **Post-installation Command**.
- The supported NPU types and OSs are as follows:

NPU Type	Supported OS
D310	EulerOS 2.5 x86, CentOS 7.6 x86, EulerOS 2.9 x86, and EulerOS 2.8 arm

Step 4 Click **Install**.

----End

Components

Table 14-70 Add-on components

Component	Description	Resource Type
npu-driver-installer	Used for installing an NPU driver on NPU nodes.	DaemonSet
huawei-npu-device-plugin	Allowed containers to use Huawei NPUs.	DaemonSet

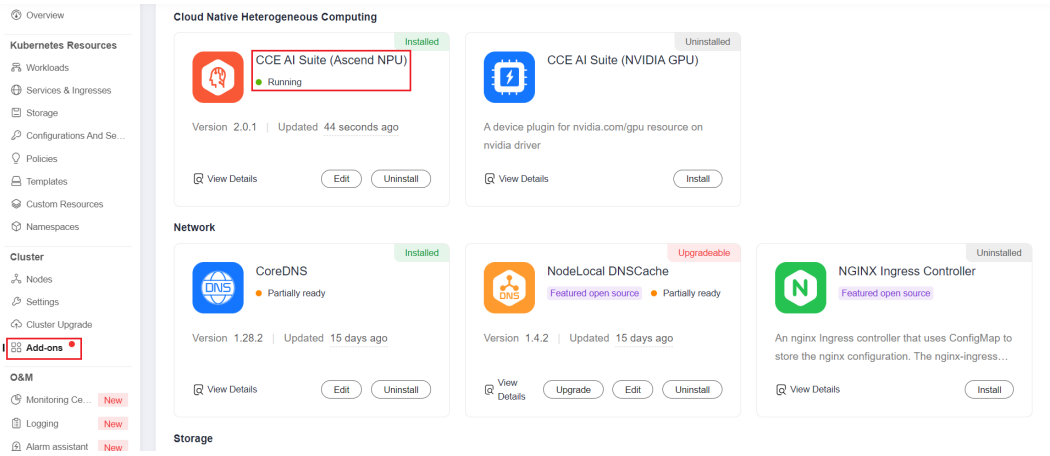
NPU Metrics

Metric	Monitoring Level	Remarks
cce_npu_memory_total	NPU cards	Total NPU memory
cce_npu_memory_used	NPU cards	NPU memory usage
cce_npu_utilization	NPU cards	NPU compute usage

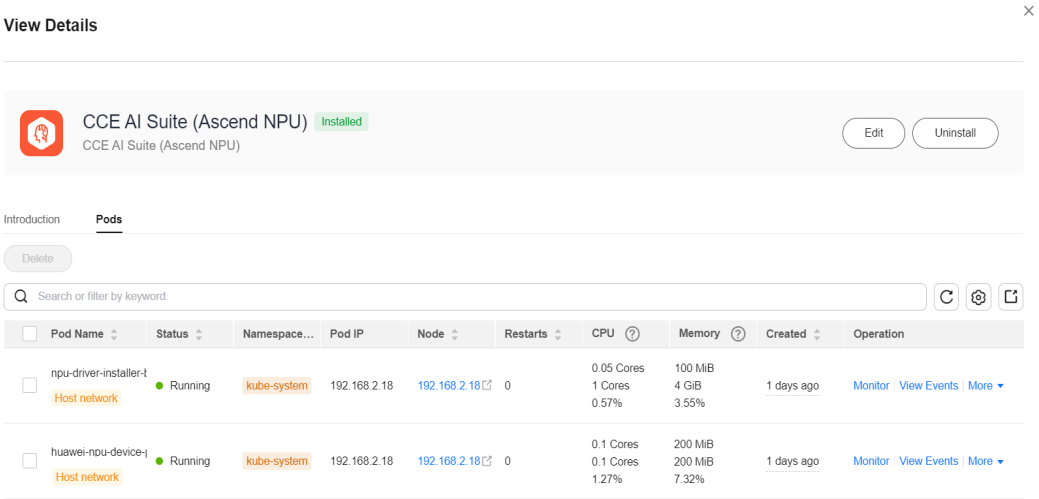
How to Check Whether the NPU Driver Has Been Installed on a Node

After ensuring that the driver is successfully installed, restart the node for the driver to take effect. Otherwise, the driver cannot take effect and NPU resources are unavailable. To check whether the driver is installed, perform the following operations:

Step 1 On the **Add-ons** page, click **CCE AI Suite (Ascend NPU)**.



Step 2 Verify that the node where npu-driver-installer is deployed is in the **Running** state.



NOTE

If the node is restarted before the NPU driver is installed, the driver installation may fail, and a message is displayed on the **Nodes** page indicating that the driver is not ready. In this case, uninstall the NPU driver from the node and restart the **npu-driver-installer** pod to reinstall the NPU driver. After confirming that the driver is installed, restart the node. For details about how to uninstall the driver, see **Uninstalling the NPU Driver**.

----End

Uninstalling the NPU Driver

Log in to the node, obtain the driver operation records in the **/var/log/ascend_seclog/operation.log** file, and find the driver run package used in last installation. If the lof file does not exist, the driver is installed using the **npu_x86_latest.run** or **npu_arm_latest.run** driver combined package. After finding the driver installation package, run the **bash {run *package name*} --uninstall** command to uninstall the driver and restart the node as prompted.

Step 1 Log in to the node where the NPU driver needs to be uninstalled and find the **/var/log/ascend_seclog/operation.log** file.

Step 2 If the **/var/log/ascend_seclog/operation.log** file can be found, view the driver installation log to find the driver installation record.

```
[root@y00379955-w-ai1s-e25 ~]# ll /var/log/ascend_seclog/operation.log
-rw-r----- 1 root root 285 Dec  1 20:00 /var/log/ascend_seclog/operation.log
[root@y00379955-w-ai1s-e25 ~]# cat /var/log/ascend_seclog/operation.log
[Install] SUGGESTION root 2022-12-01 19:53:47 127.0.0.1 Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run success install_type=full; cmdlist=--quiet --full.
```

If the **/var/log/ascend_seclog/operation.log** file cannot be found, the driver may be installed using the **npu_x86_latest.run** or **npu_arm_latest.run** driver combined package. You can confirm this by checking whether the **/usr/local/HiAI/driver/** directory exists.

NOTE

The combined package of the NPU driver is stored in the **/root/d310_driver** directory, and other driver installation packages are stored in the **/root/npu-drivers** directory.

Step 3 After finding the driver installation package, run the **bash {run *package path*} --uninstall** command to uninstall the driver. The following uses **Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run** as an example:

```
bash /root/npu-drivers/Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
```

```
[root@y00379955-w-ai1s-e25 npu-drivers]# ./Ascend310-hdk-npu-driver_6.0.rc1_linux-x86-64.run --uninstall
Verifying archive integrity...  100%   SHA256 checksums are OK. All good.
Uncompressing ASCEND DRIVER RUN PACKAGE  100%
[Driver] [2022-12-01 19:59:53] [INFO]Start time: 2022-12-01 19:59:53
[Driver] [2022-12-01 19:59:53] [INFO]LogFile: /var/log/ascend_seclog/ascend_install.log
[Driver] [2022-12-01 19:59:53] [INFO]OperationLogFile: /var/log/ascend_seclog/operation.log
[Driver] [2022-12-01 19:59:53] [INFO]base version is 22.0.3.


[Driver] [2022-12-01 20:00:04] [INFO]Driver package uninstalled successfully! Reboot needed for uninstallation to take effect!
[Driver] [2022-12-01 20:00:04] [INFO]End time: 2022-12-01 20:00:04
```

Step 4 Restart the node as prompted. (The installation and uninstallation of the current NPU driver take effect only after the node is restarted.)

----End

Change History

Table 14-71 Release history

Add-on Version	Supported Cluster Version	New Feature
2.1.46	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.
2.1.23	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Fixed some issues.
2.1.22	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• Fixed display issues on some pages. • Supernodes can be obtained. • NPU topology can be reported. • Resolved log printing issues.
2.1.14	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Fixed some issues.

Add-on Version	Supported Cluster Version	New Feature
2.1.7	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Resolved the issue that npu-smi fails to be automatically mounted to a service container.
2.1.5	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• CCE clusters 1.29 are supported. • Added silent fault codes.
2.0.9	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed the issue that process-level fault recovery and annotation adding to workloads occasionally fail.
2.0.5	v1.21 v1.23 v1.25 v1.27 v1.28	• CCE clusters 1.28 are supported. • Supported liveness probe. • Ascend drivers can be automatically mounted to service containers.
1.2.14	v1.19 v1.21 v1.23 v1.25 v1.27	Supported NPU monitoring.
1.2.6	v1.19 v1.21 v1.23 v1.25	Supported automatic installation of NPU drivers.
1.2.5	v1.19 v1.21 v1.23 v1.25	Supported automatic installation of NPU drivers.

Add-on Version	Supported Cluster Version	New Feature
1.2.4	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.
1.2.2	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.2.1	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.1.8	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.
1.1.2	v1.15 v1.17 v1.19	Added the default seccomp profile.
1.1.1	v1.15 v1.17 v1.19	CCE clusters 1.15 are supported.
1.1.0	v1.17 v1.19	CCE clusters 1.19 are supported.
1.0.8	v1.13 v1.15 v1.17	Adapted to the D310 C75 driver.
1.0.6	v1.13 v1.15 v1.17	Supported the C75 driver.
1.0.5	v1.13 v1.15 v1.17	Allowed containers to use Huawei NPUs.
1.0.3	v1.13 v1.15 v1.17	Allowed containers to use Huawei NPUs.

14.5 Container Network Add-ons

14.5.1 CoreDNS

Introduction

CoreDNS is a DNS server that provides domain name resolution for Kubernetes clusters through a chain add-on.

CoreDNS is an open-source software and has been a part of CNCF. It provides a means for cloud services to discover each other in cloud-native deployments. Each of the plugins chained by CoreDNS provides a particular DNS function. You can integrate CoreDNS with only the plugins you need to make it fast, efficient, and flexible. When used in a Kubernetes cluster, CoreDNS can automatically discover services in the cluster and provide domain name resolution for these services. By working with DNS server, CoreDNS can resolve external domain names for workloads in a cluster.

This add-on is installed by default during cluster creation.

Kubernetes backs CoreDNS as the official default DNS for all clusters going forward.

CoreDNS official website: **https://coredns.io/**

Open source community: **https://github.com/coredns/coredns**

NOTE

For details, see **DNS**.

Notes and Constraints

To run CoreDNS properly or upgrade CoreDNS in a cluster, ensure the number of available nodes in the cluster is greater than or equal to the number of CoreDNS instances and all CoreDNS instances are running. Otherwise, the add-on will malfunction or the upgrade will fail.

Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CoreDNS** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small qps**, **Medium qps**, or **Large qps** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

The small one can handle up to 2500 external and 10,000 internal domain names QPS. The medium specification can handle up to 5000 external and 20,000 internal domain names QPS. The large specification can handle up to 10,000 external and 40,000 internal domain names QPS.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. QPS of the CoreDNS add-on is positively correlated with the CPU consumption. If the number of nodes or containers in the cluster grows, the CoreDNS pods will bear heavier workloads. It is recommended that you adjust the number of the CoreDNS pods and their CPU and memory quotas based on the cluster scale. For details, see **Table 14-72**.

Table 14-72 Recommended CoreDNS quotas

Nodes	Recommended QPS	Pods	Requested vCPUs	vCPU Limit	Requested Memory	Memory Limit
50	2500	2	500m	500m	512 MiB	512 MiB
200	5000	2	1000m	1000m	1024 MiB	1024 MiB
1000	10000	2	2000m	2000m	2048 MiB	2048 MiB
2000	20000	4	2000m	2000m	2048 MiB	2048Mi

Step 3 Configure the add-on parameters.

Table 14-73 Add-on parameters

Parameter	Description
Stub Domain	A domain name server for a custom domain name. The format is a key-value pair. The key is a domain name suffix, and the value is one or more DNS IP addresses, for example, **acme.local -- 1.2.3.4,6.7.8.9**. For details, see **Configuring the Stub Domain for CoreDNS**.

Parameter	Description

Extended Parameter Settings	• **parameterSyncStrategy**: indicates whether to configure consistency check when the add-on is upgraded.
	– **ensureConsistent**: indicates that the configuration consistency check is enabled. If the delivered configuration differs from the current effective configuration, the current effective configuration will be replaced. However, if the delivered configuration is the same as the current effective configuration, the current effective configuration will be preserved. The **ensureConsistent** parameter ensures the configuration consistency. If a ConfigMap is modified manually, the add-on cannot be upgraded. In such cases, you will need to use the **force** or **inherit** policy to upgrade the add-on.
	– **force**: indicates that the configuration consistency check is ignored during an upgrade. The configuration provided during the add-on upgrade will be used, so it is important to make sure that it matches the current effective configuration. After the add-on is upgraded, you need to restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	– **inherit**: If the configuration provided during the add-on upgrade differs from the current effective configuration, the current effective configuration will be used instead. After the add-on is upgraded, you need to restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	• **servers**: nameservers, which are available in CoreDNS v1.23.1 and later versions. You can customize nameservers. For details, see **dns-custom-nameservers**.
	plugins indicates the configuration of each component in CoreDNS. Retain the default settings typically to prevent CoreDNS from being unavailable due to configuration errors. Each plugin component contains **name**, **parameters** (optional), and **configBlock** (optional). The format of the generated Corefile is as follows:
	`$name  $parameters {`
	`$configBlock`
	`}`
	Table 14-74 describes common plugins. For details, see **Plugins**.
	• **upstream_nameservers**: specifies the IP address of the upstream DNS server.
	Example:
	`{`
	`"annotations": {},`
	`"parameterSyncStrategy": "ensureConsistent",`
	`"servers": [`
	`{`
	`"plugins": [`
	`{`

Parameter	Description

	"name": "bind",
	"parameters": "{$POD_IP}"
	},
	{
	"name": "cache",
	"configBlock": "servfail 5s",
	"parameters": 30
	},
	{
	"name": "errors"
	},
	{
	"name": "health",
	"parameters": "{$POD_IP}:8080"
	},
	{
	"name": "ready",
	"parameters": "{$POD_IP}:8081"
	},
	{
	"configBlock": "pods insecure\nfallthrough in-addr.arpa ip6.arpa",
	"name": "kubernetes",
	"parameters": "cluster.local in-addr.arpa ip6.arpa"
	},
	{
	"name": "loadbalance",
	"parameters": "round_robin"
	},
	{
	"name": "prometheus",
	"parameters": "{$POD_IP}:9153"
	},
	{
	"configBlock": "policy random",
	"name": "forward",
	"parameters": ". /etc/resolv.conf"
	},
	{
	"name": "reload"
	}
	],
	"port": 5353,
	"zones": [
	{
	"zone": "."
	}
	]
	}
	],
	"upstream_nameservers": ["8.8.8.8", "8.8.4.4"]
	}

Table 14-74 Default CoreDNS configurations

Plugin Name	Type	Description
bind	Default configuration	Host IP address listened by CoreDNS. Retain the default value **{$POD_IP}**. For details, see **bind**.

Plugin Name	Type	Description
cache	Default configuration	Enables DNS cache. For details, see **cache**. If the add-on version is 1.25.10 or later, the servfail cache can be disabled. To disable the servfail cache, set **configBlock** to **servfail 0**. Otherwise, the unit of the servfail cache is second and cannot be omitted.
errors	Default configuration	Errors are logged to stdout. For details, see **errors**.
health	Default configuration	Health check for CoreDNS. {$POD_IP}:8080 is listened to. Retain the default setting. Otherwise, the CoreDNS health check will fail and the add-on will restart repeatedly. For details, see **health**.
ready	Default configuration	Whether the backend server is ready to receive traffic. {$POD_IP}:8081 is listened to. If the backend server is not ready, CoreDNS will suspend DNS resolution until the backend server is ready. For details, see **ready**.
kubernetes	Default configuration	CoreDNS Kubernetes plugin, which provides the service parsing capability in a cluster. For details, see **kubernetes**.
loadbalance	Default configuration	Round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in an answer. For details, see **loadbalance**.
prometheus	Default configuration	API for obtaining CoreDNS metrics. {$POD_IP}:9153 is listened to by default. Retain the default setting. Otherwise, Prometheus cannot collect CoreDNS metrics. For details, see **Prometheus**.
forward	Default configuration	Forwards any queries that are not within the cluster domain of Kubernetes to predefined resolvers (**/etc/resolv.conf**). For details, see **forward**.
reload	Default configuration	Automatically reloads modified Corefiles. After you modify a ConfigMap, wait for two minutes for the modification to take effect. For details, see **reload**.

log	Extended configuration	Enables CoreDNS logging. For details, see **log**.
		Example:
		`{`
		`  "name": "log"`
		`}`

Plugin Name	Type	Description

template	Extended configuration	A quick response template, where **AAAA** indicates an IPv6 request. If **NXDOMAIN** is returned in an **rcode** response, no IPv6 resolution result is returned. For details, see **template**.
		Example:
		{
		"configBlock": "rcode NXDOMAIN",
		"name": "template",
		"parameters": "ANY AAAA"
		}

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-75 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Configure CoreDNS Using Corefile

NOTE

If you install the CoreDNS add-on, the Corefile view configuration is not available. This configuration is supported only when you are editing or upgrading the add-on.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CoreDNS** on the right, and click **Edit**.

Step 2 In the **Parameters** area, select whether to switch to the **Corefile View** (supported by add-on 1.30.3 and later versions).

Once the function is enabled, the ConfigMap of CoreDNS in the kube-system namespace will be directly configured in the Corefile format. Any existing stub domain configurations and parameters such as **parameterSyncStrategy**, **servers**,

and **upstream_nameservers** in the advanced configuration will no longer be in effect. It is important to verify that the Corefile configuration is accurate.

For description of the Corefile format, see **Configuration**.

NOTE

- Once the Corefile view is disabled, the ConfigMap of CoreDNS will continue to be configured based on to the stub domain configurations and parameters such as **parameterSyncStrategy**, **servers**, and **upstream_nameservers** in the advanced configuration. It is important to verify that the configuration is correct during the function switchover.
- Once the Corefile view is enabled, the add-on can be upgraded. However, if the Corefile view is disabled again, the add-on upgrade will override the current configurations. To complete the upgrade, **parameterSyncStrategy** must be set to either **force** or **inherit**.
- Once the Corefile configuration is modified, simply wait for CoreDNS' reload mechanism to automatically update the configuration. This typically takes about 10 seconds for the changes to take effect.

Step 3 After editing the Corefile, click **OK**.

----End

Components

Table 14-76 Add-on components

Component	Description	Resource Type
CoreDNS	DNS server for clusters	Deployment

How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be configured for each pod. Kubernetes supports DNS policies **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. For details, see **DNS for Services and Pods**. These policies are specified in the **dnsPolicy** field in the pod-specific.

- **Default**: Pods inherit the name resolution configuration from the node that the pods run on. The custom upstream DNS server and the stub domain cannot be used together with this policy.
- **ClusterFirst**: Any DNS query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream name server inherited from the node. Cluster administrators may have extra stub domains and upstream DNS servers configured.
- **ClusterFirstWithHostNet**: For pods running with hostNetwork, set its DNS policy **ClusterFirstWithHostNet**.
- **None**: It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the **dnsPolicy** field in the pod-specific.

NOTE

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

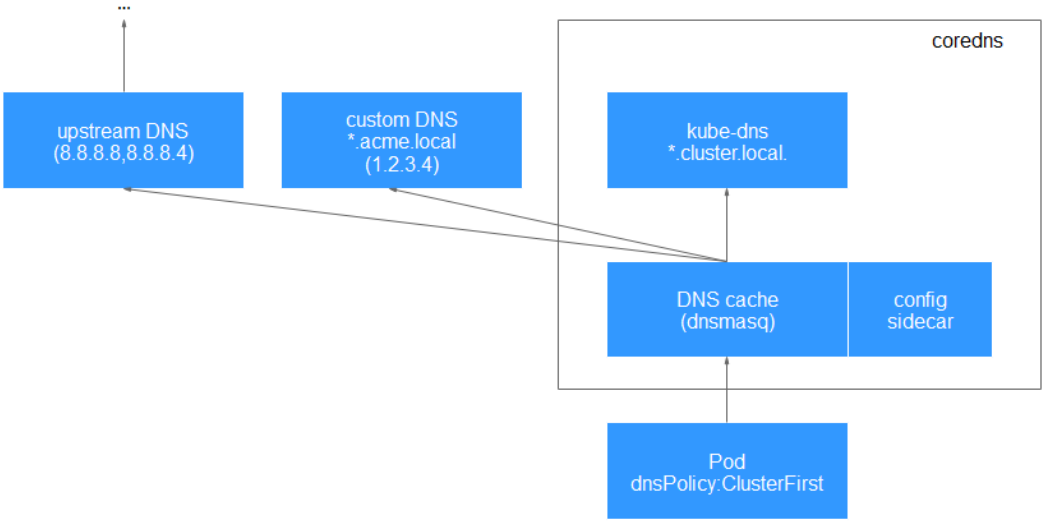
Routing

Without stub domain configurations: Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

With stub domain configurations: If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in CoreDNS.
2. From the caching layer, the suffix of the request is examined and then the request is forwarded to the corresponding DNS:
   - Names with the cluster suffix, for example, **.cluster.local**: The request is sent to CoreDNS.
   - Names with the stub domain suffix, for example, **.acme.local**: The request is sent to the configured custom DNS resolver that listens, for example, on 1.2.3.4.
   - Names that do not match the suffix (for example, **widget.com**): The request is forwarded to the upstream DNS.

Figure 14-13 Routing



Change History

Table 14-77 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.30.29	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	1.10.1
1.30.6	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	• Supported Corefile configurations. • CCE clusters 1.30 are supported.	1.10.1
1.29.5	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.	1.10.1
1.29.4	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	1.10.1
1.28.7	v1.21 v1.23 v1.25 v1.27 v1.28	Supported hot module replacement. Rolling upgrade is not required.	1.10.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.28.5	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	1.10.1
1.28.4	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.	1.10.1
1.27.4	v1.19 v1.21 v1.23 v1.25 v1.27	None	1.10.1
1.25.14	v1.19 v1.21 v1.23 v1.25	• Supported association between add-on specifications and cluster specifications. • Synchronized time zones used by the add-on and the node.	1.10.1
1.25.11	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Upgrades to its community version 1.10.1.	1.10.1
1.25.1	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	1.8.4
1.23.3	v1.15 v1.17 v1.19 v1.21 v1.23	Regular upgrade of add-on dependencies	1.8.4

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.23.2	v1.15 v1.17 v1.19 v1.21 v1.23	Regular upgrade of add-on dependencies	1.8.4
1.23.1	v1.15 v1.17 v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.	1.8.4
1.17.15	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	1.8.4
1.17.9	v1.15 v1.17 v1.19	Regular upgrade of add-on dependencies	1.8.4
1.17.7	v1.15 v1.17 v1.19	Updated the add-on to its community version v1.8.4.	1.8.4
1.17.4	v1.17 v1.19	CCE clusters 1.19 are supported.	1.6.5
1.17.3	v1.17	Supported clusters 1.17 and fixed stub domain configuration issues.	1.6.5
1.17.1	v1.17	Clusters 1.17 are supported.	1.6.5

14.5.2 NGINX Ingress Controller

Introduction

Kubernetes uses kube-proxy to expose Services and provide load balancing. The implementation is at the transport layer. When it comes to Internet applications, where a bucket-load of information is generated, forwarding needs to be more fine-grained, precisely and flexibly controlled by policies and load balancers to deliver higher performance.

This is where ingresses enter. Ingresses provide application-layer forwarding functions, such as virtual hosts, load balancing, SSL proxy, and HTTP routing, for Services that can be directly accessed outside a cluster.

Kubernetes has officially released the Nginx-based Ingress controller. CCE Nginx Ingress controller directly uses community templates and images. The NGINX Ingress Controller generates Nginx configuration and stores the configuration using ConfigMap. The configuration will be written to Nginx pods through the Kubernetes API. In this way, the Nginx configuration is modified and updated. For details, see **How the Add-on Works**.

Open source community: **https://github.com/kubernetes/ingress-nginx**

NOTE

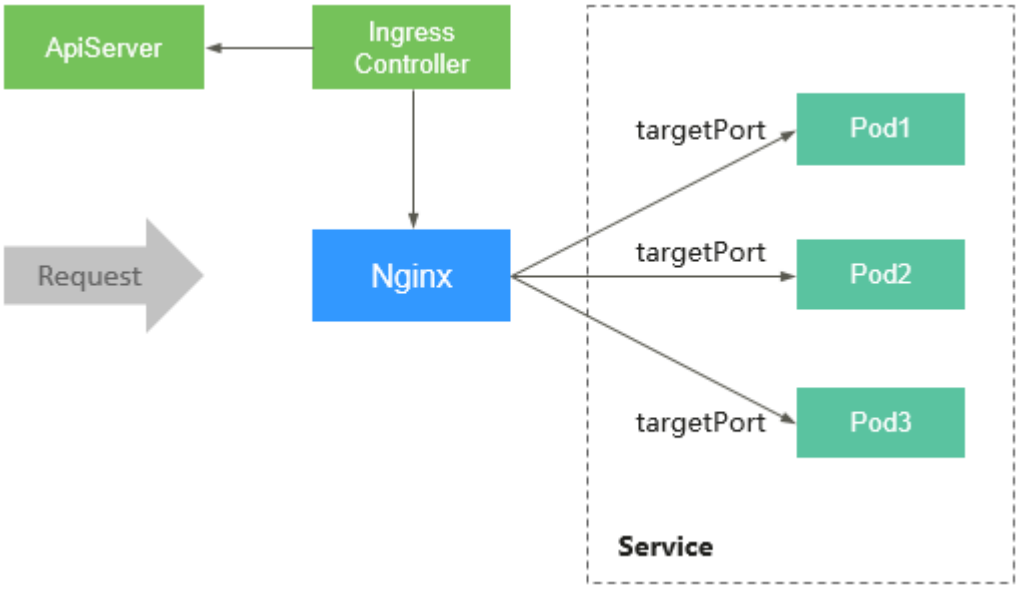
- Starting from version 2.3.3, NGINX Ingress Controller only supports TLS v1.2 and v1.3 by default. If the TLS version is earlier than v1.2, an error will occur during the negotiation process between the client and Nginx Ingress. If more versions of TLS are needed, see **TLS/HTTPS**.
- When installing the NGINX Ingress Controller, you can specify Nginx parameters. These parameters take effect globally and are contained in the **nginx.conf** file. You can search for the parameters in **ConfigMaps**. If the parameters are not included in **ConfigMaps**, the configurations will not take effect.
- After the add-on is installed, you can interconnect with Nginx and set Nginx ingress functions using **Annotations** when **creating an ingress** on the CCE console. For details about the supported annotation fields, see **Annotations**.
- Do not manually modify or delete the load balancer and listener that are automatically created by CCE. Otherwise, the workload will be abnormal. If you have modified or deleted them by mistake, uninstall the nginx-ingress add-on and re-install it.

How the Add-on Works

NGINX Ingress Controller consists of the ingress object, ingress controller, and Nginx. The ingress controller assembles ingresses into the Nginx configuration file (nginx.conf) and reloads Nginx to make the changed configurations take effect. When it detects that the pod in a Service changes, it dynamically changes the upstream server group configuration of Nginx. In this case, the Nginx process does not need to be reloaded. **Figure 14-14** shows how this add-on works.

- An ingress is a group of access rules that forward requests to specified Services based on domain names or URLs. Ingresses are stored in the object storage service etcd and are added, deleted, modified, and queried through APIs.
- The ingress controller monitors the changes of resource objects such as ingresses, Services, endpoints, secrets (mainly TLS certificates and keys), nodes, and ConfigMaps in real time and automatically performs operations on Nginx.
- Nginx implements load balancing and access control at the application layer.

Figure 14-14 Working principles of NGINX Ingress Controller



Precautions

- For clusters earlier than v1.23, **kubernetes.io/ingress.class: "nginx"** must be added to the annotation of the ingress created through the API. If a cluster has multiple NGINX Ingress Controllers installed, you need to replace *nginx* with a custom **Ingress Class** to help identify the controller instance associated with the ingress.
- A dedicated load balancer must be of the network type (TCP/UDP) and support private networks (with a private IP address).
- If the node where NGINX Ingress Controller runs and containers on this node cannot access Nginx ingress, you need to configure anti-affinity for the workload pods and Nginx Ingress Controller. For details, see **Configuring Anti-Affinity Between a Workload and Nginx Ingress Controller**.
- During the NGINX Ingress Controller pod upgrade, 10s are reserved for deleting the NGINX Ingress Controller at the ELB backend.
- The timeout duration for the graceful exit of the NGINX Ingress Controller is 300s. If the timeout duration is longer than 300s during the upgrade of the NGINX Ingress Controller, persistent connections will be disconnected, and connectivity will be interrupted for a short period of time.

Prerequisites

Before installing this add-on, you have one available cluster and there is a node running properly. If no cluster is available, create one according to **Buying a CCE Standard/Turbo Cluster**.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **NGINX Ingress Controller** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

You can adjust the number of add-on instances and resource quotas as required. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure the add-on parameters.

- **Ingress Class**: Enter a custom controller name, which uniquely identifies an Ingress controller. The name of each controller in the same cluster must be unique and cannot be set to **cce**. (**cce** is the unique identifier of the ELB Ingress Controller.) When creating an Ingress, you can specify the controller name to declare which controller should manage this Ingress.
- **Namespace for add-on installation**: Select a namespace for the ingress controller.
- **Load Balancer**: Select a shared or dedicated load balancer. If no load balancer is available, create one. The load balancer has at least two listeners, and ports 80 and 443 are not occupied by listeners.

  If you select a shared load balancer, you can enable the function of obtaining the client IP address. With this function enabled, you can use Nginx ingresses to access application servers, and the servers can obtain the source IP address of clients. When a dedicated load balancer is used, the function of obtaining the client IP address is enabled by default.

  NOTE

  This function is available only in clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

  If backend services have been configured when you enable or disable the function of obtaining a client IP address, traffic will be interrupted. Exercise caution when performing this operation.
- **Admission Check**: Admission control is performed on Ingresses to ensure that the controller can generate valid configurations. Admission verification is performed on the configuration of Nginx Ingresses. If the verification fails, the request will be intercepted. For details about admission verification, see **Access Control**.

  NOTE

  - Admission check slows down the responses to Ingress requests.
  - Only add-ons of version 2.4.1 or later support admission verification.
- **Nginx Parameters**: You can configure the **nginx.conf** file, which will affect all managed ingresses. You can select **GUI** or **YAML**. **GUI** is supported by the NGINX Ingress Controller of version 2.2.75, 2.6.26, 3.0.1, or later.

  To configure custom parameters supported by the Kubernetes community, choose **YAML** and find the related parameters in **ConfigMaps**. For example, you can use the **keep-alive-requests** parameter to describe how to set the maximum number of requests for keeping active connections to 100.

  ```
  {
    "keep-alive-requests": "100"
  }
  ```


NOTE

- If the configured parameters are not listed in **ConfigMaps**, the configurations will not take effect.
- The parameter value must be a string. Otherwise, the installation fails.

The table below shows parameters can be configured on the GUI of the NGINX Ingress Controller add-on of version 2.2.75, 2.6.26, 3.0.1, and later.

Nginx Parameter	Description	Default Value
Maximum Worker Connections	Specifies the maximum number of connections that can be concurrently processed by each NGINX worker process. This parameter is used to control the load of worker processes. In a high-concurrency environment, you are advised to set this parameter to a large value to ensure system stability. Such connections include client connections and connections to backend servers.	65536
Maximum Keepalive Requests	Controls how many requests can be processed by a keepalive connection. If requests exhaust the limit, the connection is closed.	100
Maximum Keepalive Connections to the Upstream Server	Activates the cache for connections to upstream servers. This parameter sets how many idle keepalive connections can be stored in the cache of each worker process. If the idle connections stored in a process exhaust the limit, the connections that are not used for the longest time will be closed.	320
Maximum Keepalive Timeout of the Upstream Server	Specifies the timeout for a keepalive connection between an upstream server and a backend server, in seconds. During this period, NGINX Ingress Controller can maintain connections for reuse. This reduces the overhead required for establishing new connections and significantly improves performance in high QPS scenarios.	900

Nginx Parameter	Description	Default Value
Request Timeout	Specifies the timeout for connecting the client to the proxy server, in seconds. If the client cannot access the backend server within 10 seconds, NGINX Ingress Controller will return a 502 Bad Gateway error. It applies to scenarios where the connection speed is high.	10
Maximum Request Body Size	Specifies the maximum size of the request body that Nginx can send to the backend server through the proxy. This applies to file uploads and large data form submissions. If the size of any request body exceeds the limit, a 413 Payload Too Large error will be returned.	20m
Allow the Backend to Return Server Headers	Typically, NGINX Ingress Controller eliminates the server header information sent by a backend server to a client, which identifies the server. However, if this parameter is set to **true**, the NGINX Ingress Controller will transmit the server header information directly from the backend server to the client. To prevent revealing the server type and version, it is recommended that you disable this feature.	Disable
Allow Underscores in Headers	Some HTTP headers may contain underscores (_), such as **X_Custom-Header**, but this is not recommended according to Request For Comments (RFC) standards. So underscores are not allowed by many servers by default. You can activate this parameter if you require underscores in certain headers, such as when third-party services or clients use underscores in their header information.	Disable

Nginx Parameter	Description	Default Value
Generate a Request ID	After a request is received, NGINX Ingress Controller generates a unique request ID. This ID is usually recorded in logs or transferred to a backend server through header information. This is useful for tracing and debugging requests, especially for locating problems in distributed systems.	Enable
Ignore Invalid Headers	By default, NGINX Ingress Controller rejects HTTP requests that contain an invalid header. With this setting enabled, NGINX Ingress Controller will ignore invalid headers and keep on processing requests. It is useful for clients that do not fully comply with the HTTP standard.	Enable
Reuse Ports	Enabling **SO_REUSEPORT** allows multiple processes or threads to be bound to the same *{IP}:{Port}*. This can effectively improve the concurrent performance of servers, especially for those with multi-core CPUs. With this function enabled, a port can accept more new connections.	Enable
Allow Server Information in Request Body	Disables the server information added to a response header by NGINX Ingress Controller by default. The information usually contains the NGINX version. Disabling this option helps hide server information, enhancing security and preventing attackers from using the version information to attack the system.	Disable
Automatically Redirect HTTP to HTTPS	Disables automatic redirection from HTTP to HTTPS. For example, if you want to use HTTPS only for specific pages, such as the login page, and HTTP for other pages, you can disable the default redirection using this option.	Disable

Nginx Parameter	Description	Default Value
CPU Affinity of Worker Threads	Automatically allocates worker processes to specific CPU cores to improve the performance of multi-core systems. For example, on a multi-core server, some worker processes can be bound on a specific CPU core. This reduces context switching and improves processing efficiency.	Auto

- **Enabling Indicator Collection**: If the add-on version is 2.4.12 or later, Prometheus monitoring metrics can be collected. For details, see **Monitoring Metrics of NGINX Ingress Controller**.
- **Default certificate of the server**: Select an IngressTLS or kubernetes.io/tls key to configure the default certificate when the NGINX Ingress Controller is started. If no secret is available, click **Create TLS Secret**. For details, see **Creating a Secret**. For details about the default server certificate, see **Default SSL Certificate**.
- **404 Service**: By default, the 404 service provided by the add-on is used. To customize the 404 service, enter the namespace/service name. If the service does not exist, the add-on installation will fail.
- **Adding a TCP/UDP Service**: By default, NGINX Ingress Controller can forward only external HTTP and HTTPS traffic. You can add TCP/UDP port mapping to forward external TCP/UDP traffic to services in the cluster. For more information about adding TCP/UDP services, see **Exposing TCP and UDP services**.
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service Port**: specifies the port used by the ELB listener. The port number ranges from 1 to 65535.
  - **Target Service Namespace**: Select the namespace where the Service is in.
  - **Destination Service**: Select an existing Service. Any Services that do not match the search criteria will be filtered out automatically.
  - **Destination Service Port**: Select the access port of the destination Service.

  NOTE

  - If the cluster version is v1.19.16-r5, v1.21.8-r0, v1.23.6-r0, or later, the TCP/UDP hybrid protocols can be configured.
  - If the cluster version is v1.19.16-r5, v1.21.8-r0, v1.23.6-r0, v1.25.2-r0, or later, you can configure the TCP/UDP hybrid protocols to use the same external port.
- **(Optional) Extended Parameter Settings**: additional extended parameters of the add-on If the extended parameter settings conflict with the default settings, the extended parameter settings will be prioritized.
  - **extraArgs**: additional configurable startup parameters of the nginx-ingress-controller component. For details about the startup parameters supported by the community, see the **documentation**.

- **extraInitContainers**: initial container configuration of nginx-ingress-controller. This parameter is supported by add-on 2.2.82, 2.6.32, 3.0.8 and later, with optimized kernel settings by default.
- **maxmindLicenseKey**: Maxmind license key, which can be used to download GeoLite2 databases. This parameter is supported by add-on 2.2.82, 2.6.32, 3.0.8, and later. It is mandatory for NGINX Ingress Controller to configure the **use-geoip2** capability.
- **service**: allows services for nginx-ingress-controller. For details, see **parameter examples in GitHub**. It is supported in add-on versions 2.2.104, 2.6.53, 3.0.31, and later. You can use this parameter to configure ELB certificates for NGINX Ingress Controller. For details, see **Configuring an ELB Certificate for NGINX Ingress Controller**.
- **extraContainers**: other containers added to the nginx-ingress-controller pod. For details, see **parameter examples in GitHub**. It is supported in add-on versions 2.2.104, 2.6.53, 3.0.31, and later.
- **extraVolumeMounts**: allows for mounting additional volumes to the nginx-ingress-controller pod. For details, see **parameter examples in GitHub**. It is supported in add-on versions 2.2.104, 2.6.53, 3.0.31, and later.
- **extraVolumes**: additional volumes mounted to the nginx-ingress-controller pod. For details, see **parameter examples in GitHub**. It is supported in add-on versions 2.2.104, 2.6.53, 3.0.31, and later.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-78 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. **NOTE** The equivalent mode supports only the pods in the **kube-system** and **monitoring** namespaces. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.

Parameter	Description
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Installing Multiple NGINX Ingress Controllers

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate the installed NGINX Ingress Controller, and click **New**.

Step 2 On the page displayed, reconfigure the add-on parameters. For details, see **Installing the Add-on**.

Step 3 Click **Install**.

Step 4 Wait until the installation instruction is delivered. Go back to Add-ons, click **Manage**, and view the installed add-on instance on the add-on details page.

----End

Components

Table 14-79 Add-on components

Component	Description	Resource Type
cceaddon-nginx-ingress-<Controller name>-controller (The controller name in versions earlier than 2.5.4 is **cceaddon-nginx-ingress-controller**.)	Nginx-based ingress controller that provides flexible routing and forwarding for clusters.	Deployment
cceaddon-nginx-ingress-<Controller name>-backend (The controller name in versions earlier than 2.5.4 is **cceaddon-nginx-ingress-default-backend**.)	Default backend of the Nginx ingress. The message "default backend - 404" is returned.	Deployment

Configuring Anti-Affinity Between a Workload and Nginx Ingress Controller

To avoid a situation where the node running NGINX Ingress Controller and its containers cannot access the Nginx Ingress Controller, you should set up anti-affinity between the workload and Nginx Ingress Controller. This means that the workload pods cannot be scheduled to the same node as the Nginx Ingress Controller.

```
apiVersion: apps/v1
kind: Deployment
```



```
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:aplpine
        imagePullPolicy: IfNotPresent
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:    # Implement anti-affinity through the label of the NGINX Ingress Controller pods.
                  - key: app
                    operator: In
                    values:
                      - nginx-ingress   #If multiple NGINX Ingress Controllers are installed in the cluster, the label value is nginx-ingress-<Controller name>.
                  - key: component
                    operator: In
                    values:
                      - controller
              namespaces:
                - kube-system
              topologyKey: kubernetes.io/hostname
```

Change History

Table 14-80 Release history for NGINX Ingress Controller 3.0.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.0.31	v1.27 v1.28 v1.29 v1.30 v1.31	• CCE clusters 1.31 are supported. • Extended parameters can be configured.	1.11.2
3.0.8	v1.27 v1.28 v1.29 v1.30	• Updated the add-on to its community version v1.11.2. • Fixed the CVE-2024-7646 vulnerability.	1.11.2

Table 14-81 Release history for NGINX Ingress Controller 2.6.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.6.53	v1.25 v1.27 v1.28 v1.29	Extended parameters can be configured.	1.9.6
2.6.32	v1.25 v1.27 v1.28 v1.29	Fixed some issues.	1.9.6
2.6.5	v1.25 v1.27 v1.28 v1.29	Metric collection can be disabled in the startup command.	1.9.6
2.6.4	v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	1.9.6

Table 14-82 Release history for NGINX Ingress Controller 2.5.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.5.6	v1.25 v1.27 v1.28	Fixed some issues.	1.9.3
2.5.4	v1.25 v1.27 v1.28	• Multiple Nginx Ingress Controllers can be installed in the same cluster. • The default nginx-ingress certificate can be configured on the console. • Nginx Ingress Controller related metrics can be reported to Prometheus.	1.9.3

Table 14-83 Release history for NGINX Ingress Controller 2.4.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.4.6	v1.25 v1.27 v1.28	• CCE clusters 1.28 are supported. • Supported admission verification. • Supported graceful shutdown and hitless upgrade. • Supported equivalent distribution of add-on instances in multi-AZ deployment mode. • Fixed the CVE-2023-44487 vulnerability.	1.9.3

Table 14-84 Release history for NGINX Ingress Controller 2.3.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.3.5	v1.27	None	1.8.0

Table 14-85 Release history for NGINX Ingress Controller 2.2.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.2.82	v1.23 v1.25	Fixed some issues.	1.5.1
2.2.53	v1.23 v1.25	Fixed some issues.	1.5.1
2.2.52	v1.23 v1.25	• Multiple Nginx Ingress Controllers can be installed in the same cluster. • The default nginx-ingress certificate can be configured on the console.	1.5.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.2.42	v1.23 v1.25	• Supported graceful shutdown and hitless upgrade. • Supported equivalent distribution of add-on instances in multi-AZ deployment mode.	1.5.1
2.2.7	v1.25	• Synchronized time zones used by the add-on and the node. • Supports IPv4 and IPv6 dual stack.	1.5.1
2.2.3	v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Added the tolerance time during which the pods with temporary storage volumes cannot be scheduled. • The default taint tolerance duration is changed to 60s.	1.5.1
2.2.1	v1.25	• CCE clusters 1.25 are supported. • Updated the add-on to its community version v1.5.1.	1.5.1

Table 14-86 Release history for NGINX Ingress Controller 2.1.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.54	v1.19 v1.21 v1.23	Fixed some issues.	1.2.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.33	v1.19 v1.21 v1.23	• Supported graceful shutdown and hitless upgrade. • Supported equivalent distribution of add-on instances in multi-AZ deployment mode.	1.2.1
2.1.9	v1.19 v1.21 v1.23	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • The default taint tolerance duration is changed to 60s. • Synchronized time zones used by the add-on and the node. • Supports IPv4 and IPv6 dual stack.	1.2.1
2.1.5	v1.19 v1.21 v1.23	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • The default taint tolerance duration is changed to 60s.	1.2.1
2.1.3	v1.19 v1.21 v1.23	Enables publishService for nginx-ingress.	1.2.1
2.1.1	v1.19 v1.21 v1.23	Updated the add-on to its community version v1.2.1.	1.2.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.0	v1.19 v1.21 v1.23	• Updated the add-on to its community version v1.2.0. • Fixed the CVE-2021-25746 vulnerability and added rules to disable some annotations values that may cause unauthorized operations. • Fixed the CVE-2021-25745 vulnerability and added rules to disable some access paths that may cause unauthorized operations.	1.2.0

Table 14-87 Release history for NGINX Ingress Controller 2.0.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.0.1	v1.19 v1.21 v1.23	• CCE clusters 1.23 are supported. • Updated the add-on to its community version v1.1.1.	1.1.1

Table 14-88 Release history for NGINX Ingress Controller 1.3.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.3.2	v1.15 v1.17 v1.19 v1.21	• CCE clusters 1.21 are supported. • Updated the add-on to its community version v0.49.3.	0.49.3

Table 14-89 Release history for NGINX Ingress Controller 1.2.x

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.2.6	v1.15 v1.17 v1.19	Added the default seccomp profile.	**0.46.0**
1.2.5	v1.15 v1.17 v1.19	Updated the add-on to its community version v0.46.0.	**0.46.0**
1.2.3	v1.15 v1.17 v1.19	CCE clusters 1.19 are supported.	**0.43.0**
1.2.2	v1.15 v1.17	Updated the add-on to its community version v0.43.0.	**0.43.0**

14.5.3 NodeLocal DNSCache

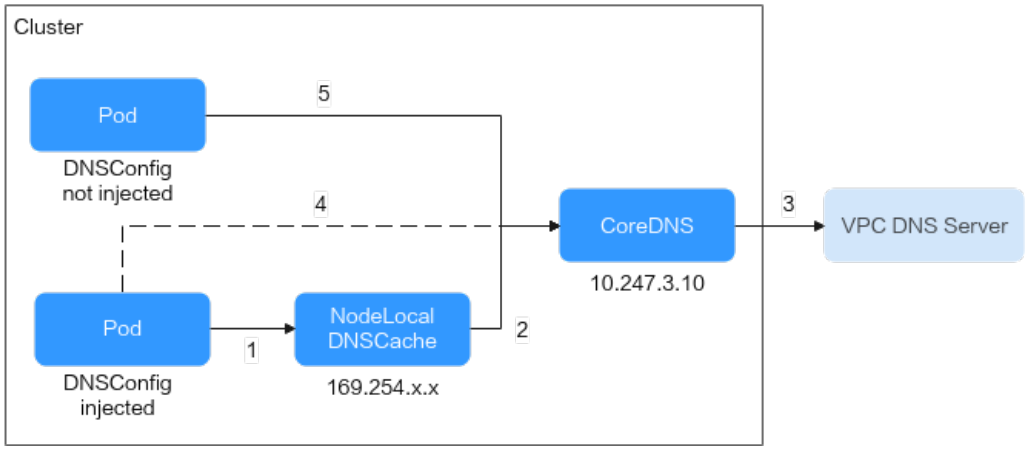
Introduction

The NodeLocal DNSCache (formerly node-local-dns) add-on is developed based on the community **NodeLocal DNSCache**. This add-on functions as a DaemonSet to run the DNS cache proxy on each cluster node to improve cluster DNS performance.

Open source community: **https://github.com/kubernetes/dns**

After NodeLocal DNSCache is enabled, a DNS query goes through the path as shown below.

Figure 14-15 NodeLocal DNSCache query path



The resolution lines are described as follows:

- 1. By default, the pods with DNSConfig injected use NodeLocal DNSCache to resolve requested domain names.
- 2. If NodeLocal DNSCache cannot resolve domain names, it will ask CoreDNS for resolution.
- 3. CoreDNS resolves domain names outside of the cluster by using the DNS server in the VPC.
- 4. If a pod with DNSConfig injected cannot access NodeLocal DNSCache, CoreDNS will resolve the domain name.
- 5. By default, CoreDNS resolves domain names for the pods without DNSConfig injected.

Notes and Constraints

- This feature is available only to clusters of v1.19 or later.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **NodeLocal DNSCache** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Standalone** or **HA** based on the cluster scale. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  Standalone means that a DNSConfig admission controller is deployed on a single pod, which helps to reduce resource usage. **HA** means that DNSConfig admission controllers are deployed on two pods, which enhances the availability of the add-on.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure the add-on parameters.

- **DNSConfig**: After this function is enabled, a DNSConfig admission controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-local-dns-injection=enabled** based on admission webhooks and automatically configures DNSConfig for pods. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.

  After automatic injection is enabled, you can customize the following configuration items for DNSConfig (supported when the add-on version is 1.6.7 or later):

  NOTE

  If DNSConfig has been configured in the pod when automatic injection is enabled, DNSConfig in the pod will be used first.

  – (Optional) **IP Address of DNS Server**: IP address list of the DNS server obtained when the container resolves the domain name. NodeLocal

DNSCache and CoreDNS IP addresses are added by default. You have the option to add an additional IP address, but duplicates will be removed.

- (Optional) **Search Domain**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. You can add up to three extra search domains, but any duplicates will be removed.
- (Optional) **ndots**: specifies that if a domain name has fewer periods (.) than the specified value of **ndots**, it will be combined with the search domain list for DNS query. If the domain name still cannot be resolved, it will be used for DNS query. The system will perform a DNS query on a domain name if the number of periods (.) in it is greater than or equal to the value of **ndots**. If the domain name cannot be resolved correctly, the system will sequentially combine it with the search domain list and then perform a DNS query.

- **Target Namespace**: This parameter is available after **DNSConfig** is enabled. Only NodeLocal DNSCache of v1.3.0 or later supports this function.
  - **Enable All**: CCE adds the **node-local-dns-injection=enabled** label to all created namespaces excluding built-in ones (such as **kube-system**), identifies namespace creation requests, and automatically adds the label to newly created namespaces.
  - **Manual Configure**: You must manually add the **node-local-dns-injection=enabled** label to the namespaces requiring the injection of DNSConfig. For details, see **Managing Namespace Labels**.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-90 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-91 Add-on components

Component	Description	Resource Type
node-local-dns-admission-controller	Automatic DNSConfig injecting	Deployment
node-local-dns-cache	DNS cache proxy on nodes to improve the DNS performance of the cluster	DaemonSet

Using NodeLocal DNSCache

By default, application requests are sent through the CoreDNS proxy. To use NodeLocal DNSCache as the DNS cache proxy, use any of the following methods: (For details, see **Using NodeLocal DNSCache**.)

- Auto injection: Automatically configure the **dnsConfig** field of the pod when creating the pod. (This function is not available for pods in system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

Uninstalling the Add-on

Uninstalling the add-on will affect the pods that have used the node-local-dns address for domain name resolution. Before uninstalling the add-on, delete the **node-local-dns-injection=enabled** label from the involved namespaces, and delete and recreate the pods with the **node-local-dns-webhook.k8s.io/status: injected** label.

Step 1 Check the add-on.

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **NodeLocal DNSCache** on the right, and click **Edit**.
2. In the **Parameters** area, check whether **DNSConfig** is enabled.

   If **DNSConfig** has been enabled:

   a. In the navigation pane, choose **Namespaces**.

   b. Locate the rows that contain the namespaces with the **node-local-dns-injection=enabled** label and delete the label. For details, see **Managing Namespace Labels**.

   c. Delete the pods in these namespaces and recreate pods.

   If **DNSConfig** has not been enabled:

a. Use kubectl to access the cluster.

b. Check the pods with DNSConfig manually injected. If multiple namespaces are involved, check all the pods in these namespaces.

   For example, to check pods in the **default** namespace, run the following command:

   ```
   kubectl get pod -n default -o yaml
   ```

c. Manually remove DNSConfig and recreate pods.

Step 2 Uninstall NodeLocal DNSCache.

1. In the navigation pane, choose **Add-ons**. Locate **NodeLocal DNSCache** and click **Uninstall**.
2. In the displayed dialog box, click **Yes**.

----End

Helpful Links

Using NodeLocal DNSCache to Improve DNS Performance

Change History

Table 14-92 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.6.62	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	1.22.20
1.6.37	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	The admission-controller logs have been updated to standard output logs.	1.22.20
1.6.36	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	1.22.20

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.6.8	v1.23 v1.25 v1.27 v1.28 v1.29	Optimized custom injection configuration experience.	1.22.20
1.6.7	v1.23 v1.25 v1.27 v1.28 v1.29	Added custom injection configurations.	1.22.20
1.6.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	1.22.20
1.5.2	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	1.22.20
1.5.1	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.	1.22.20
1.5.0	v1.21 v1.23 v1.25 v1.27 v1.28	• CCE clusters 1.28 are supported. • Supported equivalent distribution of add-on instances in multi-AZ deployment mode. • Changed the base image OS of the add-on pods to Huawei Cloud EulerOS 2.0.	1.22.20

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.4.0	v1.19 v1.21 v1.23 v1.25 v1.27	Resolved the issue that CCI pods took an excessively long time to access the Internet.	1.22.20
1.2.7	v1.19 v1.21 v1.23 v1.25	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.	1.21.1
1.2.4	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.	1.21.1
1.2.2	v1.19 v1.21 v1.23	Supports customized NodeLocal DNSCache specifications.	1.21.1

14.6 Container Storage Add-ons

14.6.1 CCE Container Storage (Everest)

Introduction

Everest is a cloud native container storage system, which enables clusters of Kubernetes v1.15.6 or later to access cloud storage services through the CSI.

Everest is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.15 or later is created.

Notes and Constraints

- In version 1.2.0 of the Everest add-on, **key authentication** is optimized when OBS is used. After the Everest add-on is upgraded from a version earlier than 1.2.0, restart all workloads that use OBS in the cluster. Otherwise, workloads may not be able to use OBS.
- Nodes running Huawei Cloud EulerOS 1.1 support the Everest add-ons of v2.x.x (v2.1.9 or later) and v1.2.x (v1.2.70 or later), but do not support the add-ons of v1.3.x.

Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Container Storage (Everest)** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small**, **Medium**, or **Large** as needed. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  The small specification is best for clusters with up to 50 nodes and 500 PVCs. The medium specification works well for clusters with up to 200 nodes and 2000 PVCs. The large specification is perfect for clusters with up to 1000 nodes and 10,000 PVCs.

- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. The requested CPUs and memory can be adjusted based on the number of nodes and PVCs. For details, see **Table 14-93**.

  In non-typical scenarios, the formulas for estimating the limits are as follows:

  – everest-csi-controller

    ▪ CPU limit: 250m for 200 or fewer nodes, 350m for 1000 nodes, and 500m for 2000 nodes

    ▪ Memory limit = (200 MiB + Number of nodes x 1 MiB + Number of PVCs x 0.2 MiB) x 1.2

  – everest-csi-driver

    ▪ CPU limit: 300m for 200 or fewer nodes, 500m for 1000 nodes, and 800m for 2000 nodes

    ▪ Memory limit: 300 MiB for 200 or fewer nodes, 600 MiB for 1000 nodes, and 900 MiB for 2000 nodes

Table 14-93 Recommended configuration limits in typical scenarios

Configuration Scenario			everest-csi-controller		everest-csi-driver	
Nodes	PVs/ PVCs	Add-on Pods	CPU Cores (Limit = Request)	Memory (Limit = Request)	CPU Cores (Limit = Request)	Memory (Limit = Request)
50	1000	2	250m	600 MiB	300m	300 MiB
200	1000	2	250m	1 GiB	300m	300 MiB
1000	1000	2	350m	2 GiB	500m	600 MiB
1000	5000	2	450m	3 GiB	500m	600 MiB

Configuration Scenario			everest-csi-controller		everest-csi-driver	
2000	5000	2	550m	4 GiB	800m	900 MiB
2000	10000	2	650m	5 GiB	800m	900 MiB

Step 3 Configure the add-on parameters.

Table 14-94 Add-on parameters

Item	Configuration Method	Description
Reserved EVS Disks for Non-Container Workloads	Visualized GUI configuration	Number of disks on the node reserved for custom use. This parameter is supported when the add-on version is 2.3.11 or later. Assume that a maximum of 20 EVS disks can be attached to a node, and the value of this parameter is set to **6**. Then 14 (20-6) disks can be attached to this node when the system schedules the EVS disk attachment workloads. The reserved six disks include one system disk and one data disk that have been attached to the node. You can attach four EVS disks to this node as additional data disks or raw disks for a local storage pool.
Prohibit Global Secret from Mounting Object Storage	Visualized GUI configuration	Whether the default AK/SK can be used when an object bucket or parallel file system is mounted. • **is**: If you do not specify a custom secret for mounting OBS storage, the global secret you uploaded will be used. • **No**: If you do not specify a custom secret for mounting OBS storage, only the global secret you uploaded can be used. Otherwise, the mounting will fail.
Concurrent EVS Disk Attaching Tasks	Visualized GUI configuration	Number of workers that can be concurrently processed by Everest for attaching EVS volumes. The default value is **60**.

Item	Configuration Method	Description
Concurrent EVS Disk Detaching Tasks	Visualized GUI configuration	Number of workers that can be concurrently processed by Everest for detaching EVS volumes. The default value is **60**.
Distributed Volume Mounting	Visualized GUI configuration	When enabled, the everest-csi-driver component on each node is responsible for attaching or detaching EVS disks. If disabled, the everest-csi-controller component takes on this responsibility.
flow_control	Extended Parameter Settings	The default configuration is used. After installing the CCE Container Storage (Everest) add-on, you can refer to ConfigMap everest-driver-th-config in the kube-system namespace for more details.
over_subscription	Extended parameter settings	Overcommitment ratio of the local storage pool (**local_storage**). The default value for the local storage pool size is set to **80**. If the pool size is 100 GiB, it can be overcommitted up to 180 GiB. (Total capacity after overcommitment = (1 + Overcommitment ratio) x Actual capacity)
enable_local_autoexpander	Extended parameter settings	Whether to enable automatic scale-out for the local storage pool of thin volumes. If this function is enabled, automatic scale-out is triggered based on the scale-out threshold and scale-out step of the local storage pool.
expansion_threshold	Extended parameter settings	Capacity expansion threshold of the local storage pool of the thin volume. When the usage of the local storage pool of the thin volume exceeds the threshold, the local storage pool is automatically expanded.

Item	Configuration Method	Description
expansion_step	Extended parameter settings	Capacity expansion step of a single EVS disk in the local storage pool of thin provisioning volumes (unit: Gi)
expansion_max_evs_size	Extended parameter settings	Maximum capacity of a single EVS disk in the local storage pool of thin provisioning volumes (unit: Gi)
volume_attaching_flow_ctrl	Extended parameter settings	Maximum number of EVS volumes that can be attached by the Everest add-on within 1 minute. The default value is **0**, indicating that the performance of attaching EVS volumes is determined by the underlying storage resources.

NOTE

In the extended parameter settings, you can customize the advanced configurations that are not displayed on the GUI. If the settings in the extended parameters conflict with those on the GUI, the settings in the extended parameters will work.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-95 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-96 Add-on components

Component	Description	Resource Type
everest-csi-controller	Used to create, delete, snapshot, expand, attach, and detach storage volumes. If the cluster version is 1.19 or later and the add-on version is 1.2.*x*, the pod of the everest-csi-controller component also has an everest-localvolume-manager container by default. This container manages the creation of LVM storage pools and local PVs on the node.	Deployment
everest-csi-driver	Used to mount and unmount PVs and resize file systems. If the add-on version is 1.2.*x* and the region where the cluster is located supports node-attacher, the pod of the everest-csi-driver component also contains an everest-node-attacher container. This container is responsible for distributed EVS attaching. This configuration item is available in some regions.	DaemonSet

Collecting Prometheus Metrics

everest-csi-controller exposes Prometheus metrics over port 3225. You can create an on-premises Prometheus collector to identify and obtain everest-csi-controller metrics from **http://{***{everest-csi-controller pod IP address}***}:3225/metrics**.

NOTE

Prometheus metrics can be exposed only when the Everest add-on version is 2.4.4 or later.

Table 14-97 Key metrics

Metric	Type	Description	Label	Example
everest_action_result_total	Counter	Invoking of different functions	action: indicates different functions. For details, see **Table 14-98**. result: indicates that the invoking is successful or fails.	everest_action_result_total{action="create_snapshot:disk.csi.everest.io",result="success"} 2

Metric	Type	Description	Label	Example
everest_function_duration_seconds_bucket	Histogram	Number of times that different functions are executed at different time	function: indicates different functions. For details, see **Table 14-98**.	everest_function_duration_seconds_bucket{function="create_snapshot:disk.csi.everest.io",le="10"} 2
everest_function_duration_seconds_sum	Histogram	Total invoking time of different functions	function: indicates different functions. For details, see **Table 14-98**.	everest_function_duration_seconds_sum{function="create:disk.csi.everest.io"} 24.381399053
everest_function_duration_seconds_count	Histogram	Number of invoking times of different functions	function: indicates different functions. For details, see **Table 14-98**.	everest_function_duration_seconds_count{function="attach:disk.csi.everest.io"} 4

action and **function** specify different CSI drivers and their functions, and are in the format of *{Function}:{CSI driver}*. For example, **create:disk.csi.everest.io** specifies that the function is to create a volume and the volume type is EVS disk.

Table 14-98 Functions

Operation	Description
create	Creates a volume.
delete	Deletes a volume.
attach	Mounts a volume.
detach	Detaches a volume.
expand	Expands the capacity of a volume.
create_snapshot	Creates a volume snapshot.
delete_snapshot	Deletes a volume snapshot

Change History

Table 14-99 Release history

Add-on Version	Supported Cluster Version	New Feature
2.4.105	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.
2.4.75	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	On Huawei Cloud EulerOS 2.0 nodes, you can configure the EVS PVC's fstype to xfs.
2.4.72	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.
2.4.44	v1.23 v1.25 v1.27 v1.28 v1.29	• Subdirectories can be created in an SFS 3.0 file system. • The usage of storage volumes can be obtained.
2.4.28	v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
2.4.21	v1.23 v1.25 v1.27 v1.28 v1.29	• Supported OBS multi-AZ storage redundancy policy. • The prefixes of storage volume names can be customized. • Supported yearly/monthly EVS disks.

Add-on Version	Supported Cluster Version	New Feature
2.4.8	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• CCE clusters 1.29 are supported. • Supported GPSSD2 disks. • Supported DSS. The cluster version must be v1.21.15-r0, v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.
2.3.23	v1.21 v1.23 v1.25 v1.27 v1.28	Subdirectories can be created in an SFS Turbo file system.
2.3.21	v1.21 v1.23 v1.25 v1.27 v1.28	Fixed some issues.
2.3.14	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters 1.28 are supported.
2.1.51	v1.19 v1.21 v1.23 v1.25 v1.27	Supported Huawei Cloud EulerOS 2.0.
2.1.30	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of add-on pods on nodes in different AZs. • Adapts the obsfs package to Ubuntu 22.04.
2.1.13	v1.19 v1.21 v1.23 v1.25	Optimized the performance of creating subpath PVCs in batches for SFS Turbo volumes.

Add-on Version	Supported Cluster Version	New Feature
2.1.9	v1.19 v1.21 v1.23 v1.25	• Supported graceful exit of the controller. • CCE clusters 1.25 are supported.
2.0.9	v1.19 v1.21 v1.23	• Rebuilt certain code and architecture of everest to improve its scalability and stability. • Enabled graceful exit. • Supported OBS process monitoring.
1.3.28	v1.19 v1.21 v1.23	• Enabled graceful exit. • Supported OBS process monitoring.
1.3.22	v1.19 v1.21 v1.23	Handled occasional read and write failures after repeated disk mounting.
1.3.20	v1.19 v1.21 v1.23	Handled occasional read and write failures after repeated disk mounting.
1.3.17	v1.19 v1.21 v1.23	• Updated the rollingUpdates.maxUnavailable of everest-csi-driver from 10 to 10%. • Supported user-defined pod anti-affinity rules. • Counted the maximum number of SCSI volumes that can be managed by the CSI plug-in on a node. • Drivers can be deployed based on customized resource specifications.
1.3.8	v1.23	CCE clusters 1.23 are supported.
1.3.6	v1.23	CCE clusters 1.23 are supported.
1.2.78	v1.15 v1.17 v1.19 v1.21	Supported anti-affinity scheduling of add-on pods on nodes in different AZs.

Add-on Version	Supported Cluster Version	New Feature
1.2.70	v1.15 v1.17 v1.19 v1.21	Optimized the performance of creating subpath PVCs in batches for SFS Turbo volumes.
1.2.67	v1.15 v1.17 v1.19 v1.21	• Supported graceful exit of the controller. • Supported OBS process monitoring.
1.2.61	v1.15 v1.17 v1.19 v1.21	• Enabled graceful exit. • Supported OBS process monitoring.
1.2.55	v1.15 v1.17 v1.19 v1.21	Handled occasional read and write failures after repeated disk mounting.
1.2.53	v1.15 v1.17 v1.19 v1.21	Handled occasional read and write failures after repeated disk mounting.
1.2.51	v1.15 v1.17 v1.19 v1.21	• Updated the rollingUpdates.maxUnavailable of everest-csi-driver from 10 to 10%. • Supported user-defined pod anti-affinity rules. • Counted the maximum number of SCSI volumes that can be managed by the CSI plug-in on a node.
1.2.44	v1.15 v1.17 v1.19 v1.21	• Enterprise projects can be selected for EVS and OBS volumes. • By default, the **enable_noobj_cache** parameter is no longer used for mounting OBS buckets.
1.2.42	v1.15 v1.17 v1.19 v1.21	• Enterprise projects can be selected for EVS and OBS volumes. • By default, the **enable_noobj_cache** parameter is no longer used for mounting OBS buckets.

Add-on Version	Supported Cluster Version	New Feature
1.2.30	v1.15 v1.17 v1.19 v1.21	Supported emptyDir.
1.2.28	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.
1.2.27	v1.15 v1.17 v1.19 v1.21	Supported extreme SSD (ESSD) and general-purpose SSD (GPSSD) EVS disks.
1.2.13	v1.15 v1.17 v1.19	Supported EulerOS 2.10.
1.2.9	v1.15 v1.17 v1.19	• Enhanced the reliability of PV resource lifecycle maintenance. • Attach/Detach Controller can be used to attach or detach volumes in clusters 1.19.10. • Improved SFS mounting stability. • Changed the default EVS creation type of a new cluster to SAS.
1.2.5	v1.15 v1.17 v1.19	• Improves the reliability of mounting-related capabilities. • Optimized the authentication function of using OBS, which requires you to upload the access key. • Improved the compatibility of the everest add-on with FlexVolume volumes. • Improved running stability of the add-on.
1.1.12	v1.15 v1.17	Enhanced the reliability of the everest-csi-controller component.

Add-on Version	Supported Cluster Version	New Feature
1.1.11	v1.15 v1.17	• Supported security hardening. • Supported third-party OBS storage. • Switched to the EVS query API with better performance. • Used snapshots to create disks in clone mode by default. • Optimized and enhanced disk status detection and log output for attaching and detaching operations. • Improved the reliability of determining authentication expiration.
1.1.8	v1.15 v1.17	Supported CCE 1.17. If CCE 1.13 is upgraded to 1.15, Everest can take over all functions of the FlexVolume driver.
1.1.7	v1.15 v1.17	Supported CCE 1.17. If CCE 1.13 is upgraded to 1.15, Everest can take over all functions of the FlexVolume driver.

14.6.2 FlexVolume (Discarded)

Introduction

CCE Container Storage (FlexVolume), also called storage-driver, functions as a standard Kubernetes FlexVolume plugin to allow containers to use EVS, SFS, OBS, and SFS Turbo storage resources. By installing and upgrading storage-driver, you can quickly install and update cloud storage capabilities.

FlexVolume is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.13 or earlier is created.

Notes and Constraints

- For clusters created in CCE, Kubernetes v1.15.11 is a transitional version in which the FlexVolume add-on is compatible with the CSI add-on (**Everest**). Clusters of v1.17 and later versions do not support FlexVolume anymore. Use the Everest add-on.
- The FlexVolume add-on will be maintained by Kubernetes developers, but new functionality will only be added to **Everest**. Do not create CCE storage that uses the FlexVolume add-on (storage-driver) anymore. Otherwise, storage may malfunction.
- This add-on can be installed only in **clusters of v1.13 or earlier**. By default, the **Everest** add-on is installed when clusters of v1.15 or later are created.

NOTE

In a cluster of v1.13 or earlier, when an upgrade or bug fix is available for storage functionalities, you only need to install or upgrade the storage-driver add-on. Upgrading the cluster or creating a cluster is not required.

Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

If storage-driver is not installed in a cluster, perform the following steps to install it:

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Container Storage (FlexVolume)** on the right, and click **Install**.

Step 2 Click **Install** to install the add-on. Note that the storage-driver has no configurable parameters and can be directly installed.

----End

14.7 Container Security Add-ons

14.7.1 CCE Secrets Manager for DEW

Introduction

The CCE Secrets Manager for DEW add-on (formerly dew-provider) is used to interconnect with **Data Encryption Workshop (DEW)**, which allows you to mount secrets stored outside a cluster (DEW for storing sensitive information) to pods. In this way, sensitive information can be decoupled from the cluster environment, which prevents information leakage caused by program hardcoding or plaintext configuration.

Notes and Constraints

- DEW includes Key Management Service (KMS), Cloud Secret Management Service (CSMS), and Key Pair Service (KPS). Currently, the dew-provider add-on can interconnect only with CSMS.
- A maximum of 500 SecretProviderClass objects can be created.
- When the add-on is uninstalled, related CRD resources are deleted accordingly. Even if the add-on is reinstalled, the original SecretProviderClass object is unavailable. If you want to use the original SecretProviderClass resources after the add-on is uninstalled and then reinstalled, manually create them again.

How the Add-on Works

- Basic mounting: After the dew-provider add-on is installed, you can create a SecretProviderClass object and declare and reference the volume in a pod.

When the pod is started, the secret declared in the SecretProviderClass object is mounted to the pod.

- Scheduled rotation: After a pod runs properly, if the secret declared in the SPC object and stored in CSMS is updated, the latest secret values can be updated to the pod through scheduled rotation. When using this capability, set the secret version to **latest**.
- Real-time awareness of SPC changes: After a pod runs properly, if a user modifies the secret declared in the SPC object (for example, a secret is added or the version number is changed), the add-on can detect the change in real time and update the secret to the pod.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **CCE Secrets Manager for DEW** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure parameters in the **Parameters** area, as listed in the following table.

Item	Parameter	Description
Secret Synchronization Period	rotation_poll_interval	Rotation interval, in unit of m (instead of min). The rotation interval indicates the interval for sending a request to CSMS and obtaining the latest secret. The proper interval range is [1m, 1440m]. The default value is **2m**.

Step 3 Click **Install**.

After the add-on is installed, select the cluster and choose **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 14-100 Add-on components

Component	Description	Resource Type
dew-provider	A component that obtains specified secrets from CSMS and mounts them to the pods	DaemonSet

Component	Description	Resource Type
csi-secrets-store	A component that maintains two CRDs, SecretProviderClass (SPC) and SecretProviderClassPodStatus (spcPodStatus). **SPC** is used to describe the secret that users are interested in (such as the secret version and name). It is created by users and will be referenced in pods. **spcPodStatus** is used to trace the binding relationships between pods and secrets. It is automatically created by csi-driver and requires no manual operation. One pod corresponds to one spcPodStatus. After a pod is started, a spcPodStatus is generated for the pod. When the pod lifecycle ends, the spcPodStatus is deleted accordingly.	DaemonSet

Mounting a Credential Using a Volume

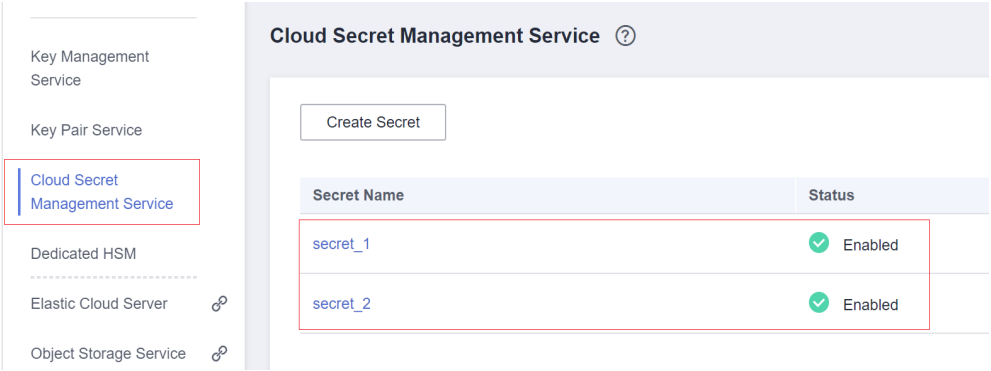
Step 1 Create a ServiceAccount.

1. Create a ServiceAccount object, **which declares the secret names that can be used by services. If a user references a secret that is not declared here, the mounting will fail. As a result, the pod cannot run**.

   Create the **serviceaccount.yaml** file based on the template below, and declare the secret names that can be used by services in the **cce.io/dew-resource** field. Here, **secret_1** and **secret_2** are declared, indicating that the service is allowed to reference two secrets. In subsequent operations, if the user references **secret_3** in the service, the verification fails. As a result, the secret cannot be mounted and the pod cannot run.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: nginx-spc-sa
     annotations:
       cce.io/dew-resource: "[\"secret_1\",\"secret_2\"]"  #secrets that allow pod to use
   ```

   Ensure that the secrets declared here exist in CSCM, as shown in the following figure. Otherwise, even if the verification is successful, an error occurs when the corresponding secret is obtained from CSCM. As a result, the pod cannot run properly.

2. Run the following command to create the ServiceAccount:

   kubectl apply -f serviceaccount.yaml

3. Check whether the ServiceAccount object is successfully created.

   ```
   $ kubectl get sa
   NAME       SECRETS   AGE
   default        1         18d   # This is the default ServiceAccount object of the system.
   nginx-spc-sa    1         19s   # This is the newly created ServiceAccount object.
   ```

   A ServiceAccount object named **nginx-spc-sa** has been created. This object will be referenced in pods.

Step 2 Create a SecretProviderClass.

1. The SecretProviderClass object is used to describe the secret information (such as the version and name) that users are interested in. It is created by users and will be referenced in pods.

   Create the **secretproviderclass.yaml** file using the template below. Pay attention to the **objects** field in **parameters**, which is an array used to declare the secret to be mounted.

   ```
   apiVersion: secrets-store.csi.x-k8s.io/v1
   kind: SecretProviderClass
   metadata:
     name: spc-test
   spec:
     provider: cce     # The value is fixed at cce.
     parameters:
       objects: |
           - objectName: "secret_1"
             objectVersion: "v1"
             objectType: "csms"
   ```

Parameter	Type	Mandatory	Description
objectName	String	Yes	Credential name. Set this parameter to the secret referenced in ServiceAccount. If there are multiple object names defined in the same SecretProviderClass, each value of the **objectName** parameter must be unique. Otherwise, the mounting fails.
objectAlias	String	No	File name of the secret written into the container. If this parameter is not specified, the file name of the secret written into the container is the value of **objectName** by default. If this parameter is specified, the value must be different from **objectName** and from the **objectAlias** and **objectName** values of other secrets. Otherwise, the mounting fails.
objectType	String	Yes	Secret type. Only **csms** is supported. A value other than **csms** is invalid.

Parameter	Type	Mandatory	Description
objectVersion	String	Yes	Secret version – Specify a version, for example, **v1** or **v2**. – Use the latest version, for example, **latest**. When **objectVersion** is set to **latest**, if the corresponding secret in CSCM is updated, the secret will be updated to the pod after a certain interval (**rotation_poll_interval**).

2. Run the following command to create a SecretProviderClass object:

   kubectl apply -f secretproviderclass.yaml

3. Check whether the SecretProviderClass object has been created.

   ```
   $ kubectl get spc
   NAME   AGE
   spc-test   20h
   ```

   A SecretProviderClass object named **spc-test** is created. This object will be referenced in pods.

Step 3 Create a pod.

The following describes how to create an Nginx application.

1. Define a workload, reference the created ServiceAccount object in **serviceAccountName**, and reference the created SPC object in **secretProviderClass**, specify the mount path of the container in **mountPath**. (Do not specify special directories such as **/** and **/var/run**. Otherwise, the container may fail to be started.)

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nginx-spc
     labels:
       app: nginx
   spec:
     replicas: 1
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         serviceAccountName: nginx-spc-sa   # Reference the created ServiceAccount.
         volumes:
           - name: secrets-store-inline
             csi:
               driver: secrets-store.csi.k8s.io
               readOnly: true
               volumeAttributes:
                 secretProviderClass: "spc-test"  # Reference the created SPC.
         containers:
           - name: nginx-spc
             image: nginx:alpine
             imagePullPolicy: IfNotPresent
             volumeMounts:
               - name: secrets-store-inline
   ```



```
          mountPath: "/mnt/secrets-store"  # Define the mount path of secrets in the container.
          readOnly: true
    imagePullSecrets:
      - name: default-secret
```

2. Create a pod.

```
kubectl apply -f deployment.yaml
```

3. Check whether the pod has been created.

```
$ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-spc-67c9d5b594-642np     1/1     Running    0          20s
```

4. Access the container and check whether the specified secret is written properly. For example:

```
$ kubectl exec -ti nginx-spc-67c9d5b594-642np -- /bin/bash
root@nginx-spc-67c9d5b594-642np:/#
root@nginx-spc-67c9d5b594-642np:/# cd /mnt/secrets-store/
root@nginx-spc-67c9d5b594-642np:/mnt/secrets-store#
root@nginx-spc-67c9d5b594-642np:/mnt/secrets-store# ls
secret_1
```

The command output shows that **secret_1** declared in the SPC object has been written to the pod.

In addition, you can obtain **spcPodStatus** to check the binding relationship between pods and secrets. For example:

```
$ kubectl get spcps
NAME                                                AGE
nginx-spc-67c9d5b594-642np-default-spc-test   103s
$ kubectl get spcps nginx-spc-67c9d5b594-642np-default-spc-test -o yaml
......
status:
mounted: true
objects:    # Mounted secret
- id: secret_1
version: v1
podName: nginx-spc-67c9d5b594-642np   # Pod that references the SPC object
secretProviderClassName: spc-test              # SPC object
targetPath: /mnt/paas/kubernetes/kubelet/pods/6dd29596-5b78-44fb-9d4c-a5027c420617/volumes/
kubernetes.io~csi/secrets-store-inline/mount
```

----End

Mounting a Credential Using a Secret

NOTE

CCE Secrets Manager for DEW must be 1.1.1 or later.

Step 1 Create a ServiceAccount.

1. Create a ServiceAccount object, **which declares the secret names that can be used by services. If a user references a secret that is not declared here, the mounting will fail. As a result, the pod cannot run**.

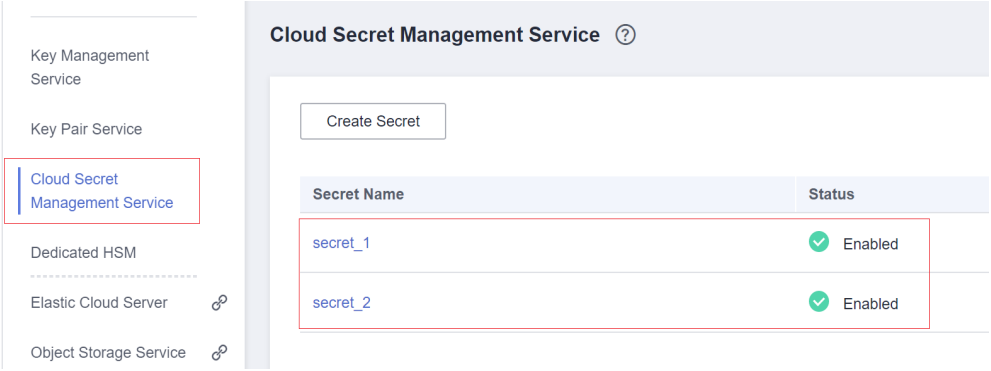
   Create the **serviceaccount.yaml** file based on the template below, and declare the secret names that can be used by services in the **cce.io/dew-resource** field. Here, **secret_1** and **secret_2** are declared, indicating that the service is allowed to reference two secrets. In subsequent operations, if the user references **secret_3** in the service, the verification fails. As a result, the secret cannot be mounted and the pod cannot run.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nginx-spc-sa
```



```
annotations:
  cce.io/dew-resource: "[\"secret_1\",\"secret_2\"]"  #secrets that allow pod to use
```

Ensure that the secrets declared here exist in CSCM, as shown in the following figure. Otherwise, even if the verification is successful, an error occurs when the corresponding secret is obtained from CSCM. As a result, the pod cannot run properly.



2. Run the following command to create the ServiceAccount:

   kubectl apply -f serviceaccount.yaml

3. Check whether the ServiceAccount object is successfully created.

```
$ kubectl get sa
NAME       SECRETS   AGE
default        1         18d   # This is the default ServiceAccount object of the system.
nginx-spc-sa   1         19s   # This is the newly created ServiceAccount object.
```

   A ServiceAccount object named **nginx-spc-sa** has been created. This object will be referenced in pods.

Step 2 Create a SecretProviderClass.

1. Create the **secretproviderclass.yaml** file using the template below.

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: nginx-deployment-spc-k8s-secrets
spec:
  provider: cce
  parameters:
    # Reference a secret in CSMS.
    objects: |
      - objectName: "secret_1"
        objectType: "csms"
        objectVersion: "latest"
        jmesPath:
          - path: username
            objectAlias: dbusername
          - path: password
            objectAlias: dbpassword
  # Create a CCE secret based on the CSMS secret and mount the secret to a pod.
  secretObjects:
    - secretName: my-secret-01
      type: Opaque
      data:
        - objectName: dbusername
          key: db_username_01
        - objectName: dbpassword
          key: db_password_01
```


Table 14-101 objects parameter

Parameter	Type	Mandatory	Description
objectName	String	Yes	Credential name. Set this parameter to the secret referenced in ServiceAccount. If there are multiple object names defined in the same SecretProviderClass, each value of the **objectName** parameter must be unique. Otherwise, the mounting fails.
objectType	String	Yes	Secret type. Only **csms** is supported. A value other than **csms** is invalid.
objectVersion	String	Yes	Secret version – Specify a version, for example, **v1** or **v2**. – Use the latest version, for example, **latest**. When **objectVersion** is set to **latest**, if the corresponding secret in CSCM is updated, the secret will be updated to the pod after a certain interval (**rotation_poll_interval**).
jmesPath	Array of Object	Yes	**jmesPath** is used to extract key-value pairs from objects in JSON format. CCE Secrets Manager for DEW uses this tool to support Secret mounting. – **path**: Enter the key in a DEW secret. **NOTE** The key cannot contain special characters such as +, -, {}, and (). – **objectAlias**: name of the file mounted to a pod. The value must be the same as the value of **objectName** defined in **secretObjects**.

Table 14-102 secretObjects parameter

Parameter	Type	Mandatory	Description
secretName	String	Yes	Secret name
type	String	Yes	Secret type

Parameter	Type	Mandatory	Description
data	Array of Object	Yes	– **objectName**: name of the file mounted to the pod. The value must be the same as the value of **objectAlias** specified in **objects**. – **key**: key in a secret. The value can be used to reference the encrypted content in a pod.

2. Run the following command to create a SecretProviderClass object:

   kubectl apply -f secretproviderclass.yaml

3. Check whether the SecretProviderClass object has been created.

```
$ kubectl get spc
NAME   AGE
nginx-deployment-spc-k8s-secrets   20h
```

Step 3 Create a pod. The following describes how to create an Nginx application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-k8s-secrets
  labels:
    app: nginx-k8s-secrets
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-k8s-secrets
  template:
    metadata:
      labels:
        app: nginx-k8s-secrets
    spec:
      serviceAccountName: nginx-spc-sa   # Reference the created ServiceAccount.
      containers:
      - name: nginx-deployment-k8s-secrets
        image: nginx
        volumeMounts:   # Mount the secret to a container.
          - name: secrets-store-inline   # Name of the volume to be mounted
            mountPath: "/mnt/secrets"    # Path of the container to be mounted
            readOnly: true
        env:    # Reference the secret in environment variables.
          - name: DB_USERNAME_01      # Environment variable name in the workload
            valueFrom:
              secretKeyRef:
                name: my-secret-01    # Secret name specified in the SPC
                key: db_username_01   # The value of the key specified in the SPC
          - name: DB_PASSWORD_01
            valueFrom:
              secretKeyRef:
                name: my-secret-01
                key: db_password_01
      imagePullSecrets:
        - name: default-secret
      volumes:    # Use the secret specified in the SPC to create a volume.
        - name: secrets-store-inline  # Custom volume name
          csi:
            driver: secrets-store.csi.k8s.io
            readOnly: true
            volumeAttributes:
```



```
            secretProviderClass: nginx-deployment-spc-k8s-secrets   # Name of the SPC created in the
previous step
```

Step 4 Verify the result.

```
$ kubectl get secrets
NAME            TYPE                             DATA   AGE
default-secret  kubernetes.io/dockerconfigjson   1      33d
my-secret-01    Opaque                           2      1h
```

The output shows that a secret named **my-secret-01** has been created using **secret_1** specified in the SPC object.

----End

Scheduled Rotation

As described before, you can use this add-on to complete the mount secrets, that is, you can write the secrets stored in CSMS to a pod.

To change the secret version declared in the SPC object to **latest**, run the following command:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: spc-test
spec:
  provider: cce
  parameters:
    objects: |
          - objectName: "secret_1"
            objectVersion: "latest"  # change "v1"to "latest"
            objectType: "csms"
```

After the SPC object is updated, the add-on periodically sends a request to CSMS to obtain the value of secret_1 of the latest version and updates the value to the pod that references the SPC object. The interval for the add-on to periodically send requests is specified by **rotation_poll_interval** set in **Installing the Add-on**.

Real-Time Detection of SPC Changes

SPC changes are already detected in real time in **Mounting a Credential Using a Volume** and **Scheduled Rotation**. For demonstration, add secret **secret_2** to the SPC object as follows:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: spc-test
spec:
  provider: cce
  parameters:
    objects: |
          - objectName: "secret_1"
            objectVersion: "latest"
            objectType: "csms"
          - objectName: "secret_2"
            objectVersion: "v1"
            objectType: "csms"
```

After the SPC object is updated, the new **secret_2** is quickly mounted to the pod that references the SPC object.

Viewing Component Logs

View the pod where the add-on runs.

```
$ kubectl get pod -n kube-system
NAME                          READY   STATUS      RESTARTS   AGE
csi-secrets-store-76tj2       3/3     Running     0          11h
dew-provider-hm5fq            1/1     Running     0          11h
```

View pod logs of the dew-provider component.

```
$ kubectl logs dew-provider-hm5fq -n kube-system
...Log information omitted...
...
```

View the pod logs of the csi-secrets-store component. As the pod of the csi-secrets-store component contains multiple containers, you must run the **-c** command to specify a container when viewing pod logs. The secrets-store container is the major service container of the add-on and contains the majority of the logs.

```
$ kubectl logs csi-secrets-store-76tj2 -c secrets-store -n kube-system
...Log information omitted...
...
```

Change History

Table 14-103 Release history

Add-on Version	Supported Cluster Version	New Feature
1.1.57	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.
1.1.33	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.1.3	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	Fixed some issues.
1.1.2	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	• CCE clusters 1.29 are supported. • Secrets can be created during CSMS secret synchronization.
1.0.31	v1.21 v1.23 v1.25 v1.27 v1.28	• CCE clusters 1.27 are supported. • CCE clusters 1.28 are supported.
1.0.9	v1.19 v1.21 v1.23 v1.25	None
1.0.6	v1.19 v1.21 v1.23 v1.25	None
1.0.3	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.
1.0.2	v1.19 v1.21 v1.23	CCE clusters 1.23 are supported.
1.0.1	v1.19 v1.21	Actively detects SecretProviderClass object changes.

14.7.2 Container Image Signature Verification

Introduction

The Container Image Signature Verification add-on (formerly swr-cosign) is used to sign image files and verify their integrity and authenticity. This prevents image files from being tampered with or implanted with malicious code.

Notes and Constraints

- An SWR Enterprise instance has been created before you use the image signature verification function.

Installing the Add-on

Step 1 Log in to the CCE console and click a cluster name to access the cluster. In the navigation pane, choose **Add-ons**, locate **Container Image Signature Verification** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** based on the cluster scale. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  The small specification specifies that the add-on runs in one pod, which is ideal for clusters with fewer than 50 concurrent image downloads. The large specification specifies that the add-on runs in two pods, which are more appropriate for clusters with fewer than 300 concurrent image downloads.
- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Configure the add-on parameters.

Table 14-104 swr-cosign parameters

Parameter	Description
KMS key	Select a key. Only EC_P256, EC_P384, and SM2 are supported. You can add a key using KMS.
Signature Verification Image	Enter a regular expression for the path to a signature verification image. For example, if you enter docker.io/**, the signature of the image in the docker.io image repository will be verified. To verify the signatures of all images, enter **.

Step 4 Click **Install**.

After the add-on is installed, select the cluster and choose **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 14-105 Add-on components

Component	Description	Resource Type
swr-cosign	swr-cosign verifies digital signatures of image files to ensure that the image files are not tampered with.	Deployment

How to Use

Step 1 Install swr-cosign and configure the KMS key and image address as instructed in **Installing the Add-on**.

Step 2 Add the **policy.sigstore.dev/include:true** label to the namespace that requires signature verification.

1. In the navigation pane of the cluster console, click **Namespaces**.
2. Locate the namespace to be verified. In the **Operation** column, choose **More** > **Manage Label**.
3. Add a label.
   - Key: **policy.sigstore.dev/include**
   - Value: **true**
4. Click **OK**.

Step 3 Check whether image signature verification is enabled.

1. In the navigation pane of the cluster console, click **Workloads**.
2. Click **Create Workload** in the upper right corner.
3. Select the namespace where the label was added, enter the unsigned image path, and set other parameters as instructed in **Creating a Deployment**.
4. Click **Create Workload**.

   Unsigned images will be blocked. The following information is displayed:

   ```
   admission webhook "policy.sigstore.dev" denied the request: validation failed: failed policy: cip-key-secret-match: spec.template.spec.containers[0].image ...
   ```

Step 4 Sign an image.

1. Log in to the SWR enterprise repository and access an existing repository.
2. In the navigation pane, choose **Security** > **Image Signature** and create a signature rule.
   - **Name**: Name the signature rule.

- **Organization**: Select a container image organization.
- **Application Scope**:
  - **Image**: Select the image to be signed. You can also use a regular expression to match multiple images.
  - **Version**: Select an image version. If this parameter is left blank or set to ******, all versions of the image are matched.
- **Signing Method**: Select **KMS**.
- **Signature Key**: Select a KMS key. The key must be the same as that used during add-on installation.
- **Trigger Mode**:
  - **Manual**: After a signature rule is created, manually execute the rule to sign the image.
  - **Event + manual**: The image can be signed by events or manually.
- **Description**: Enter the description of the rule.

3. After the signature rule is created, click **Execute** to sign the selected image.
4. After the image is signed, in the navigation pane, choose **Artifact Repositories** > **Image Repositories** and click the image name to view the image details. The image already has a signature attachment.

Step 5 Go back to the CCE console, and check whether the signed image can be used to create a workload successfully.

----End

Change History

Table 14-106 Release history

Add-on Version	Supported Cluster Version	New Feature
1.0.26	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Clusters 1.31 are supported.
1.0.2	v1.23 v1.25 v1.27	Clusters 1.27 are supported.
1.0.1	v1.23 v1.25	Supports verification of container image signatures.

14.8 Other Add-ons

14.8.1 Kubernetes Dashboard

Introduction

Kubernetes Dashboard is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself, by running commands.

With Kubernetes Dashboard, you can:

- Deploy containerized applications to a Kubernetes cluster.
- Diagnose containerized application problems.
- Manage cluster resources.
- View applications running in a cluster.
- Create and modify Kubernetes resources (such as Deployments, jobs, and DaemonSets).
- Check errors that occur in a cluster.

For example, you can scale a Deployment, perform a rolling update, restart a pod, or deploy a new application.

Open source community: **https://github.com/kubernetes/dashboard**

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Kubernetes Dashboard** on the right, and click **Install**.

Step 2 (Supported by 3.0.2 and later versions) In the window that slides out from the right, configure **Specifications**.

Table 14-107 Add-on configuration

Parameter	Description
Add-on Specifications	Select **Preset** or **Custom** for the add-on specifications.
Containers	If you selected **Custom**, you can adjust the container specifications as needed.

Step 3 In the **Parameters** area, configure the following parameters:

- **Access Mode**: NodePort is supported. This add-on can be accessed through the EIP bound to the node. If the cluster node does not have an EIP bound to it, then this function will not be available.

- **Certificate**: Configure a certificate for the dashboard.
  - Use a custom certification.

    You need to fill in the **Certificate File** and **Certificate Private Key** in PEM format by referring to the example.
  - Use the default certificate.

  NOTICE

  The default certificate generated by the dashboard is invalid, which affects the normal access to the dashboard through a browser. You are advised to manually upload a valid certificate so that the browser can verify your access and secure your connection.

Step 4 Click **Install**.

----End

Accessing the dashboard Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **Add-ons** in the navigation pane. On the page displayed, verify that the dashboard add-on is in the **Running** state and click **Access**.

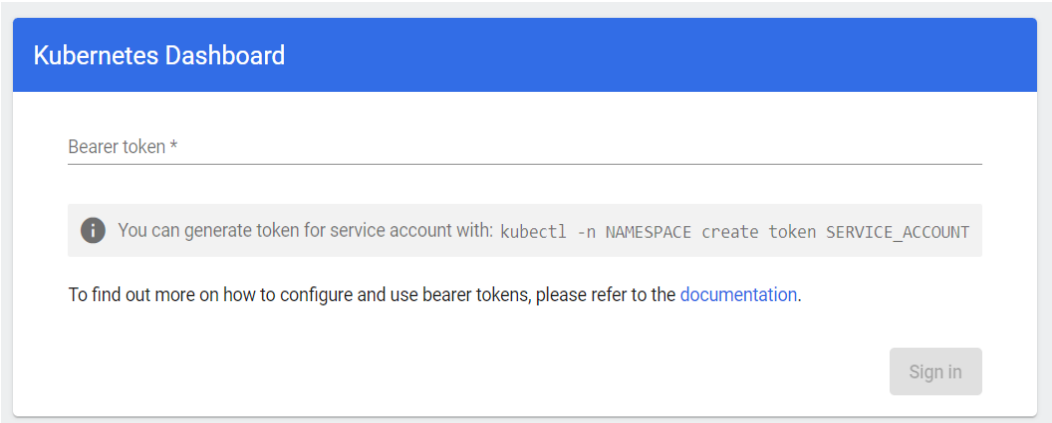
Step 2 Copy the token in the dialog box displayed.

Step 3 On the dashboard login page, select **Token**, paste the copied token, and click **SIGN IN**.

NOTE

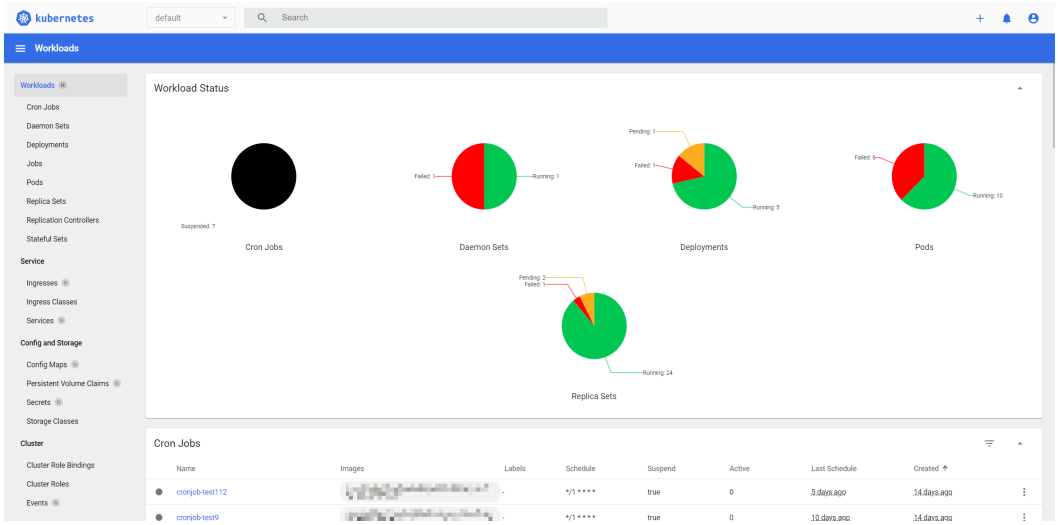
By default, this add-on does not support login using kubeconfig authenticated by certificate. You are advised to use the token mode for login. For details, see **https://github.com/kubernetes/dashboard/issues/2474#issuecomment-348912376**.

Figure 14-16 Token login



Step 4 View the dashboard page as shown in **Figure 14-17**.

Figure 14-17 Dashboard page



----End

Modifying Permissions

After the dashboard is installed, the initial role can only view a majority of resources that are displayed on the dashboard. To apply for the permissions to perform other operations on the dashboard, modify RBAC authorization resources in the background.

Procedure

Modify the **kubernetes-dashboard-minimal** rule in the ClusterRole.

For details about how to use RBAC authorization, visit **https://kubernetes.io/docs/reference/access-authn-authz/rbac/**.

Components

Table 14-108 Add-on components (for versions earlier than 3.0.2)

Component	Description	Resource Type
Dashboard	Visualized monitoring UI	Deployment

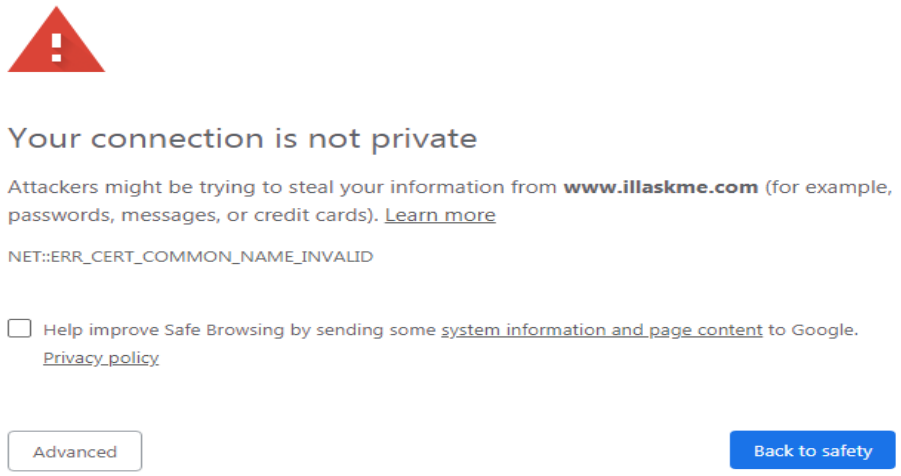
Table 14-109 Add-on components (for 3.0.2 or later)

Component	Description	Resource Type
dashboard-api	Back-end API of Dashboard, which provides a RESTful API for communicating with the Kubernetes API server. This component obtains cluster information from the Kubernetes API server and reports the information to the web component.	Deployment
dashboard-auth	Authentication module of Dashboard, which provides a token-based authentication mechanism. When a user logs in to the Dashboard, this component checks whether the token provided by the user is valid. If the token is valid, the user can access the Dashboard.	Deployment
dashboard-web	Front-end UI of Dashboard, which provides a user-friendly UI for viewing and managing Kubernetes clusters. This component is a single-page application based on the React framework and can be accessed through HTTPS.	Deployment
metrics-scraper	Metric collection module of Dashboard, which collects metric data from Kubernetes clusters and reports the data to the web component. Metrics-scraper uses Heapster or Metrics Server to get metric data, and it stores the data in the Kubernetes API server.	Deployment
dashboard-kong	Open-source API gateway component on which Dashboard depends, which helps manage APIs and implement authentication and authorization.	Deployment

Troubleshooting Access Problems

When Google Chrome is used to access the dashboard, the error message "ERR_CERT_INVALID", instead of the login page, is displayed. The possible cause is that the default certificate generated by the dashboard does not pass Google Chrome verification. There are two solutions to this problem:

Figure 14-18 Error message displayed on Google Chrome



- Solution 1: Use the Firefox browser to access the dashboard. In the **Exceptions** area of the **Proxy Settings** page, add the dashboard address to the addresses that will bypass the proxy server. Then, the dashboard login page will be displayed.
- Solution 2: Start Google Chrome with the **--ignore-certificate-errors** flag to ignore the certificate error.

  Windows: Save the dashboard address. Close all active Google Chrome windows. Press the Windows key + R to display the **Run** dialog box. Enter **chrome --ignore-certificate-errors** in the **Run** dialog box to open a new Google Chrome window. In the address bar, enter the dashboard address to open the login page.

Change History

Table 14-110 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
4.0.1	v1.30 v1.31	• CCE clusters 1.31 are supported. • Updated the add-on to its community version v7.10.0.	7.10.0
3.0.25	v1.27 v1.28 v1.29	Fixed some issues.	7.3.2
3.0.4	v1.27 v1.28 v1.29	Fixed some issues.	7.3.2

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.0.2	v1.27 v1.28 v1.29	• Supported clusters of v1.27, v1.28, and v1.29. • Updated the add-on to its community version 7.3.2.	7.3.2
2.2.27	v1.21 v1.23 v1.25	Fixed some issues.	2.7.0
2.2.7	v1.21 v1.23 v1.25	None	2.7.0
2.2.5	v1.21 v1.23 v1.25	Synchronized time zones used by the add-on and the node.	2.7.0
2.2.3	v1.21 v1.23 v1.25	None	2.7.0
2.1.1	v1.19 v1.21 v1.23	• CCE clusters 1.23 are supported. • Updated the add-on to its community version v2.5.0.	2.5.0
2.0.10	v1.15 v1.17 v1.19 v1.21	CCE clusters 1.21 are supported.	2.0.0
2.0.4	v1.15 v1.17 v1.19	Added the default seccomp profile.	2.0.0
2.0.3	v1.15 v1.17 v1.19	CCE clusters 1.15 are supported.	2.0.0
2.0.2	v1.17 v1.19	CCE clusters 1.19 are supported.	2.0.0
2.0.1	v1.15 v1.17	-	2.0.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.0.0	v1.17	Enabled interconnection with CCE v1.17	2.0.0

14.8.2 OpenKruise

Introduction

OpenKruise is an extended component suite for Kubernetes. It leverages CRDs to offer advanced workload and application management features, including automatic deployment, release, O&M, and availability protection for cloud native applications. This simplifies and streamlines application management, making it more efficient.

OpenKruise has the following core capabilities:

- Advanced workloads: It contains a set of advanced workloads, such as CloneSets and Advanced StatefulSets.
- Application sidecar management: It provides many SidecarSets to make sidecar inject easier and offers other capabilities like in-place sidecar upgrade.
- Application security protection: It protects your Kubernetes resources from being interfered by the cascading deletion mechanism.
- Efficient application O&M: It provides many advanced O&M capabilities to help you better manage applications. For example, you can use the ImagePullJob CRD to pull some images from any nodes beforehand or restart containers in a running pod.

Open source community: **https://github.com/openkruise/kruise**

Notes and Constraints

If you have deployed the community OpenKruise in your cluster, uninstall it and then install the CCE OpenKruise add-on. Otherwise, the add-on may fail to be installed.

Precautions

OpenKruise has added webhooks to its open-source components. The default pod failure policy has been set to **Fail** by the community. This means that if kruise-controller-manager becomes unavailable, operations like pod creation and deletion will be blocked. Before using this add-on, it is important to carefully assess the risks and configure HA for kruise-controller-manager to ensure that the webhook server can handle requests properly.

NOTICE

OpenKruise is an open-source add-on that CCE has selected, adapted, and integrated into its services. CCE offers comprehensive technical support, but is not responsible for any service disruptions caused by defects in the open-source software, nor does it provide compensation or additional services for such disruptions. It is highly recommended that users regularly upgrade their software to address any potential issues.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **OpenKruise** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications as needed.

- If you selected **Preset**, you can choose between **Small** or **Large** based on the cluster scale. The system will automatically set the number of add-on pods and resource quotas according to the preset specifications. You can see the configurations on the console.

  The small specification specifies that the add-on runs in one pod, which is ideal for clusters with fewer than 50 nodes. The large specification specifies that the add-on runs in two pods, which are suitable for clusters with more than 50 nodes.
- If you selected **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.

Step 3 Check whether to enable **Kruise-daemon**.

kruise-daemon, a new DaemonSet component, has been added to OpenKruise. It provides image warm-up and container restart.

NOTICE

If you install OpenKruise v1.0.3 in a cluster of v1.25 or later, kruise-daemon cannot run on a Docker node. In this case, use a containerd node. For details, see **Components**.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-111 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-112 Add-on components

Component	Description	Resource Type
kruise-controller-manager	Core component of OpenKruise controller, which includes admission webhooks for Kruise CRDs and pods. kruise-controller-manager creates webhook configurations to configure which resources need to be processed and provides Services that can be called by kube-apiserver.	Deployment
kruise-daemon	Deployed on each node through DaemonSets to provide functions such as image warm-up and container restart.	DaemonSet

NOTICE

Since version 1.24, the Kubernetes community no longer supports Dockershim. CCE uses cri-dockerd as an alternative to Dockershim in clusters of v1.25 or later to accommodate users' Docker habits. However, the OpenKruise community does not support cri-dockerd. For details, see **issue**. This issue will be solved in later versions.

Therefore, if you install OpenKruise v1.0.3 in a cluster of v1.25 or later, kruise-daemon cannot run on a Docker node. In this case, use a containerd node.

Troubleshooting

When a workload is being created, the following error occurs:

```
Error creating: Internal error occurred: failed calling webhook "mpod.kb.io": failed to call webhook: Post "https://kruise-webhook-service.kube-system.svc:443/mutate-pod?timeout=10s": dial tcp 10.247.10.181:443: connect: connection refused
```

The issue is caused by the unavailability of the kruise-controller-manager component. This results in the interception of pod creation, update, and deletion operations in certain namespaces (excluding the **kube-system** namespace or namespaces without the **control-plane: openkruise** label).

Solution

Restore kruise-controller-manager. The causes and solutions are as follows:

- The resources required by kruise-controller-manager are not enough for kruise-controller-manager to be properly scheduled. You are advised to configure more resources for the add-on.

- A scheduling or affinity policy configured for kruise-controller-manager may prevent the pod from being scheduled correctly. You are advised to check the scheduling policy and configure a proper one to allow kruise-controller-manager to be scheduled smoothly.

Change History

Table 14-113 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.0.23	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	1.5.4
1.0.12	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	1.5.4
1.0.3	v1.23 v1.25 v1.27 v1.28 v1.29	The OpenKruise add-on is now available.	1.5.4

14.8.3 Gatekeeper

Introduction

Gatekeeper is a customizable cloud native policy controller based on OPA. It helps enhance policy execution and governance and provides more security policy rules that comply with Kubernetes application scenarios in clusters.

Open source community: **https://github.com/open-policy-agent/gatekeeper**

For how to use the add-on, see the **Gatekeeper documentation**.

Notes and Constraints

If you have deployed the community Gatekeeper in your cluster, uninstall it and then install the CCE Gatekeeper add-on. Otherwise, the add-on may fail to be installed.

Precautions

Gatekeeper's webhooks can impact the utilization of fundamental Kubernetes resources. It is crucial to use the webhooks for services and carefully assess the potential risks associated with the add-on.

NOTICE

Gatekeeper is an open-source add-on that CCE has selected, adapted, and integrated into its services. CCE offers comprehensive technical support, but is not responsible for any service disruptions caused by defects in the open-source software, nor does it provide compensation or additional services for such disruptions. It is highly recommended that users regularly upgrade their software to address any potential issues.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **Gatekeeper** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 14-114 Add-on configuration

Parameter	Description
Add-on Specifications	Select **HA**, **Standalone**, or **Custom**.
Pods	Number of pods that will be created to match the selected add-on specifications. If you selected **Custom**, you can adjust the number of pods as needed. High availability is not possible with a single pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.
Containers	If you selected **Custom**, you can adjust the container specifications as needed.

Step 3 Change the values of the configurations that you want to modify. For details, see the **parameters in GitHub**.

Step 4 Configure deployment policies for the add-on pods.

NOTE

- Scheduling policies do not take effect on add-on pods of the DaemonSet type.
- When configuring multi-AZ deployment or node affinity, ensure that there are nodes meeting the scheduling policy and that resources are sufficient in the cluster. Otherwise, the add-on cannot run.

Table 14-115 Configurations for add-on scheduling

Parameter	Description
Multi-AZ Deployment	• **Preferred**: Deployment pods of the add-on will be preferentially scheduled to nodes in different AZs. If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to different nodes in that AZ. • **Equivalent mode**: Deployment pods of the add-on are evenly scheduled to the nodes in the cluster in each AZ. If a new AZ is added, you are advised to increase add-on pods for cross-AZ HA deployment. With the Equivalent multi-AZ deployment, the difference between the number of add-on pods in different AZs will be less than or equal to 1. If resources in one of the AZs are insufficient, pods cannot be scheduled to that AZ. • **Forcible**: Deployment pods of the add-on are forcibly scheduled to nodes in different AZs. There can be at most one pod in each AZ. If nodes in a cluster are not in different AZs, some add-on pods cannot run properly. If a node is faulty, add-on pods on it may fail to be migrated.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Specify node**: Specify the nodes where the add-on is deployed. If you do not specify the nodes, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Specify node pool**: Specify the node pool where the add-on is deployed. If you do not specify the node pools, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Customize affinity**: Enter the labels of the nodes where the add-on is to be deployed for more flexible scheduling policies. If you do not specify node labels, the add-on will be randomly scheduled based on the default cluster scheduling policy. If multiple custom affinity policies are configured, ensure that there are nodes that meet all the affinity policies in the cluster. Otherwise, the add-on cannot run.

Parameter	Description
Toleration	Using both taints and tolerations allows (not forcibly) the add-on Deployment to be scheduled to a node with the matching taints, and controls the Deployment eviction policies after the node where the Deployment is located is tainted. The add-on adds the default tolerance policy for the **node.kubernetes.io/not-ready** and **node.kubernetes.io/unreachable** taints, respectively. The tolerance time window is 60s. For details, see **Configuring Tolerance Policies**.

Step 5 Click **Install**.

----End

Components

Table 14-116 Add-on components

Component	Description	Resource Type
gatekeeper-audit	Provide audit-related information.	Deployment
gatekeeper-controller-manager	Provide Gatekeeper webhooks to control Kubernetes resources based on custom policies.	Deployment

How to Use the Add-on

The following shows how to use Gatekeeper to enforce a constraint that requires a pod created in a specific namespace to have a label called **test-label**. For details, see **How to use Gatekeeper**.

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a **test-gatekeeper**namespace for testing:

```
kubectl create ns test-gatekeeper
```

Step 3 Create a policy template for checking labels:

```
kubectl apply -f - <<EOF
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
```



```
      kind: K8sRequiredLabels
    validation:
      openAPIV3Schema:
        properties:
          labels:
            type: array
            items:
              type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredlabels
        violation[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
EOF
```

Step 4 Create a constraint for the preceding policy template. This constraint enforces the requirement for a pod created in the **test-gatekeeper** namespace to have the label **test-label**.

```
kubectl apply -f - <<EOF
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: pod-must-have-test-label
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - test-gatekeeper
  parameters:
    labels: ["test-label"]
EOF
```

Step 5 Verify the constraint effect.

1. Create a pod that does not have the label **test-label** in the **test-gatekeeper** namespace:

   ```
   kubectl -n test-gatekeeper run test-deny --image=nginx --restart=Never
   ```

   Expected output:

   ```
   Error from server (Forbidden): admission webhook "validation.gatekeeper.sh" denied the request: [pod-must-have-test-label] you must provide labels: {"test-label"}
   ```

   The pod that does not have the label **test-label** cannot be created in the **test-gatekeeper** namespace.

2. Create a pod that has the label **test-label** in the **test-gatekeeper** namespace:

   ```
   kubectl -n test-gatekeeper run test -l test-label=test --image=nginx --restart=Never
   ```

   Check the pod. The pod has been created.

   ```
   kubectl get pod test -n test-gatekeeper
   ```

Based on the previous verification, Gatekeeper is used to enforce a constraint that requires a pod created in a specific namespace to have a label called **test-label**.

----End

Change History

Table 14-117 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.0.22	v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters 1.31 are supported.	3.16.3
1.0.10	v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	CCE clusters 1.30 are supported.	3.16.3
1.0.3	v1.23 v1.25 v1.27 v1.28 v1.29	The Gatekeeper add-on is now available.	3.16.3

14.8.4 CCE Cluster Backup & Recovery (EOM)

Introduction

The CCE Cluster Backup & Recovery add-on (formerly e-backup) offers cluster backup and restoration. It backs up application data and service data to OBS and provides local and remote data backup.

Notes and Constraints

- Do not add, delete, or modify the cluster during the backup/restore. Otherwise, the backup/restore may fail or become incomplete.
- If you change the cluster, you are advised to wait for 15 minutes until the cluster is stable and then perform the backup operation.
- When EVS disk snapshots are used for backup, only EVS PVs are supported and the snapshot constraints apply (for example, cross-AZ restoration is not supported). The pricing is the same as EVS disk snapshots.
- When restic is used for backup, data of EVS, SFS, SFS Turbo, and OBS PVs is backed up and uploaded to the OBS backup repository.

- restic creates a snapshot for the data at the backup time point and uploads the data, which does not affect subsequent data read and write. However, restic does not verify the file content and service consistency. restic restrictions apply.
- The memory occupied by restic is related to the size of the PV data backed up for the first time. If the data size is greater than 500 GB, you are advised to use the migration methods provided by cloud storage services. If you use this add-on, you can modify the resource quotas of the restic container by referring to the operation guide.
- You can use Hooks to ensure service data consistency for stateful applications during backup, for example, synchronizing memory data to files.
- During the restore, you can adjust configurations to adapt to the environment differences before and after the migration.
  - An application can be restored from the original namespace to another specified namespace. However, confirm that the application is not accessed through a fixed Service during the restore.
  - You can change the image address (repo) of the application to another image path. The image name and tag remain unchanged during the restore.
  - You can change the name of the storage class used by the application to a new one. Note that the backend storage resources must be of the same type, for example, from block storage to block storage.
- Velero and restic constraints apply. For example, during the restore, the Service will clear the ClusterIP to better adapt to the differences between the source and target Kubernetes clusters.

Installing the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. Locate the e-backup add-on and click **Install**.

Step 2 On the **Install Add-on** page, select the cluster, set parameters, and click **Install**.

The following parameter is supported:

volumeWorkerNum: number of concurrent volume backup jobs. The default value is **3**.

----End

Using the Add-on

e-backup uses OBS buckets as backup storage location. Before backing up data, perform operations in **Preparing Keys** and **Creating a Storage Location**.

Backups can be **immediate** and **scheduled**. Restores can be **immediate**.

Preparing Keys

Step 1 Obtain an access key.

Log in to the CCE console, move the cursor to the username in the upper right corner, and choose **My Credentials**. In the navigation pane on the left, choose **Access Keys**. On the page displayed, click **Add Access Key**.

Step 2 Create a key file and format it into a string using Base64.

```
# Create a key file.
$ vi credential-for-huawei-obs
HUAWEI_CLOUD_ACCESS_KEY_ID=your_access_key
HUAWEI_CLOUD_SECRET_ACCESS_KEY=your_secret_key

# Use Base64 to format the string.
$ base64 -w 0 credential-for-huawei-obs
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

Step 3 Create a secret.

Create a secret using the following YAML content:

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    secret.everest.io/backup: 'true'   # The secret is used by e-backup to access the backup storage location.
  name: secret-secure-opaque
  namespace: velero                  # The value must be velero. The secret must be in the same namespace as e-backup.
type: cfe/secure-opaque
data:
  # String obtained after the credential file is Base64-encoded.
  cloud: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

- The secret must be in the same namespace as e-backup, that is, **velero**.
- **secret.data** stores the secret for accessing OBS. The key must be **cloud**, and the value is the string Base64-encoded in **Step 2**. Generally, the displayed Base64-encoded string contains line breaks. Manually delete them when writing the string into **secret.data**.
- The secret must be labeled **secret.everest.io/backup: true**, indicating that the secret is used to manage the backup storage location.

----End

Creating a Storage Location

Create a Kubernetes resource object used by e-backup as the backup storage location to obtain and detect information about the backend OBS.

```
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  name: backup-location-001
  namespace: velero            # The object must be in the same namespace as e-backup.
spec:
  config:
    endpoint: obs.{regionname}.myhuaweicloud.com   # OBS endpoint
  credential:
    name: secret-secure-opaque   # Name of the created secret
    key: cloud                   # Key in secret.data
  objectStorage:
    bucket: tools-cce        # OBS bucket name
    prefix: for-backup       #Subpath name
  provider: huawei           # Uses the OBS service.
```

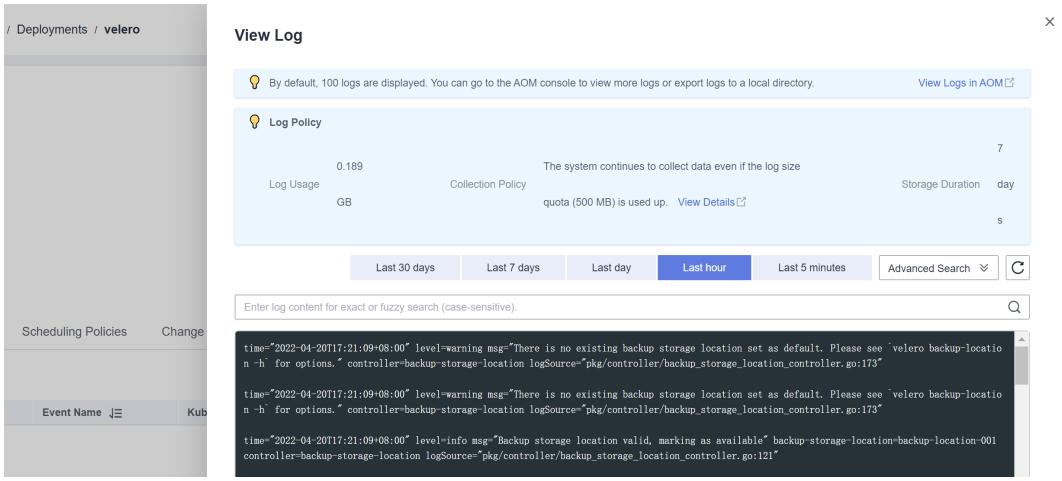
- The **prefix** field is optional, and other fields are mandatory. The value of **provider** is fixed at **huawei**.

- You can obtain the endpoint from **Regions and Endpoints**. Ensure that all nodes in the cluster can access the endpoint. If the endpoint does not carry a protocol header (http or https), **https** is used by default.
- Correctly set **name** and **key** in the credential. Otherwise, e-backup cannot access the storage location.

After the creation is complete, wait for 30 seconds for check and synchronization of the backup storage location. Then check whether **PHASE** is **Available**. The location is available only when the value is **Available**.

```
$ kubectl get backupstoragelocations.velero.io backup-location-001 -n velero
NAME                 PHASE       LAST VALIDATED   AGE   DEFAULT
backup-location-001  Available   23s              23m
```

If **PHASE** is not **Available** for a long time, you can view e-backup logs to locate the fault. After e-backup is installed, a workload named **velero** is created in the **velero** namespace, recorded in the logs of velero.



Immediate Backup

The backup process starts immediately and stops upon completion. This mode is commonly used for cloning and migration.

You can use the Backup manifest below and run **kubectl create** to create a backup task.

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: backup-01
  namespace: velero
spec:
  includedNamespaces:
  - nginx
  - mysql
  labelSelector:
    matchExpressions:
    - key: direction
      operator: In
      values:
      - back
      - front
    matchLabels:
      app: nginx
      backup: velero
```



```
runMode: Normal
appData:
  volumes: Restic
hooks:
  resources:
  - name: hook01
    includedNamespaces:
    - nginx
    labelSelector: {}
    pre:
    - exec:
        command:
        - /bin/sh
        - -c
        - echo hello > hello.txt && echo goodbye > goodbye.txt
        container: container-0
        onError: Fail
        timeout: 30s
    post:
    - exec:
        command:
        - /bin/sh
        - -c
        - echo hello > hello.txt && echo goodbye > goodbye.txt
        container: container-0
        onError: Fail
        timeout: 30s
storageLocation: backup-location-001
ttl: 720h0m0s
```

Parameters:

- Backup parameters
  - **storageLocation**: (**mandatory**) name of the backup storage location where the data to be backed up is stored.
  - **ttl**: duration for storing backups in the location, after which the backups are deleted. The value must be in the specified format. **h**, **m**, and **s** indicate hour, minute, and second, respectively. For example, **24h** indicates one day, and **3h4m5s** indicates three hours, four minutes, and five seconds. The default value is **720h0m0s** (30 days).
- Resource filtering: The following parameters are used as filters. The intersection of these fields, if all configured, is used to filter all resources in the cluster.
  - **includedNamespaces** and **excludedNamespaces**: whether to back up resources in certain namespaces. These two parameters conflict with each other. Choose one to configure. By default, all namespaces are selected.
  - **labelSelector**: backs up resources with specific labels. The working principle is the same as that in Kubernetes.
  - **runMode**: (**mandatory**) backup mode. Value options include **Normal** (backing up applications and data), **AppOnly** (backing up applications only), **DataOnly** (backing up data only), and **DryRun** (not backing up applications and data; for verification only).
- Service data backup: The generated service data can be backed up through Everest snapshots (supported only when the EVS PVs as the data volumes) and restic backups (which back up all data volumes except hostPath ones). These two modes can be used together.
  - **appData**: PV data backup mode. The value can be **Restic** or **Snapshot** (not used by default). The **Snapshot** mode takes effect only when the

storage supports snapshots and the CSI snapshot plugin is deployed in the cluster.

- **hook**: Hooks are the commands executed before or after a backup to precisely manage your backups. A hook is similar to the **kubectl exec** command and applies to pods only.
  - **includedNamespaces** and **excludedNamespaces**: whether to execute a hook on pods in certain namespaces. These two parameters conflict with each other. Choose one to configure. By default, all namespaces are selected.
  - **labelSelector**: executes a hook on pods with certain labels. The working principle is the same as that in Kubernetes.
  - **command**: command to be executed.
  - **container**: name of the container on which the command is executed. Defaults to the first container when there are multiple containers in the pod.
  - **onError**: action to take when the hook fails to be executed. The value can be **Continue** or **Fail**. Defaults to **Fail**.
  - **Continue** indicates that the subsequent operations go on regardless of hook execution failures. **Fail** indicates that subsequent operations will not continue upon a hook execution failure.
  - **timeout**: hook execution timeout, after which the hook fails. Defaults to 30s.

  Hook failures affect only pods. The backup of other objects such as Services is not affected.

  Hooks are not globally available. If the pod to execute a hook on is not selected as the backup object, the hook will not be executed. It can be considered that you further filter the objects to be backed up through **includedNamespaces** or **excludedNamespaces**.

NOTE

All configurable items are described above. The following provides some **backup configuration suggestions**.

- Retain backups by day (24 hours).
- Use **includeNamespace** to specify the backup scope because in most cases, applications are deployed in a specific namespace. Use **labelSelector** to control backup objects more precisely. Before this, all target objects must have corresponding labels. Using **includeNamespace** and **labelSelector** together can satisfy most scenarios.
- When using Restic to back up service data, if you are not familiar with the OUT/IN mode, you can skip adding annotations to the pods that require volume backup. Instead, set **defaultVolumesToRestic** to **true** to back up the service data of the pod volumes. The value **false** indicates no backups.
- Use hooks to precisely control your backups. Avoid long-time running tasks. Do not directly operate the file system when running the commands in the hook.

After the backup is complete, run the following commands to view the backup status (**status**):

```
$ kubectl -n velero get backups backup-01 -o yaml | grep  "phase"
  phase: Completed

$ kubectl -n velero get backups backup-01 -o yaml
......
```

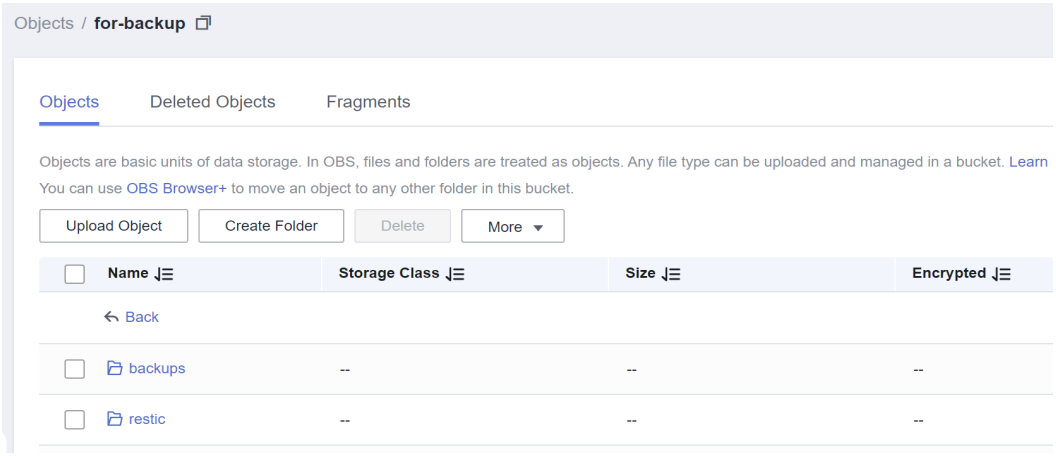


```
status:
  ......
```

Backup statuses

- **FailedValidation**: The backup manifest is incorrectly configured. Check **Backup.Status.ValidationErrors** to find the cause.
- **InProgress**: The backup is in progress.
- **Completed**: The backup is complete and no error occurs.
- **PartiallyFailed**: The backup is complete, but an error (such as hook execution error) occurs during the backup of certain objects.
- **Failed**: The backup fails, and an error that affects the entire process occurs.
- **Deleting**: The backup is being deleted.

After the initial backup is complete, the **backups** and **restic** folders are displayed in the OBS bucket.



Backup logs are stored in an OBS bucket. Assume that the backup name is **backup-001**. Go to the OBS console, locate the storage location based on the configured bucket name and sub-path name, go to the **backups/backup-01** directory, and find the **backup-01-logs.gz** file. Then, download, decompress, and view the logs.

Periodic Backup

Data is backed up periodically as configured. This mode is commonly used for disaster recovery.

You can use the Schedule manifest below and run the **kubectl create** command to create a schedule. You can label the schedule as required. The labels you add in the manifest will be attached to the backups created by the schedule. After a schedule is created in a cluster, a backup is performed immediately. Then, data is backed up periodically as specified.

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: schedule-backup-001
  namespace: velero
spec:
  schedule: 0 */10 * * *
  template:
```



```
  runMode: Normal
  hooks: {}
  includedNamespaces:
  - nginx
  - mysql
  labelSelector:
    matchExpressions:
    - key: direction
      operator: In
      values:
      - back
      - front
    matchLabels:
      app: nginx
      backup: velero
  storageLocation: backup-location-001
  ttl: 720h0m0s
```

Parameters:

- **schedule**: execution time of periodic backups. The @every format and standard Linux cron expressions are supported.
  - @every *N*Unit: **N** is a positive integer. The units **s**, **m**, and **h**, stand for seconds, minutes, and hours, respectively. For example, **@every 2h30m** indicates that the backup is triggered every 2 hours and 30 minutes.
  - Cron expression: The five values stand for minutes, hours, day-of-month, month, and day-of-week, respectively.
- **template**: backup manifest, which is the same as **spec** in **Immediate Backup**.

Deleting a Backup

You can delete the backup objects and related objects (such as backups, restorations, and schedules) from a cluster and delete backups from the storage location when a large amount of backup data is generated.

You can use the DeleteBackupRequest manifest below and run the **kubectl create** command to create a backup deletion request.

```
apiVersion: velero.io/v1
kind: DeleteBackupRequest
metadata:
  name: backup-001-delete
  namespace: velero
spec:
  backupName: backup-001  # Name of the backup to be deleted.
```

Query the status.

```
$ kubectl -n velero get deletebackuprequests backup-001-delete -o yaml | grep " phase"
  phase: InProgress
```

- **InProgress**: The deletion task is in progress.
- **Processed**: The deletion task has been processed.

CAUTION

- The **Processed** state indicates that e-backup has processed the task but may not complete it. You can check the errors in the **deletebackuprequest.status.errors** field. If e-backup correctly and completely processes the deletion task, the **DeleteBackupRequest** object is also deleted.
- Do not manually delete the content in the storage location (OBS bucket).

Immediate Restore

Use an immediate backup as the data source and restore data to another namespace or cluster. This mode applies to all scenarios.

You can use the Restore manifest below and run the **kubectl create** command to create a backup deletion request.

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-01
  namespace: velero
spec:
  backupName: backup-01
  hooks:
    resources:
    - name: restore-hook-1
      includedNamespaces:
      - mysql
      labelSelector: {}
      postHooks:
      - init:
          initContainers:
          - name: restore-hook-init1
            image: alpine:latest
            volumeMounts:
            - mountPath: /restores/pvc1-vm
              name: pvc1-vm
            command:
            - /bin/ash
            - -c
            - echo -n "FOOBARBAZ" >> /restores/pvc1-vm/foobarbaz
          - name: restore-hook-init2
            image: alpine:latest
            volumeMounts:
            - mountPath: /restores/pvc2-vm
              name: pvc2-vm
            command:
            - /bin/ash
            - -c
            - echo -n "DEADFEED" >> /restores/pvc2-vm/deadfeed
      - exec:
          execTimeout: 1m
          waitTimeout: 5m
          onError: Fail
          container: mysql
          command:
          - /bin/bash
          - '-c'
          - 'while ! mysql_isready; do sleep 1; done'
      - exec:
          container: mysql
          waitTimeout: 6m
          execTimeout: 1m
          onError: Continue
```



```
      command:
      - /bin/bash
      - '-c'
      - 'mysql < /backup/backup.sql'
includedNamespaces:
- nginx
- mysql
namespaceMapping:
  nginx: nginx-another
  mysql: mysql-another
labelSelector: {}
preserveNodePorts: false
storageClassMapping:
  disk: csi-disk
  obs: csi-obs
imageRepositoryMapping:
  quay.io/coreos:  swr.ap-southeast-1.myhuaweicloud.com/everest
```

Parameters:

- Data source

  backupName: (**mandatory**) immediate backup that is used as the data source.

- Resource filtering parameters: similar to those in **Immediate Backup**.
- Customized processing
  - **namespaceMapping**: restores the backup data to another namespace. The value is a mapping in the format of *Source: Target*. The new namespace does not need to exist in the destination cluster.
  - **storageClassMapping**: changes the storageClassName used by backup resources such as PVs and PVCs. The storageClass types must be the same.
  - **imageRepositoryMapping**: changes the **images** field of the backup. It is used for repository mapping, excluding the change of the image name and tag (to prevent the migration and upgrade from being coupled). For example, after you migrate **quay.io/coreos/etcd:2.5** to SWR, you can use **swr.ap-southeast-1.myhuaweicloud.com/everest/etcd:2.5** in the local image repository. The configuration format is as follows: **quay.io/coreos: swr.ap-southeast-1.myhuaweicloud.com/everest**
  - **preserveNodePorts**: If you set this parameter to **false**, the system preserves only the nodePorts you configure, not those automatically generated by the Service.
- **hooks**: You can add init hooks (used to add initContainers to the pod) and exec hooks (used to execute some commands). For details about how to configure an init hook, see the definition of initContainers in Kubernetes. The following describes the overall hook configuration and the parameters of an exec hook.
  - **includedNamespaces** and **excludedNamespaces**: whether to execute a hook on pods in certain namespaces. These two parameters conflict with each other. Choose one to configure. By default, all namespaces are selected.
  - **labelSelector**: executes a hook on pods with certain labels. The working principle is the same as that in Kubernetes.
  - **command**: command to be executed.

  - **container**: name of the container on which the command is executed. Defaults to the first container when there are multiple containers in the pod.
  - **onError**: action to take when the hook fails to be executed. The value can be **Continue** or **Fail**. Defaults to **Fail**.
  - **Continue** indicates that the subsequent operations go on regardless of hook execution failures. **Fail** indicates that subsequent operations will not continue upon a hook execution failure.
  - **execTimeout**: hook execution timeout, after which the hook fails. Defaults to 30s.
  - **waitTimeout**: timeout period from the time when e-backup prepares to execute the hook to the time when the container starts to execute the hook. If this period is exceeded, the hook fails. The default value is 0s, indicating that there is no timeout limit.

NOTE

- Select a correct data source and ensure that the backup is in the **Completed** state.
- Set the parameters related to resource filtering only when necessary.
- Service data is restored by e-backup based on the selected backup mode. No manual configurations or operations are required.
- For details about how to use hooks, see the usage suggestions in *Immediate Backup*. You can skip **waitTimeout** unless necessary.
- You are advised to restore what has been backed up to a new namespace to avoid misconfigurations that may disable the restored application.

After the restoration is complete, run the following commands to view the restoration status (**status**):

```
$ kubectl -n velero get restores restore-01 -o yaml | grep " phase"
  phase: Completed

$ kubectl -n velero get restores restore-01 -o yaml
......
status:
  ......
```

Status description

- **FailedValidation**: The restore manifest is incorrectly configured. Check **Restore.Status.ValidationErrors** to find the cause.
- **InProgress**: The restore is in progress.
- **Completed**: The restore is complete and no error occurs.
- **PartiallyFailed**: The restore is complete, but an error (such as hook execution error) occurs during the restore of certain objects.
- **Failed**: The restore fails, and an error that affects the entire process occurs.

Check the logs, warnings, and errors generated during the restore.

Assume that the restore name is **restore-01**. Go to the OBS console, locate the storage location based on the configured bucket name and sub-path name, and go to the **restores/restore-01** directory. The following two files exist:

- **restore-01-logs.gz**: log file, which can be downloaded, decompressed, and viewed.

- **restore-01-results.gz**: restore result file, including warnings and errors.

Change History

Table 14-118 Release history

Add-on Version	Supported Cluster Version	New Feature
1.2.0	v1.15 v1.17 v1.19 v1.21	• Supported EulerOS 2.0 (SP5, SP9). • Supported security hardening. • Optimized some functions.

14.8.5 Kubernetes Web Terminal (EOM)

The Kubernetes Web Terminal add-on (formerly web-terminal) is a lightweight terminal server that allows you to use kubectl on the web UI. It provides a remote CLI via web browser and HTTP, and can be easily integrated into an independent system. You can directly access the add-on as a service to obtain information and log in to a server through cmdb.

web-terminal can run on all operating systems supported by Node.js and does not depend on local modules. It is fast and easy to install and supports multiple sessions.

Open source community: **https://github.com/rabchev/web-terminal**

Notes and Constraints

- This add-on can be installed only in clusters of v1.21 or earlier. Arm clusters are not supported.
- web-terminal is no longer evolved. Use CloudShell instead.
- When installing web-terminal to use kubectl, you must log in using your cloud account or as an IAM user with the CCE Administrator permission. For details about how to control the kubectl permission, see **Controlling web-terminal Permissions**.
- The web-terminal add-on can be used only after CoreDNS is installed in a cluster.

Precautions

The web-terminal add-on can be used to manage CCE clusters. Keep the login password secure to prevent unexpected operation.

Installing the Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**, locate **web-terminal** on the right, and click **Install**.

Step 2 Configure the following parameters:

- **Access Mode**: The value is fixed to **NodePort**. The web-terminal add-on is accessed in the NodePort mode by default and can be used only if any node in the cluster has an EIP. If this access type is selected, an EIP must be bound to the cluster where web-terminal will be installed.
- **Username**: The default value is **root** and cannot be changed.
- **Password**: password for logging in to web-terminal. Keep secure the password. The web-terminal add-on can be used to manage CCE clusters. Keep the login password secure to prevent unexpected operation.
- **Confirm Password**: Enter the password again.

Step 3 Click **Install**.

----End

Connecting to a Cluster Using the web-terminal Add-on

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **Add-ons**.

Step 2 Find **web-terminal** on the right and click **Access**.

----End

Controlling web-terminal Permissions

After web-terminal is installed, kubectl uses the ClusterRole cluster-admin by default and can operate Kubernetes resources in the cluster. To manually change to another ClusterRole, you can run **kubectl edit clusterrolebinding web-terminal** to modify the web-terminal ServiceAccount.

For details about ClusterRole and ClusterRoleBinding, see **Namespace Permissions (Kubernetes RBAC-based)**.

NOTICE

- Manually configured web-terminal permissions could be reset after the web-terminal add-on is upgraded. You are advised to back up the configurations before the upgrade.
- Before using kubectl to modify ClusterRoleBindings, ensure that kubectl has been configured with the required permissions.

Change History

Table 14-119 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.1.12	v1.15 v1.17 v1.19 v1.21	• CCE clusters 1.21 are supported.	**0.6.6**
1.1.6	v1.15 v1.17 v1.19	• Added the default seccomp profile.	**0.6.6**
1.1.5	v1.15 v1.17 v1.19	• CCE clusters 1.15 are supported.	**0.6.6**
1.1.3	v1.17 v1.19	• CCE clusters 1.19 are supported.	**0.6.6**
1.0.6	v1.15 v1.17	• Adds pod security policies.	**0.6.6**
1.0.5	v1.9 v1.11 v1.13 v1.15 v1.17	• Clusters 1.17 are supported.	**0.6.6**

15 Helm Chart

15.1 Overview of a Chart

CCE provides a console for managing Helm charts, helping you easily deploy applications using the charts and manage applications on the console. CCE uses Helm v3.8.2 and supports the upload of Helm v3 chart packages. For details, see **Deploying an Application from a Chart**.

You can also use the Helm client to directly deploy applications. If you use the Helm client to deploy applications, version control is not supported. You can use Helm v2 or Helm v3. For details, see **Deploying an Application Through the Helm v2 Client** and **Deploying an Application Through the Helm v3 Client**.

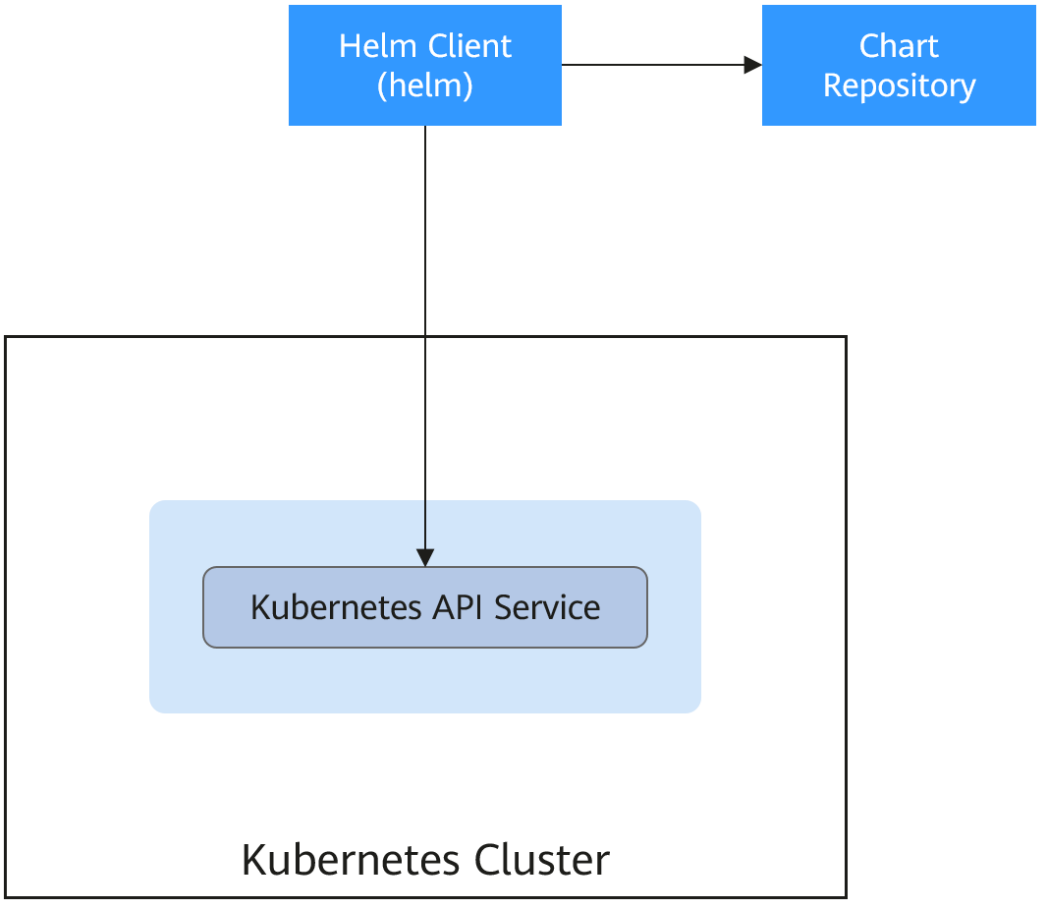
Helm

Helm is a package manager for Kubernetes and manages charts. A Helm chart is a series of YAML files used to encapsulate native Kubernetes applications. When deploying an application, you can customize some metadata of the application for easy application distribution. Application releasors can use Helm to package applications, manage application dependencies and application versions, and release applications to the software repository. After using Helm, users do not need to compile complex application deployment files. They can easily search for, install, upgrade, roll back, and uninstall applications on Kubernetes.

The relationship between Helm and Kubernetes is as follows:

- Helm <–> Kubernetes
- Apt <–> Ubuntu
- Yum <–> CentOS
- Pip <–> Python

The following figure shows the solution architecture:



Helm can help application orchestration for Kubernetes:

- Manages, edits, and updates a large number of Kubernetes configuration files.
- Deploys a complex Kubernetes application that contains a large number of configuration files.
- Shares and reuses Kubernetes configurations and applications.
- Supports multiple environments with parameter-based configuration templates.
- Manages the release of applications, including rolling back the application, finding differences (using the **diff** command), and viewing the release history.
- Controls phases in a deployment cycle.
- Tests and verifies the released version.

15.2 Deploying an Application from a Chart

On the CCE console, you can upload a Helm chart package, deploy it, and manage the deployed pods.

NOTICE

CCE has gradually switched to Helm v3 since September 2022. Helm v2 charts are no longer supported on the console. If you cannot switch to Helm v3 for now, you can use the Helm v2 client to manage Helm v2 charts in the background.

Notes and Constraints

- The number of charts that can be uploaded by a single user is limited. The value displayed on the console of each region is the allowed quantity.
- CCE uses Helm v3.8.2 and allows uploading Helm v3 chart packages.
- A chart with multiple versions consumes the same amount of portion of chart quota.
- Users with chart operation permissions can perform multiple operations on clusters. Therefore, exercise caution when assigning users the chart lifecycle management permissions, including uploading charts and creating, deleting, and updating chart releases.

Chart Specifications

The Redis workload is used as an example to illustrate the chart specifications.

- **Naming Requirement**

  A chart package is named in the format of **{name}-{version}**.tgz, where **{version}** indicates the version number in the format of *Major version number.Minor version number.Revision number*, for example, **redis-0.4.2.tgz**.

  NOTE

  The chart name {name} can contain a maximum of 64 characters.

  The version number must comply with the **semantic versioning** rules.

  - The main and minor version numbers are mandatory, and the revision number is optional.
  - The major and minor version numbers and revision number must be integers, greater than or equal to 0, and less than or equal to 99.

- **Directory Structure**

  The directory structure of a chart is as follows:

```
redis/
  templates/
  values.yaml
  README.md
  Chart.yaml
  .helmignore
```

  As listed in **Table 15-1**, the parameters marked with * are mandatory.

Table 15-1 Parameters in the directory structure of a chart

Parameter	Description
* templates	Stores all templates.

Parameter	Description
* values.yaml	Describes configuration parameters required by templates. **NOTICE** Make sure that the image address set in the **values.yaml** file is the same as the image address in the container image repository. Otherwise, an exception occurs when you create a workload, and the system displays a message indicating that the image fails to be pulled. To obtain the image address, perform the following operations: Log in to the CCE console. In the navigation pane, choose **Image Repository** to access the SWR console. Choose **My Images** > **Private Images** and click the name of the uploaded image. On the **Image Tags** tab page, obtain the image address from the pull command. You can click  to copy the command in the **Image Pull Command** column.
README.md	A markdown file, including: ● The workload or services provided by the chart. ● Prerequisites for running the chart. ● Configurations in the **values.yaml** file. ● Information about chart installation and configuration.
* Chart.yaml	Basic information about the chart. Note: The API version of Helm v3 is switched from v1 to v2.
.helmignore	Files or data that does not need to read templates during workload installation.

Uploading a Chart

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane and click **Upload Chart** in the upper right corner.

Step 2 Click **Select File**, select the chart to be uploaded, and click **Upload**.

NOTE

When you upload a chart, the naming rule of the OBS bucket is changed from cce-charts-{*region*}-{**domain_name**} to cce-charts-{*region*}-{**domain_id**}. In the old naming rule, the system converts the **domain_name** value into a Base64 string and uses the first 63 characters. If you cannot find the chart in the OBS bucket with the new name, search for the bucket with the old name.

----End

Creating a Release

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane, choose **App Templates**.

Step 2 On the **My Charts** tab page, click **Install** of the target chart.

Step 3 Set workload installation parameters by referring to **Table 15-2**.

Table 15-2 Installation parameters

Parameter	Description
Instance	Unique name of the chart release.
Namespace	Namespace to which the workload will be deployed.
Select Version	Version of a chart.
Configuration File	You can import and replace the **values.yaml** file or directly edit the chart parameters online. **NOTE** An imported **values.yaml** file must comply with YAML specifications, that is, KEY:VALUE format. The fields in the file are not restricted. The key value of the imported values.yaml must be the same as that of the selected chart package. Otherwise, the values.yaml does not take effect. That is, the key cannot be changed. 1. Click **Select File**. 2. Select the corresponding **values.yaml** file and click **Open**.

Step 4 Click **Install**.

On the **Releases** tab page, you can view the installation status of the release.

----End

Upgrading a Chart-based Workload

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.

Step 2 Click **Upgrade** in the row where the desired workload resides and set the parameters for the workload.

Step 3 Select a chart version for **Chart Version**.

Step 4 Follow the prompts to modify the chart parameters. Confirm the modification and click **Upgrade**.

Step 5 If the execution status is **Upgraded**, the workload has been upgraded.

----End

Rolling Back a Chart-based Workload

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.

Step 2 Choose **More** > **Roll Back** for the release to be rolled back, select the target release version, and roll the release back.

In the workload list, if the status is **Rollback successful**, the workload is rolled back successfully.

----End

Uninstalling a Chart-based Workload

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. Choose **App Templates** in the navigation pane. In the right pane, click the **Releases** tab.

Step 2 Click **More** > **Uninstall** next to the release to be uninstalled, and click **Yes**. Exercise caution when performing this operation because releases cannot be restored after being uninstalled.

----End

15.3 Differences Between Helm v2 and Helm v3 and Adaptation Solutions

Helm v2 stops at version 2.17.0. Currently, Helm v3 is the standard in the Helm community. You are advised to switch your charts to **Helm v3 format** as soon as possible.

Changes since Helm v2:

1. **Removal of Tiller**

   Helm v3 is simpler and easier to use. It removes tiller and directly connects to the API server using kubeconfig, simplifying the security model.

2. **Improved upgrade strategy: 3-way strategic merge patches**

   Helm v2 used a two-way strategic merge patch. During an upgrade, it compared the most recent chart's manifest against the proposed chart's manifest to determine what changes needed to be applied to the resources in Kubernetes. If changes were applied to the cluster out-of-band (such as during a kubectl edit), those changes were not considered. This resulted in resources being unable to roll back to its previous state.

   Helm v3 uses a three-way strategic merge patch. Helm considers the original manifest, its live state, and the new manifest when generating a patch. Helm compares the current live state with the live state of the original manifest, checks whether the new manifest is modified, and automatically supplements the new manifest to generate the final update patch.

   For details and examples, see **https://v3.helm.sh/docs/faq/changes_since_helm2**.

3. **Secrets as the default storage driver**

   Helm v2 used ConfigMaps by default to store release information. In Helm v3, Secrets are now used as the default storage driver.

4. **Release names are now scoped to the namespace**

In Helm v2, the information about each release was stored in the same namespace as Tiller. In practice, this meant that once a name was used by a release, no other release could use that same name, even if it was deployed in a different namespace. In Helm v3, information about a particular release is now stored in the same namespace as the release itself. This means that the release name can be used in different namespaces. The namespace of the application is the same as that of the release.

5. **Verification mode change**

   Helm v3 verifies the chart format more strictly. For example, Helm v3 bumps the apiVersion in Chart.yaml from v1 to v2. For the Chart.yaml of v2, apiVersion must be set to v1. After installing the Helm v3 client, you can run the **helm lint** command to check whether the chart format complies with the Helm v3 specifications.

   Adaptation solution: Adapt the Helm v3 chart based on the Helm official document **https://helm.sh/docs/topics/charts/**. The **apiVersion** field is mandatory.

6. **Removal of the crd-install hook**

   The **crd-install** hook has been removed in favor of the **crds/** directory in Helm v3. Note that the resources in the **crds/** directory are deployed only during the release installation and are not updated during the upgrade. When the resources are deleted, the resources are retained in the **crds/** directory. If the CRD already exists, it will be skipped with a warning during the repeated installation.

   Adaptation solution: According to the **Helm document**, you can hold your CRD in the **crds/** directory or a separate chart. Helm cannot upgrade or delete the CRD. Therefore, you are advised to put the CRD in one chart, and then put any resources that use that CRD in another chart.

7. **Resources that are not created using Helm are not forcibly updated. Releases are not forcibly upgraded by default.**

   The forcible upgrade logic of Helm v3 is changed. After the upgrade fails, the system does not delete and rebuild the Helm v3. Instead, the system directly uses the **put** logic. Therefore, the CCE release upgrade uses the non-forcible update logic by default. Resources that cannot be updated through patches will make the release unable to be upgraded. If a release with the same name exists in the environment and does not have the home tag **app.kubernetes.io/managed-by: Helm** of Helm v3, a conflict message is displayed.

   Adaptation solution: Delete related resources and create them using Helm.

8. **Limit on release historical records**

   Only the latest 10 release versions are retained by default.

For more changes and details, see Helm official documents.

- Differences between Helm v2 and Helm v3: **https://v3.helm.sh/docs/faq/changes_since_helm2**
- How to migrate from Helm v2 to Helm v3: **https://helm.sh/docs/topics/v2_v3_migration**

15.4 Deploying an Application Through the Helm v2 Client

NOTICE

CCE has gradually switched to Helm v3 since September 2022. Helm v2 charts are no longer supported on the console. If you cannot switch to Helm v3 for now, you can use the Helm v2 client to manage Helm v2 charts in the background.

Prerequisites

The Kubernetes cluster created on CCE has been connected to kubectl. For details, see **Using kubectl**.

Precautions

CCE will attempt to convert v2 releases to v3 ones. If you delete a Helm v2 release in the background, the release information is still displayed on the charts page on the CCE console. In this case, delete it.

Installing Helm v2

This section uses Helm v2.17.0 as an example.

For other versions, visit **https://github.com/helm/helm/releases**.

Step 1 Download the Helm client from the VM connected to the cluster.

```
wget https://get.helm.sh/helm-v2.17.0-linux-amd64.tar.gz
```

Step 2 Decompress the Helm package.

```
tar -xzvf helm-v2.17.0-linux-amd64.tar.gz
```

Step 3 Copy Helm to the system path, for example, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

Step 4 RBAC is enabled on the Kubernetes API server. Create the service account name **tiller** for the tiller and assign cluster-admin, a system ClusterRole, to the tiller. Create a tiller resource account as follows:

vim tiller-rbac.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
```



```
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
```

Step 5 Deploy the tiller resource account.

```
kubectl apply -f tiller-rbac.yaml
```

Step 6 Initialize the Helm and deploy the pod of tiller.

```
helm init --service-account tiller --skip-refresh
```

Step 7 Query the status.

```
kubectl get pod -n kube-system -l app=helm
```

Command output:

```
NAME                            READY   STATUS    RESTARTS   AGE
tiller-deploy-7b56c8dfb7-fxk5g   1/1     Running   1          23h
```

Step 8 Query the Helm version.

```
# helm version
Client: &version.Version{SemVer:"v2.17.0", GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe",
GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.17.0", GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe",
GitTreeState:"clean"}
```

----End

Installing the Helm Chart

If the charts provided by CCE do not meet requirements, download a chart and install it.

You can obtain the required chart in the **stable** directory at **https://github.com/helm/charts**, download the chart, and upload it to the node.

1. Download and decompress the obtained chart. Generally, the chart is in ZIP format.
   ```
   unzip chart.zip
   ```
2. Install the Helm chart.
   ```
   helm install aerospike/
   ```
3. After the installation is complete, run the **helm list** command to check the status of the chart releases.

Common Issues

- The following error message is displayed after the **Helm version** command is executed:
  ```
  Client:
  &version.Version{SemVer:"v2.17.0",
  GitCommit:"a690bad98af45b015bd3da1a41f6218b1a451dbe", GitTreeState:"clean"}
  E0718 11:46:10.132102    7023 portforward.go:332] an error occurred
  forwarding 41458 -> 44134: error forwarding port 44134 to pod
  d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable
  to do port forwarding: socat not found.
  Error: cannot connect to Tiller
  ```

  The preceding information is displayed because the socat is not installed. Run the following command to install the socat:

  ```
  yum install socat -y
  ```

- When you run the **yum install socat –y** command on a node running EulerOS 2.9 or Huawei Cloud EulerOS, the following error message is displayed:

No match for argument: socat

Error: Unable to find a match: socat

The image does not contain socat. In this case, manually download the RPM chart and run the following command to install it (replace the RPM chart name with the actual one):

```
rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm
```

Table 15-3 Addresses for downloading socat RPM charts

OS	Where to Download
EulerOS 2.9	• **x86** • **Arm**
Huawei Cloud EulerOS 2.0	• **x86** • **Arm**
Huawei Cloud EulerOS 1.1	**x86**

- When the socat has been installed and the following error message is displayed after the **helm version** command is run:

  ```
  test@local:~/k8s/helm/test$ helm version
  Client: &version.Version{SemVer:"v3.3.0",
  GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
  Error: cannot connect to Tiller
  ```

  The Helm chart reads the configuration certificate in the **.Kube/config** file to communicate with Kubernetes. The preceding error indicates that the kubectl configuration is incorrect. In this case, reconnect the cluster to kubectl. For details, see **Using kubectl**.

- Storage fails to be created after you have connected to cloud storage services.

  This issue may be caused by the **annotation** field in the created PVC. Change the chart name and install the chart again.

- If kubectl is not properly configured, the following error message is displayed after the **helm install** command is run:

  ```
  [root@prometheus-57046 ~]# helm install prometheus/ --generate-name
  WARNING: This chart is deprecated
  Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?timeout=32s": dial tcp
  [::1]:8080: connect: connection refused
  ```

  Solution: Configure kubeconfig for the node. For details, see **Using kubectl**.

15.5 Deploying an Application Through the Helm v3 Client

Prerequisites

- The Kubernetes cluster created on CCE has been connected to kubectl. For details, see **Using kubectl**.
- To pull a public image when deploying Helm, ensure an EIP has been bound to the node.

Installing Helm v3

This section uses Helm v3.3.0 as an example.

For other versions, visit **https://github.com/helm/helm/releases**.

Step 1 Download the Helm client from the VM connected to the cluster.

```
wget https://get.helm.sh/helm-v3.3.0-linux-amd64.tar.gz
```

Step 2 Decompress the Helm package.

```
tar -xzvf helm-v3.3.0-linux-amd64.tar.gz
```

Step 3 Copy Helm to the system path, for example, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

Step 4 Query the Helm version.

```
helm version
version.BuildInfo{Version:"v3.3.0", GitCommit:"e29ce2a54e96cd02ccfce88bee4f58bb6e2a28b6",
GitTreeState:"clean", GoVersion:"go1.13.4"}
```

----End

Installing the Helm Chart

You can use Helm to install a chart. Before using Helm, you may need to understand the following concepts to better use Helm:

- Chart: contains resource definitions and a large number of configuration files of Kubernetes applications.
- Repository: stores shared charts. You can download charts from the repository to a local path for installation or install them online.
- Release: running result of after a chart is installed in a Kubernetes cluster using Helm. A chart can be installed multiple times in a cluster. A new release will be created for each installation. A MySQL chart is used as an example. To run two databases in a cluster, install the chart twice. Each database has its own release and release name.

For more details, see **Using Helm**.

Step 1 Search for a chart from the **Artifact Hub** repository recommended by Helm and configure the Helm repository.

```
helm repo add {repo_name} {repo_addr}
```

The following uses the **WordPress chart** as an example:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Step 2 Run the **helm install** command to install the chart.

```
helm install {release_name} {chart_name} --set key1=val1
```

For example, to install WordPress, the WordPress chart added in **Step 1** is **bitnami/wordpress**, the release name is **my-wordpress**, and mandatory parameters have been configured.

```
helm install my-wordpress bitnami/wordpress \
    --set mariadb.primary.persistence.enabled=true \
    --set mariadb.primary.persistence.storageClass=csi-disk \
    --set mariadb.primary.persistence.size=10Gi \
    --set persistence.enabled=false
```


Run the **helm show values** *{chart_name}* command to view the configurable options of the chart. For example, to view the configurable items of WordPress, run the following command:

```
helm show values bitnami/wordpress
```

Step 3 View the installed chart release.

```
helm list
```

----End

Common Issues

- The following error message is displayed after the **Helm version** command is executed:

  ```
  Client:
  &version.Version{SemVer:"v3.3.0",
  GitCommit:"012cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
  E0718 11:46:10.132102    7023 portforward.go:332] an error occurred
  forwarding 41458 -> 44134: error forwarding port 44134 to pod
  d566b78f997eea6c4b1c0322b34ce8052c6c2001e8edff243647748464cd7919, uid : unable
  to do port forwarding: socat not found.
  Error: cannot connect to Tiller
  ```

  The preceding information is displayed because the socat is not installed. Run the following command to install the socat:

  ```
  yum install socat -y
  ```

- When you run the **yum install socat –y** command on a node running EulerOS 2.9 or Huawei Cloud EulerOS, the following error message is displayed:

  ```
  No match for argument: socat
  Error: Unable to find a match: socat
  ```

  The node image does not contain socat. In this case, manually download the RPM chart and run the following command to install it (replace the RPM chart name with the actual one):

  ```
  rpm -i socat-1.7.3.2-8.oe1.x86_64.rpm
  ```

  Table 15-4 Addresses for downloading socat RPM charts

OS	Where to Download
EulerOS 2.9	• **x86** • **Arm**
Huawei Cloud EulerOS 2.0	• **x86** • **Arm**
Huawei Cloud EulerOS 1.1	**x86**

- When the socat has been installed and the following error message is displayed after the **helm version** command is run:

  ```
  $ helm version
  Client: &version.Version{SemVer:"v3.3.0",
  GitCommit:"021cb0ac1a1b2f888144ef5a67b8dab6c2d45be6", GitTreeState:"clean"}
  Error: cannot connect to Tiller
  ```

  The Helm chart reads the configuration certificate in **.Kube/config** to communicate with Kubernetes. The preceding error indicates that the kubectl configuration is incorrect. In this case, reconnect the cluster to kubectl. For details, see **Using kubectl**.

- Storage fails to be created after you have connected to cloud storage services.

  This issue may be caused by the **annotation** field in the created PVC. Change the chart name and install the chart again.

- If kubectl is not properly configured, the following error message is displayed after the **helm install** command is run:

  ```
  # helm install prometheus/ --generate-name
  WARNING: This chart is deprecated
  Error: Kubernetes cluster unreachable: Get "http://localhost:8080/version?timeout=32s": dial tcp
  [::1]:8080: connect: connection refused
  ```

  Solution: Configure kubeconfig for the node. For details, see **Using kubectl**.

15.6 Converting a Release from Helm v2 to v3

Context

CCE fully supports Helm v3. This section guides you to convert a Helm v2 release to Helm v3. Helm v3 discards or reconstructs some Helm v2 functions at the bottom layer. Therefore, the conversion is risky to some extent. Simulation is required before conversion.

For details, see the **community documentation**.

Precautions

- Helm v2 stores release information in ConfigMaps. Helm v3 does so in secrets.
- When you query, update, or operate a Helm v2 release on the CCE console, CCE will attempt to convert the release to v3. If you operate in the background, convert the release by following the instructions below.

Conversion Process (Without Using the Helm v3 Client)

Step 1 Download the helm 2-to-3 conversion plugin on the CCE node.

```
wget https://github.com/helm/helm-2to3/releases/download/v0.10.2/helm-2to3_0.10.2_linux_amd64.tar.gz
```

Step 2 Decompress the plugin package.

```
tar -xzvf helm-2to3_0.10.2_linux_amd64.tar.gz
```

Step 3 Perform the simulated conversion.

Take the test-convert release as an example. Run the following command to simulate the conversion: If the following information is displayed, the simulation is successful.

```
# ./2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

Step 4 Perform the conversion. If the following information is displayed, the conversion is successful.

```
# ./2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```



```
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid conflicts with the migrated v3 release.
v2 release information should only be removed using `helm 2to3` cleanup and when all releases have been migrated over.
```

Step 5 After the conversion is complete, simulate the resource clearance. After the simulation, clear the v2 release resources.

Simulated clearance:

```
# ./2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

Formal clearance:

```
# ./2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" d
```

----End

Conversion Process (Using the Helm v3 Client)

Step 1 Install the Helm v3 client. For details, see **Installing Helm v3**.

Step 2 Install the conversion plugin.

```
# helm plugin install https://github.com/helm/helm-2to3
Downloading and installing helm-2to3 v0.10.2 ...
https://github.com/helm/helm-2to3/releases/download/v0.10.2/helm-2to3_0.10.2_linux_amd64.tar.gz
Installed plugin: 2to3
```

Step 3 Check whether the plugin has been installed.

```
# helm plugin list
NAME VERSION DESCRIPTION
2to3  0.10.2      migrate and cleanup Helm v2 configuration and releases in-place to Helm v3
```

Step 4 Perform the simulated conversion.

Take the test-convert release as an example. Run the following command to simulate the conversion: If the following information is displayed, the simulated conversion is successful.

```
# helm 2to3 convert --dry-run --tiller-out-cluster -s configmaps test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
```

Step 5 Perform the conversion. If the following information is displayed, the conversion is successful.


```
# helm 2to3 convert --tiller-out-cluster -s configmaps test-convert
Release "test-convert" will be converted from Helm v2 to Helm v3.
[Helm 3] Release "test-convert" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" will be created.
[Helm 3] ReleaseVersion "test-convert.v1" created.
[Helm 3] Release "test-convert" created.
Release "test-convert" was converted successfully from Helm v2 to Helm v3.
Note: The v2 release information still remains and should be removed to avoid conflicts with the migrated v3 release.
v2 release information should only be removed using `helm 2to3` cleanup and when all releases have been migrated over.
```

Step 6 After the conversion, you can view the converted release by running **helm list**.

```
# helm list
NAME            NAMESPACE   REVISION UPDATED                         STATUS      CHART           APP VERSION
test-convert    default     1        2022-08-29 06:56:28.166918487 +0000 UTC deployed    test-helmold-1
```

Step 7 After the conversion is complete, simulate the resource clearance. After the simulation, clear the v2 release resources.

Simulated clearance:

```
# helm 2to3 cleanup --dry-run --tiller-out-cluster -s configmaps --name test-convert
NOTE: This is in dry-run mode, the following actions will not be executed.
Run without --dry-run to take the actions described below:
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
```

Formal clearance:

```
# helm 2to3 cleanup --tiller-out-cluster -s configmaps --name test-convert
WARNING: "Release 'test-convert' Data" will be removed.

[Cleanup/confirm] Are you sure you want to cleanup Helm v2 data? [y/N]: y
Helm v2 data will be cleaned up.
[Helm 2] Release 'test-convert' will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" will be deleted.
[Helm 2] ReleaseVersion "test-convert.v1" deleted.
[Helm 2] Release 'test-convert' deleted.
Helm v2 data was cleaned up successfully.
```

----End

16 Permissions

16.1 Permissions Overview

CCE permissions management allows you to assign permissions to IAM users and user groups under your tenant accounts. CCE combines the advantages of Identity and Access Management (IAM) and Kubernetes Role-based Access Control (RBAC) authorization to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, cluster-scoped authorization, and namespace-wide authorization.

CCE allows you to manage permissions on clusters and related resources at a finer granularity, for example, to control the access of employees in different departments to cloud resources.

This section describes the CCE permissions management mechanism and related concepts. If your account has met your service requirements, you can skip this section.

CCE Permissions Management

CCE permissions are described as follows:

- **Cluster-level permissions**: Cluster-level permissions management evolves out of the system policy authorization feature of IAM. IAM users in the same user group have the same permissions. On IAM, you can configure system policies to describe which IAM user groups can perform which operations on cluster resources. For example, you can grant user group A to create and delete cluster X, add a node, or install an add-on, while granting user group B to view information about cluster X.

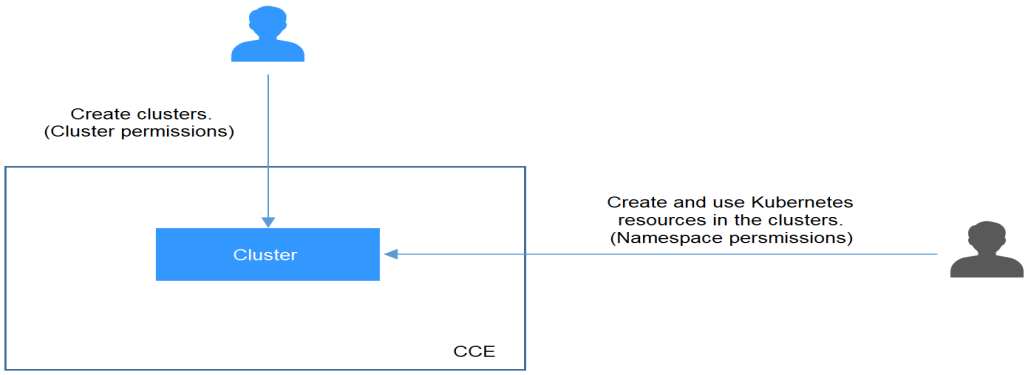
  Cluster-level permissions involve non-Kubernetes APIs in CCE clusters and support fine-grained IAM policies and enterprise project management.
- **Namespace-level permissions**: You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. CCE has also been enhanced based on open-source capabilities. It supports RBAC authorization based on IAM user or user group, and RBAC authentication on access to APIs using IAM tokens.

  Namespace-level permissions involve CCE Kubernetes APIs and are enhanced based on the Kubernetes RBAC capabilities. Namespace-level permissions can

be granted to IAM users or user groups for authentication and authorization, but are independent of fine-grained IAM policies.

In general, you configure CCE permissions in two scenarios. The first is creating and managing clusters and related resources, such as nodes. The second is creating and using Kubernetes resources in the cluster, such as workloads and Services.

Figure 16-1 Illustration on CCE permissions



These permissions allow you to manage resource users at a finer granularity.

Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). **Table 16-1** lists the namespace permissions of different users.

Table 16-1 Differences in namespace permissions

User	Clusters of v1.13 and Later
User with the Tenant Administrator permissions (for example, an account)	All namespace permissions
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

kubectl Permissions

You can use **kubectl** to access Kubernetes resources in a cluster.

When you access a cluster using kubectl, CCE uses the kubeconfig.json file generated on the cluster for authentication. This file contains user information,

based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in **Table 16-1**.

Federated Users

IAM provides the identity provider function to implement federated identity authentication based on Security Assertion Markup Language (SAML) or OpenID Connect. This function allows users in your management system to access the cloud platform through single sign-on (SSO).

Users who log in through federated identity authentication are called federated users. Federated users are equivalent to IAM users.

Pay attention to the following for federated users to use CCE:

- When a user creates a CCE cluster, the cluster-admin permission is granted to the user by default. The user ID of a federated user changes upon each login and logout. Therefore, the user is displayed as deleted on the **Permissions** page of the CCE console. Do not manually delete the permission, otherwise, the authentication fails. In this case, you are advised to grant the cluster-admin permission to a user group on CCE and add federated users to the user group.
- Federated users cannot create permanent access keys (AKs/SKs). In scenarios where AKs/SKs are required (for example, when creating OBS-related PVs/PVCs), only you or an IAM user can create the AK/SK and share them with the federated users. An access key contains the permissions granted to a user, so it is recommended that the federated user request an IAM user in the same group to create an access key.

Supported Actions

CCE provides system-defined policies that can be directly used in IAM. You can also create custom policies to supplement system-defined policies for more refined access control. Operations supported by policies are specific to APIs. The following are common concepts related to policies:

- Permissions: statements in a policy that allow or deny certain operations.
- APIs: REST APIs that can be called by a user who has been granted specific permissions.
- Actions: specific operations that are allowed or denied in a custom policy.
- Dependencies: actions which a specific action depends on. When allowing an action for a user, you also need to allow any existing action dependencies for that user.
- IAM projects/Enterprise projects: the authorization scope of a custom policy. A custom policy can be applied to IAM projects or enterprise projects or both. Policies that contain actions for both IAM and enterprise projects can be used and applied for both IAM and Enterprise Management. Policies that contain actions only for IAM projects can be used and applied to IAM only. For details about the differences between IAM and enterprise management, see **What Are the Differences Between IAM and Enterprise Management?**

NOTE

The check mark (√) and cross symbol (x) respectively indicate that an action takes effect or does not take effect for the corresponding type of projects.

CCE supports the following actions in custom policies.

Table 16-2 Cluster management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining clusters in a project	GET /api/v3/projects/{project_id}/clusters	cce:cluster:list	√	√
Obtaining a cluster	GET /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:get	√	√
Creating a cluster	POST /api/v3/projects/{project_id}/clusters	cce:cluster:create	√	√
Updating a cluster	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:update	√	√
Deleting a cluster	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}	cce:cluster:delete	√	√
Upgrading a cluster	POST /api/v2/projects/:projectid/clusters/:clusterid/upgrade	cce:cluster:upgrade	√	√
Waking up a cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/awake	cce:cluster:start	√	√
Hibernating a cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/operation/hibernate	cce:cluster:stop	√	√
Changing the specifications of a cluster	POST /api/v2/projects/{project_id}/clusters/:clusterid/resize	cce:cluster:resize	√	√
Obtaining the certificate of a cluster	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/clustercert	cce:cluster:get	√	√

Table 16-3 Node management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining all nodes in a cluster	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes	cce:node:list	√	√
Obtaining a node	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:get	√	√
Creating a node	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes	cce:node:create	√	√ **NOTE** If you use enterprise project authorization to create a node, you need to add the global permission of **evs:quota:get**.
Updating a node	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:update	√	√
Deleting a node	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodes/{node_id}	cce:node:delete	√	√

Table 16-4 Job management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining information about a job	GET /api/v3/projects/{project_id}/jobs/{job_id}	cce:job:get	√	√
Listing all jobs	GET /api/v2/projects/{project_id}/jobs	cce:job:list	√	√

Permission	API	Action	IAM Project	Enterprise Project
Deleting one or all jobs	DELETE /api/v2/projects/{project_id}/jobs DELETE /api/v2/projects/{project_id}/jobs/{job_id}	cce:job:delete	√	√

Table 16-5 Node pool management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining all node pools in a cluster	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools	cce:nodepool:list	√	√
Obtaining a node pool	GET /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:get	√	√
Creating a node pool	POST /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools	cce:nodepool:create	√	√
Updating a node pool	PUT /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:update	√	√
Deleting a node pool	DELETE /api/v3/projects/{project_id}/clusters/{cluster_id}/nodepools/{nodepool_id}	cce:nodepool:delete	√	√

Table 16-6 Chart management actions

Permission	API	Action	IAM Project	Enterprise Project
Updating a chart	PUT /v2/charts/{id}	cce:chart:update	√	x
Uploading a chart	POST /v2/charts	cce:chart:upload	√	x

Permission	API	Action	IAM Project	Enterprise Project
Listing all charts	GET /v2/charts	cce:chart:list	√	x
Obtaining information about a chart	GET /v2/charts/{id}	cce:chart:get	√	x
Deleting a chart	DELETE /v2/charts/{id}	cce:chart:delete	√	x

Table 16-7 Release management actions

Permission	API	Action	IAM Project	Enterprise Project
Updating a release	PUT /v2/releases/{name}	cce:release:update	√	√
Listing all releases	GET /v2/releases	cce:release:list	√	√
Creating a release	POST /v2/releases	cce:release:create	√	√
Obtaining information about a release	GET /v2/releases/{name}	cce:release:get	√	√
Deleting a release	DELETE /v2/releases/{name}	cce:release:delete	√	√

Table 16-8 Storage management actions

Permission	API	Action	IAM Project	Enterprise Project
Creating a PersistentVolumeClaim	POST /api/v1/namespaces/{namespace}/cloudpersistentvolume-claims	cce:storage:create	√	√

Permission	API	Action	IAM Project	Enterprise Project
Deleting a PersistentVolumeClaim	DELETE /api/v1/namespaces/{namespace}/cloudpersistentvolume-claims/{name}	cce:storage:delete	√	√
Listing all volumes	GET /storage/api/v1/namespaces/{namespace}/listvolumes	cce:storage:list	√	√

Table 16-9 Add-on management actions

Permission	API	Action	IAM Project	Enterprise Project
Creating an add-on instance	POST /api/v3/addons	cce:addonInstance:create	√	√
Obtaining an add-on instance	GET /api/v3/addons/{id}?cluster_id={cluster_id}	cce:addonInstance:get	√	√
Listing all add-on instances	GET /api/v3/addons?cluster_id={cluster_id}	cce:addonInstance:list	√	√
Deleting an add-on instance	DELETE /api/v3/addons/{id}?cluster_id={cluster_id}	cce:addonInstance:delete	√	√
Updating an add-on instance	PUT /api/v3/addons/{id}	cce:addonInstance:update	√	√

Table 16-10 Quota management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining quota details	GET /api/v3/projects/{project_id}/quotas	cce:quota:get	√	√

16.2 Granting Cluster Permissions to an IAM User

CCE cluster-level permissions are assigned based on **IAM system policies** and **custom policies**. You can use user groups to assign permissions to IAM users.

CAUTION

- Cluster permissions are granted to users for operating cluster-related resources only (such as clusters and nodes). To operate Kubernetes resources like workloads and Services, you must be granted the **namespace permissions** as well.
- When viewing a cluster on the CCE console, the information displayed depends on the namespace permissions. If you have no namespace permissions, you cannot view the resources in the cluster. For details, see **Permission Dependency of the CCE Console**.

Prerequisites

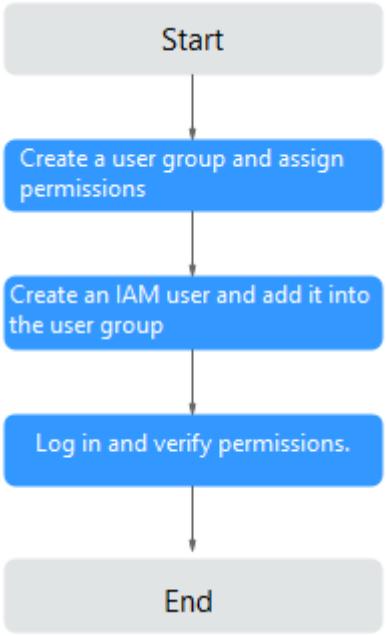
- Before granting permissions to user groups, get familiar with the system policies listed in **Permissions** for CCE. To grant permissions for other services, learn about all **System-defined Permissions** supported by IAM.
- A user with the Security Administrator role (for example, your account) has all IAM permissions except role switching. Only these users can view user groups and their permissions on the **Permissions** page on the CCE console.

Configuration

On the CCE console, when you choose **Permissions** > **Cluster-Level Permissions** to create a user group, you will be directed to the IAM console to complete the process. After the user group is created and its permissions are configured, you can view the information on the **Cluster-Level Permissions** tab page. This section describes the operations in IAM.

Process Flow

Figure 16-2 Process of granting CCE permissions



1. **Create a user group and assign permissions** to it.

   On the IAM console, create a user group and grant it CCE permissions (CCE ReadOnlyAccess as an example).

   NOTE

   CCE is deployed by region. On the IAM console, select **Region-specific projects** when assigning CCE permissions.

2. **Create a user and add it to a user group**.

   Create a user on the IAM console and add the user to the group created in **1**.

   NOTICE

   IAM users need programmatic and management console access to use CCE.

3. **Log in** and verify permissions.

   Log in to the management console as the user you created, and verify that the user has the assigned permissions.

   - Choose **Service List** > **Cloud Container Engine**. Then click **Buy Cluster** on the CCE console. If the operation failed, the **CCE ReadOnlyAccess** policy is in effect.
   - Choose another service from **Service List**. If a message appears indicating that you have insufficient permissions to perform the operation, the **CCE ReadOnlyAccess** policy is in effect.

System-defined Roles

Roles are a type of coarse-grained authorization mechanism that defines service-level permissions based on user responsibilities. Only a limited number of service-level roles are available for authorization. Roles are not ideal for fine-grained authorization and least privilege access.

The preset system role for CCE in IAM is **CCE Administrator**. When assigning this role to a user group, you must also select other roles and policies on which this role depends, such as **Tenant Guest**, **Server Administrator**, **ELB Administrator**, **OBS Administrator**, **SFS Administrator**, **APM FullAccess**, and **SWR Admin**. For more information about dependencies, see **System-defined Permissions**.

System-defined Policies

The system policies preset for CCE in IAM are **CCE FullAccess** and **CCE ReadOnlyAccess**.

- **CCE FullAccess**: common operation permissions on CCE cluster resources, excluding the namespace-level permissions for the clusters (with Kubernetes RBAC enabled) and the privileged administrator operations, such as agency configuration and cluster certificate generation
- **CCE ReadOnlyAccess**: permissions to view CCE cluster resources, excluding the namespace-level permissions of the clusters (with Kubernetes RBAC enabled)

NOTE

When purchasing a cluster or node that is billed on a yearly/monthly basis, add **custom policies** and configure payment permissions such as **bss:*:*** for the Billing Center.

Table 16-11 Permissions granted by CCE FullAccess

Action	Specific Action	Description
cce:*:*	cce:cluster:create	Create a cluster.
	cce:cluster:delete	Delete a cluster.
	cce:cluster:update	Update a cluster. For example, update cluster node scheduling parameters and provide RBAC support to clusters.
	cce:cluster:upgrade	Upgrade a cluster.
	cce:cluster:start	Wake up a cluster.
	cce:cluster:stop	Hibernate a cluster.
	cce:cluster:list	List all clusters.
	cce:cluster:get	Obtain cluster details.
	cce:node:create	Add a node.
	cce:node:delete	Delete one or more nodes.

Action	Specific Action	Description
	cce:node:update	Update a node. For example, update the node name.
	cce:node:get	Obtain node details.
	cce:node:list	List all nodes.
	cce:nodepool:create	Create a node pool.
	cce:nodepool:delete	Delete a node pool.
	cce:nodepool:update	Update a node pool.
	cce:nodepool:get	Obtain a node pool.
	cce:nodepool:list	List all node pools in a cluster.
	cce:release:create	Create a release.
	cce:release:delete	Delete a release.
	cce:release:update	Update a release.
	cce:job:list	List all cluster jobs.
	cce:job:delete	Delete one or more cluster jobs.
	cce:job:get	Obtain a specific cluster job.
	cce:storage:create	Create a storage volume.
	cce:storage:delete	Delete a storage volume.
	cce:storage:list	List all volumes.
	cce:addonInstance:create	Create an add-on instance.
	cce:addonInstance:delete	Delete an add-on instance.
	cce:addonInstance:update	Update an add-on instance.
	cce:addonInstance:get	Obtain an add-on instance.
	cce:addonTemplate:get	Obtain an add-on template.
	cce:addonInstance:list	List all add-on instances.
	cce:addonTemplate:list	List all add-on templates.
	cce:chart:list	List all charts.

Action	Specific Action	Description
	cce:chart:delete	Delete a chart.
	cce:chart:update	Update a chart.
	cce:chart:upload	Upload a chart.
	cce:chart:get	Obtain a chart.
	cce:release:get	Obtain a release.
	cce:release:list	List all releases.
	cce:userAuthorization:get	Obtain CCE user authorization.
	cce:userAuthorization:create	Create CCE user authorization.
ecs:*:*	None	Perform all operations on Elastic Cloud Server (ECS).
evs:*:*	None	Perform all operations on Elastic Volume Service (EVS). EVS disks can be attached to cloud servers and expanded to a higher capacity whenever needed.
vpc:*:*	None	Perform all operations on VPC, including enhanced ELB load balancers. A cluster must run in a VPC. When creating a namespace, create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.
bms:*:get*	None	View BMS resource details.
bms:*:list*	None	List all BMS resources.
ims:*:get*	None	View IMS resource details.
ims:*:list*	None	List all IMS resources.
elb:*:get	None	View ELB resource details.
elb:*:list	None	List all ELB resources.
nat:*:get	None	View NAT Gateway resource details.
nat:*:list	None	List all NAT Gateway resources.
sfs:*:get*	None	View SFS resource details.
sfs:shares:ShareAction	None	Share SFS resources for scaling.

Action	Specific Action	Description
sfsturbo:*:get*	None	View SFS Turbo resource details.
sfsturbo:shares:ShareAction	None	Share SFS Turbo resources for scaling.
tms:resourceTags:list	None	List TMS resources.
kps:domainKeypairs:list	None	List DEW SSH keys.
kps:domainKeypairs:get	None	View DEW SSH keys.
kms:cmk:get	None	View DEW keys.
kms:cmk:list	None	List DEW keys.
aom:*:get	None	View AOM resource details.
aom:*:list	None	List AOM resources.
aom:autoScalingRule:*	None	Perform all operations on AOM auto scaling rules.
apm:icmgr:*	None	Perform operations on the ICAgent in Application Performance Management (APM).
lts:*:*	None	Perform all operations on Log Tank Service (LTS).
smn:*:*	None	Perform all operations on SMN.

Table 16-12 Permissions granted by CCE ReadOnlyAccess

Action	Specific Action	Description
cce:*:get	cce:cluster:get	Obtain cluster details.
	cce:node:get	Obtain node details.
	cce:job:get	Obtain a specific cluster job.
	cce:addonInstance:get	Obtain an add-on instance.
	cce:addonTemplate:get	Obtain an add-on template.
	cce:chart:get	Obtain a chart.
	cce:nodepool:get	Obtain a node pool.
	cce:release:get	Obtain a release.

Action	Specific Action	Description
	cce:userAuthorization:get	Obtain CCE user authorization.
cce:*:list	cce:cluster:list	List all clusters.
	cce:node:list	List all nodes.
	cce:job:list	List all cluster jobs.
	cce:addonInstance:list	List all add-on instances.
	cce:addonTemplate:list	List all add-on templates.
	cce:chart:list	List all charts.
	cce:nodepool:list	List all node pools in a cluster.
	cce:release:list	List all releases.
	cce:storage:list	List all volumes.
cce:kubernetes:*	None	Perform operations on all Kubernetes resources. For details, see **Namespace Permissions**.
ecs:*:get	None	View details about all ECS resources. An ECS with multiple EVS disks is a cluster node in CCE.
ecs:*:list	None	List all ECS resources.
bms:*:get*	None	View BMS resource details.
bms:*:list	None	List all BMS resources.
ims:*:get*	None	View IMS resource details.
ims:*:list*	None	List all IMS resources.
evs:*:get	None	View EVS resource details. EVS disks can be attached to cloud servers and expanded to a higher capacity whenever needed.
evs:*:list	None	List all EVS resources.
evs:*:count	None	None
vpc:*:get	None	View VPC resource details. A cluster must run in a VPC. When creating a namespace, create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.

Action	Specific Action	Description
vpc:*:list	None	List all VPC resources.
elb:*:get	None	View ELB resource details.
elb:*:list	None	List all ELB resources.
nat:*:get	None	View NAT Gateway resource details.
nat:*:list	None	List all NAT Gateway resources.
sfs:*:get*	None	View SFS resource details.
sfs:shares:ShareAction	None	Share SFS resources for scaling.
sfsturbo:*:get*	None	View SFS Turbo resource details.
sfsturbo:shares:ShareAction	None	Share SFS Turbo resources for scaling.
tms:resourceTags:list	None	List TMS resources.
kps:domainKeypairs:list	None	List DEW SSH keys.
kps:domainKeypairs:get	None	View DEW SSH keys.
kms:cmk:get	None	View DEW keys.
kms:cmk:list	None	List DEW keys.
aom:*:get	None	View AOM resource details.
aom:*:list	None	List all AOM resources.
aom:autoScalingRule:*	None	Perform all operations on AOM auto scaling rules.
lts:*:get	None	View details about all LTS resources.
lts:*:list	None	List all LTS resources.
smn:*:get	None	View SMN resource details.
smn:*:list	None	List SMN resources.

Custom Policies

Custom policies can be created as a supplement to the system-defined policies of CCE. For details about actions supported in custom policies, see **Permissions and Supported Actions**.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a JSON policy or edit an existing one.

For details, see **Creating a Custom Policy**. The following lists examples of common CCE custom policies.

Examples

- Example 1: Creating a cluster named **test**

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cce:cluster:create"
            ]
        }
    ]
}
```

- Example 2: Denying node deletion

  A policy with only "Deny" permissions must be used with other policies. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

  If you want to grant the **CCEFullAccess** permission to a user but prevent them from deleting nodes (**cce:node:delete**), you can create a custom policy that denies node deletion. Then, attach this policy with the **CCEFullAccess** policy to the user. Since an explicit denial in any policy takes precedence over any allowances, the user will have permission to perform all operations on nodes except for deleting them. The following is an example of a deny policy:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "cce:node:delete"
            ]
        }
    ]
}
```

- Example 3: Creating a custom policy containing multiple actions

  A custom policy can contain the actions of multiple services that are of the global or project-level type. The following is an example policy containing multiple actions:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ecs:cloudServers:resize",
                "ecs:cloudServers:delete",
                "ecs:cloudServers:delete",
                "ims:images:list",
                "ims:serverImages:create"
            ],
            "Effect": "Allow"
        }
    ]
}
```


CCE Cluster Permissions and Enterprise Projects

CCE supports resource management and permission allocation by cluster and enterprise project.

Note that:

- IAM projects are based on physical isolation of resources, whereas enterprise projects provide global logical groups of resources, which better meet the actual requirements of enterprises. In addition, IAM policies can be managed based on enterprise projects. Therefore, use enterprise projects for permissions management. For details, see **Creating an Enterprise Project**.
- When there are both IAM projects and enterprise projects, IAM preferentially matches the IAM project policies.
- When creating a cluster or node using purchased cloud resources, ensure that IAM users have been granted the required permissions in the enterprise project to use these resources. Otherwise, the cluster or node may fail to be created.
- If a resource does not support enterprise projects, the permissions granted to the resource will not take effect.

Resource Type	Resource	Description
Supporting enterprise projects	cluster	Cluster
	node	Node
	nodepool	Node pool
	job	Job
	tag	Cluster label
	addonInstance	Add-on instance
	release	Helm release
	storage	Storage
Not supporting enterprise projects	quota	Cluster quota
	chart	Chart
	addonTemplate	Add-on template

CCE Cluster Permissions and IAM RBAC

CCE is compatible with IAM system roles for permissions management. Use fine-grained policies provided by IAM to simplify permissions management.

CCE supports the following roles:

- Basic IAM roles:
  - te_admin (Tenant Administrator): Users with this role can call all APIs of all services except IAM.

  - readonly (Tenant Guest): Users with this role can call APIs with the read-only permissions of all services except IAM.
- Custom CCE administrator role: CCE Administrator

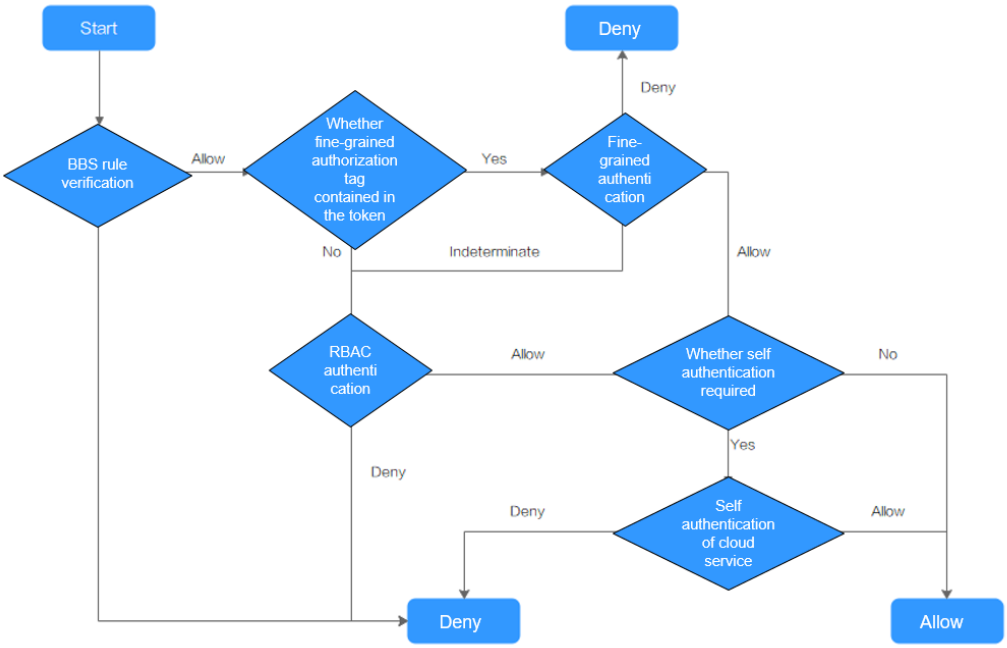
NOTE

If a user has the Tenant Administrator or CCE Administrator system role, the user has the cluster-admin permissions in Kubernetes RBAC and the permissions cannot be removed after the cluster is created.

If the user is the cluster creator, the cluster-admin permissions in Kubernetes RBAC are granted to the user by default. The permissions can be manually removed after the cluster is created.

- Method 1: Choose **Permissions** > **Namespace-Level Permissions** > **Delete** in the same role as cluster-creator on the CCE console.
- Method 2: Delete **ClusterRoleBinding: cluster-creator** through the API or kubectl.

When RBAC and IAM policies co-exist, the backend authentication logic for open APIs or console operations on CCE is as follows.



16.3 Namespace Permissions (Kubernetes RBAC-based)

Namespace Permissions (Kubernetes RBAC-based)

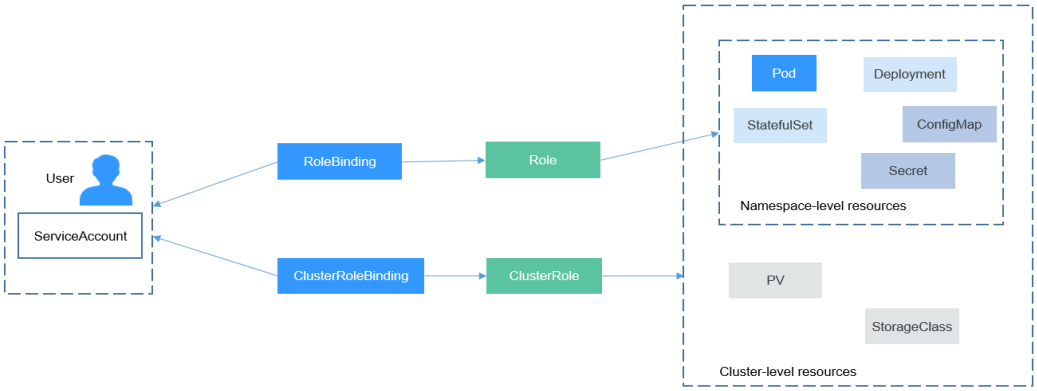
You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- Role: defines a set of rules for accessing Kubernetes resources in a namespace.
- RoleBinding: defines the relationship between users and roles.
- ClusterRole: defines a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).

- ClusterRoleBinding: defines the relationship between users and cluster roles.

Role and ClusterRole specify actions that can be performed on specific resources. RoleBinding and ClusterRoleBinding bind roles to specific users, user groups, or ServiceAccounts. Illustration:

Figure 16-3 Role binding



On the CCE console, you can assign permissions to a user or user group to access resources in one or multiple namespaces. By default, the CCE console provides the following ClusterRoles:

- view (read-only): read-only permission on most resources in all or selected namespaces.
- edit (development): read and write permissions on most resources in all or selected namespaces. If this ClusterRole is configured for all namespaces, its capability is the same as the O&M permission.
- admin (O&M): read and write permissions on most resources in all namespaces, and read-only permission on nodes, storage volumes, namespaces, and quota management.
- cluster-admin (administrator): read and write permissions on all resources in all namespaces.
- drainage-editor: drain a node.
- drainage-viewer: view the nodal drainage status but cannot drain a node.
- geip-editor-role: use of global EIPs in the cluster, but only read and write permissions on GEIPs.
- icagent-clusterRole: report Kubernetes event logs to LTS.

In addition to the preceding typical ClusterRoles, you can define Role and RoleBinding to grant the permissions to add, delete, modify, and obtain global resources (such as nodes, PVs, and CustomResourceDefinitions) and different resources (such as pods, Deployments, and Services) in namespaces for refined permission control.

Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)

Users with different cluster permissions (assigned using IAM) have different namespace permissions (assigned using Kubernetes RBAC). **Table 16-13** lists the namespace permissions of different users.

Table 16-13 Differences in namespace permissions

User	Clusters of v1.13 and Later
User with the Tenant Administrator permissions (for example, an account)	All namespace permissions
IAM user with the CCE Administrator role	All namespace permissions
IAM user with the CCE FullAccess or CCE ReadOnlyAccess role	Requires Kubernetes RBAC authorization.
IAM user with the Tenant Guest role	Requires Kubernetes RBAC authorization.

Precautions

- After you create a cluster, CCE automatically assigns the cluster-admin permission to you, which means you have full control on all resources in all namespaces in the cluster. The ID of a federated user changes upon each login and logout. Therefore, the user with the permissions is displayed as deleted. In this case, do not delete the permissions. Otherwise, the authentication fails. You are advised to grant the cluster-admin permission to a user group on CCE and add federated users to the user group.
- A user with the Security Administrator role has all IAM permissions except role switching. For example, an account in the admin user group has this role by default. Only these users can assign permissions on the **Permissions** page on the CCE console.

Configuring Namespace Permissions (on the Console)

You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

Step 1 Log in to the CCE console. In the navigation pane, choose **Permissions**.

Step 2 Select a cluster for which you want to add permissions from the drop-down list on the right.

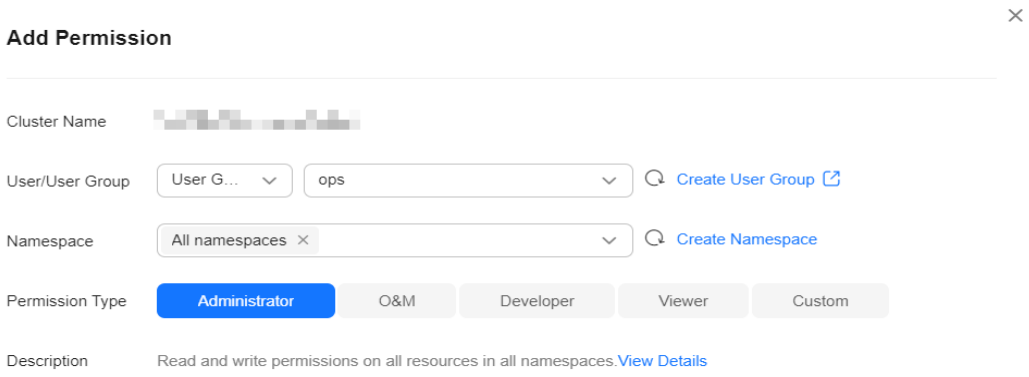
Step 3 Click **Add Permission** in the upper right corner.

Step 4 Confirm the cluster name and select the namespace to assign permissions for. For example, select **All namespaces**, the target user or user group, and select the permissions.

NOTE

If you do not have IAM permissions, you cannot select users or user groups when configuring permissions for other users or user groups. In this case, you can enter a user ID or user group ID.

Figure 16-4 Configuring namespace permissions

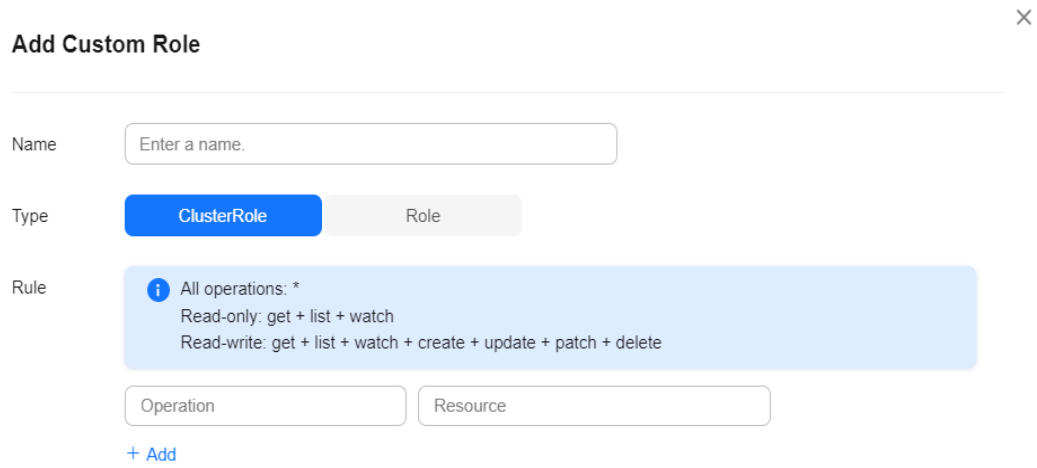


Permissions can be customized as required. After selecting **Custom** for **Permission Type**, click **Add Custom Role** on the right of the **Custom** parameter. In the dialog box displayed, enter a name and select a rule. After the custom rule is created, you can select a value from the **Custom** drop-down list box.

Custom permissions are classified into ClusterRole and Role. Each ClusterRole or Role contains a group of rules that represent related permissions. For details, see **Using RBAC Authorization**.

- A ClusterRole is a cluster-level resource that can be used to configure cluster access permissions.
- A Role is used to configure access permissions in a namespace. When creating a Role, specify the namespace to which the Role belongs.

Figure 16-5 Custom permission



Step 5 Click **OK**.

----End

Using kubectl to Configure Namespace Permissions

NOTE

When you access a cluster using kubectl, CCE uses **kubeconfig.json** generated on the cluster for authentication. This file contains user information, based on which CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a kubeconfig.json file vary from user to user. The permissions that a user has are listed in **Cluster Permissions (IAM-based) and Namespace Permissions (Kubernetes RBAC-based)**.

In addition to cluster-admin, admin, edit, and view, you can define Roles and RoleBindings to configure the permissions to add, delete, modify, and obtain resources, such as pods, Deployments, and Services, in the namespace.

The procedure for creating a Role is very simple. To be specific, specify a namespace and then define rules. The rules in the following example are to allow GET and LIST operations on pods in the default namespace.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default                  # Namespace
  name: role-example
rules:
- apiGroups: [""]
  resources: ["pods"]                 # The pod can be accessed.
  verbs: ["get", "list"]              # The GET and LIST operations can be performed.
```

- **apiGroups** indicates the API group to which the resource belongs.
- **resources** indicates the resources that can be operated. Pods, Deployments, ConfigMaps, and other Kubernetes resources are supported.
- **verbs** indicates the operations that can be performed. **get** indicates querying a specific object, and **list** indicates listing all objects of a certain type. Other value options include **create**, **update**, and **delete**.

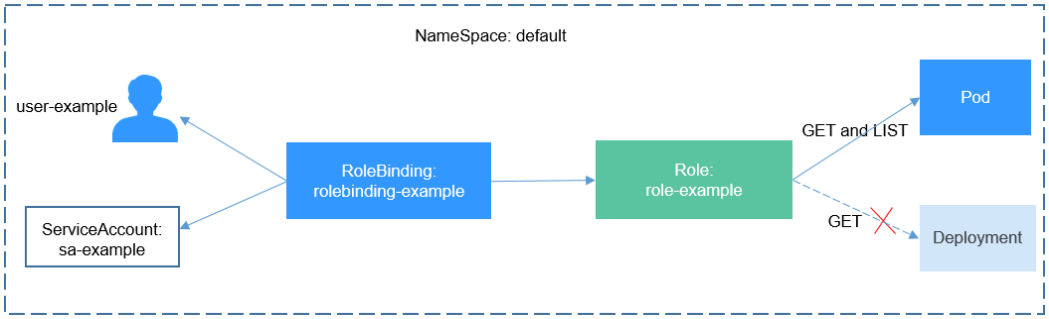
For details, see **Using RBAC Authorization**.

After creating a Role, you can bind the Role to a specific user, which is called RoleBinding. The following shows an example:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: RoleBinding-example
  namespace: default
  annotations:
    CCE.com/IAM: 'true'
roleRef:
  kind: Role
  name: role-example
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: 0c97ac3cb280f4d91fa7c0096739e1f8 # User ID of the user-example
  apiGroup: rbac.authorization.k8s.io
```

The **subjects** section binds a Role with an IAM user so that the IAM user can obtain the permissions defined in the Role, as shown in the following figure.

Figure 16-6 Binding a role to a user



You can also specify a user group in the **subjects** section. In this case, all users in the user group obtain the permissions defined in the Role.

```
subjects:
- kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7    # User group ID
  apiGroup: rbac.authorization.k8s.io
```

Use the IAM user user-example to connect to the cluster and obtain the pod information. The following is an example of the returned pod information.

```
# kubectl get pod
NAME                                  READY   STATUS    RESTARTS   AGE
deployment-389584-2-6f6bd4c574-2n9rk  1/1     Running   0          4d7h
deployment-389584-2-6f6bd4c574-7s5qw  1/1     Running   0          4d7h
deployment-3895841-746b97b455-86g77   1/1     Running   0          4d7h
deployment-3895841-746b97b455-twvpn   1/1     Running   0          4d7h
nginx-658dff48ff-7rkph                1/1     Running   0          4d9h
nginx-658dff48ff-njdhj                1/1     Running   0          4d9h
# kubectl get pod nginx-658dff48ff-7rkph
NAME                     READY   STATUS    RESTARTS   AGE
nginx-658dff48ff-7rkph   1/1     Running   0          4d9h
```

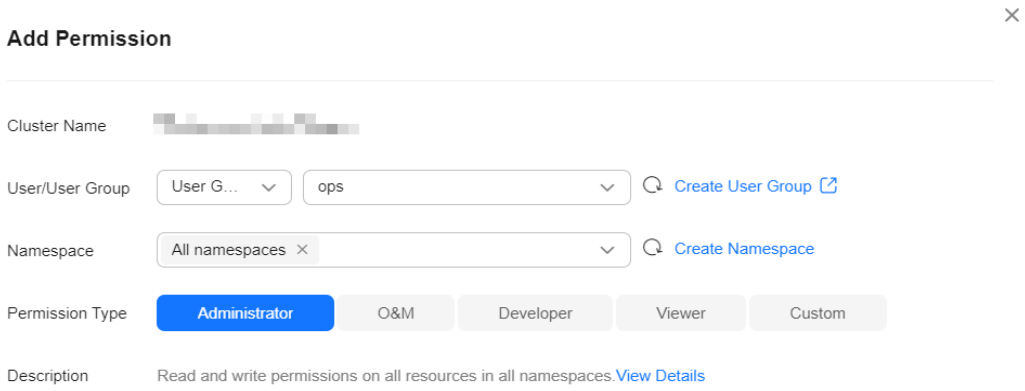
Try querying Deployments and Services in the namespace. The output shows that **user-example** does not have the required permissions. Try querying the pods in namespace kube-system. The output shows that **user-example** does not have the required permissions. This indicates that the IAM user **user-example** has only the GET and LIST Pod permissions in the **default** namespace, which is the same as expected.

```
# kubectl get deploy
Error from server (Forbidden): deployments.apps is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "deployments" in API group "apps" in the namespace "default"
# kubectl get svc
Error from server (Forbidden): services is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "services" in API group "" in the namespace "default"
# kubectl get pod --namespace=kube-system
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list resource "pods" in API group "" in the namespace "kube-system"
```

Example: Assigning Cluster Administrator Permissions (cluster-admin)

You can use the cluster-admin role to assign all permissions on a cluster. This role contains the permissions for all cluster resources.

Figure 16-7 Assigning cluster administrator permissions (cluster-admin)



In the following example kubectl output, a ClusterRoleBinding has been created and the cluster-admin role is bound to the user group **cce-role-group**.

```
# kubectl get clusterrolebinding
NAME                                                ROLE                        AGE
clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/cluster-admin      61s

# kubectl get clusterrolebinding clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-23T09:15:22Z"
  name: clusterrole_cluster-admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36659058"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/clusterrole_cluster-
admin_group0c96fad22880f32a3f84c009862af6f7
  uid: d6cd43e9-b4ca-4b56-bc52-e36346fc1320
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

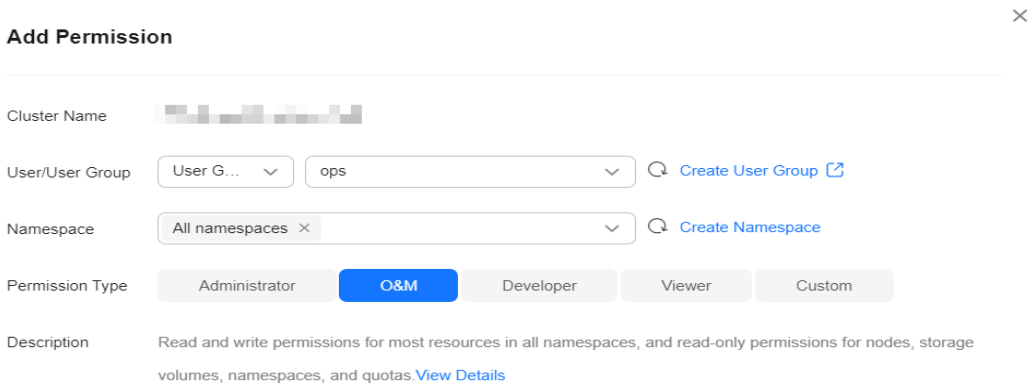
Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried, the permission configuration takes effect.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME              PROVISIONER                  RECLAIMPOLICY    VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION    AGE
csi-disk          everest-csi-provisioner      Delete           Immediate              true                   75d
csi-disk-topology   everest-csi-provisioner      Delete           WaitForFirstConsumer   true                   75d
csi-nas           everest-csi-provisioner      Delete           Immediate              true                   75d
csi-obs           everest-csi-provisioner      Delete           Immediate              false                  75d
csi-sfsturbo      everest-csi-provisioner      Delete           Immediate              true                   75d
```

Example: Assigning Namespace O&M Permissions (admin)

The admin role has the read and write permissions on most namespace resources. You can grant the admin permission on all namespaces to a user or user group.

Figure 16-8 Assigning O&M permissions on all namespaces (admin)



In the following example kubectl output, a RoleBinding has been created and the admin role is bound to the user group **cce-role-group**.

```
# kubectl get rolebinding
NAME                                                ROLE               AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/admin   18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

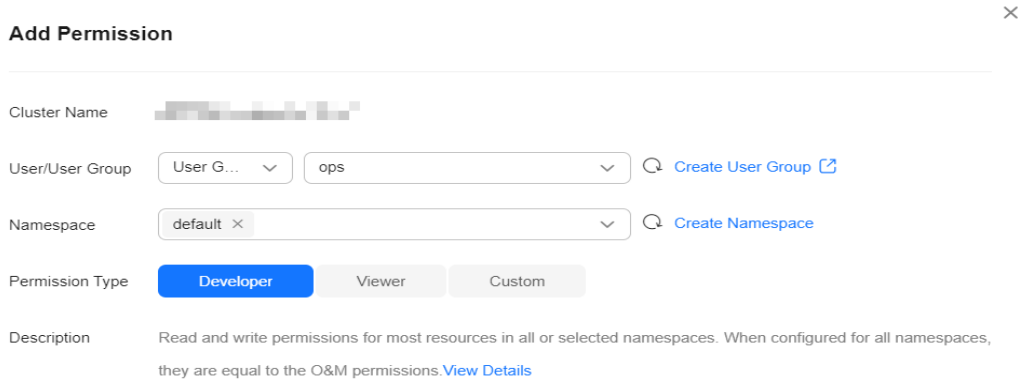
Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried but a namespace cannot be created, the permission configuration takes effect.

```
# kubectl get pv
No resources found
# kubectl get sc
NAME                PROVISIONER                     RECLAIMPOLICY    VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION    AGE
csi-disk            everest-csi-provisioner         Delete           Immediate              true                   75d
csi-disk-topology   everest-csi-provisioner         Delete           WaitForFirstConsumer   true                   75d
csi-nas             everest-csi-provisioner         Delete           Immediate              true                   75d
csi-obs             everest-csi-provisioner         Delete           Immediate              false                  75d
csi-sfsturbo        everest-csi-provisioner         Delete           Immediate              true                   75d
# kubectl apply -f namespaces.yaml
Error from server (Forbidden): namespaces is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot
create resource "namespaces" in API group "" at the cluster scope
```

Example: Assigning Namespace Developer Permissions (edit)

The edit role has the read and write permissions on most namespace resources. You can grant the edit permission on all namespaces to a user or user group.

Figure 16-9 Assigning developer permissions on the default namespace (edit)



In the following example kubectl output, a RoleBinding has been created, the edit role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME                                                ROLE               AGE
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/admin   18s
# kubectl get rolebinding clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:30:08Z"
  name: clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36963685"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_admin_group0c96fad22880f32a3f84c009862af6f7
  uid: 6c6f46a6-8584-47da-83f5-9eef1f7b75d6
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

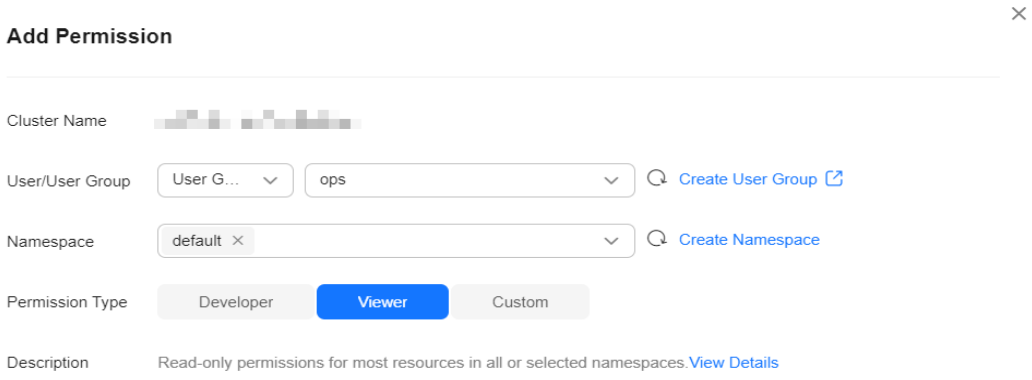
Connect to the cluster as an authorized user. In this example, you can create and obtain resources in the default namespace, but cannot query resources in the kube-system namespace or cluster resources.

```
# kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
test-568d96f4f8-brdrp   1/1     Running   0          33m
test-568d96f4f8-cgjqp   1/1     Running   0          33m
# kubectl get pod -nkube-system
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot list
resource "pods" in API group "" in the namespace "kube-system"
# kubectl get pv
Error from server (Forbidden): persistentvolumes is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8"
cannot list resource "persistentvolumes" in API group "" at the cluster scope
```

Example: Assigning Read-Only Namespace Permissions (view)

The view role has the read-only permissions on a namespace. You can assign permissions to users to view one or multiple namespaces.

Figure 16-10 Assigning read-only namespace permissions (view)



In the following example kubectl output, a RoleBinding has been created, the view role is bound to the user group **cce-role-group**, and the target namespace is the default namespace.

```
# kubectl get rolebinding
NAME                                                ROLE               AGE
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7   ClusterRole/view   7s

# kubectl get rolebinding clusterrole_view_group0c96fad22880f32a3f84c009862af6f7 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    CCE.com/IAM: "true"
  creationTimestamp: "2021-06-24T01:36:53Z"
  name: clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  namespace: default
  resourceVersion: "36965800"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/default/rolebindings/
clusterrole_view_group0c96fad22880f32a3f84c009862af6f7
  uid: b86e2507-e735-494c-be55-c41a0c4ef0dd
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: 0c96fad22880f32a3f84c009862af6f7
```

Connect to the cluster as an authorized user. In this example, you can query resources in the default namespace but cannot create resources.

```
# kubectl get pod
NAME                    READY   STATUS    RESTARTS   AGE
test-568d96f4f8-brdrp   1/1     Running   0          40m
test-568d96f4f8-cgjqp   1/1     Running   0          40m
# kubectl run -i --tty --image tutum/dnsutils dnsutils --restart=Never --rm /bin/sh
Error from server (Forbidden): pods is forbidden: User "0c97ac3cb280f4d91fa7c0096739e1f8" cannot create
resource "pods" in API group "" in the namespace "default"
```

Example: Assigning Permissions for a Specific Kubernetes Resource Object

You can assign permissions on a specific Kubernetes resource object, such as pod, Deployment, and Service. For details, see **Using kubectl to Configure Namespace Permissions**.

16.4 Example: Designing and Configuring Permissions for Users in a Department

Overview

The conventional distributed task scheduling mode is being replaced by Kubernetes. CCE allows you to easily deploy, manage, and scale containerized applications in the cloud by providing support for you to use Kubernetes.

To help enterprise administrators manage resource permissions in clusters, CCE provides multi-dimensional, fine-grained permission policies and management measures. CCE permissions are described as follows:

- **Cluster-level permissions**: allowing a user group to perform operations on clusters, nodes, charts, and add-ons. These permissions are assigned based on IAM system policies.
- **Namespace-level permissions**: allowing a user or user group to perform operations on Kubernetes resources, such as workloads, Services, storage, and namespaces. These permissions are assigned based on Kubernetes RBAC.

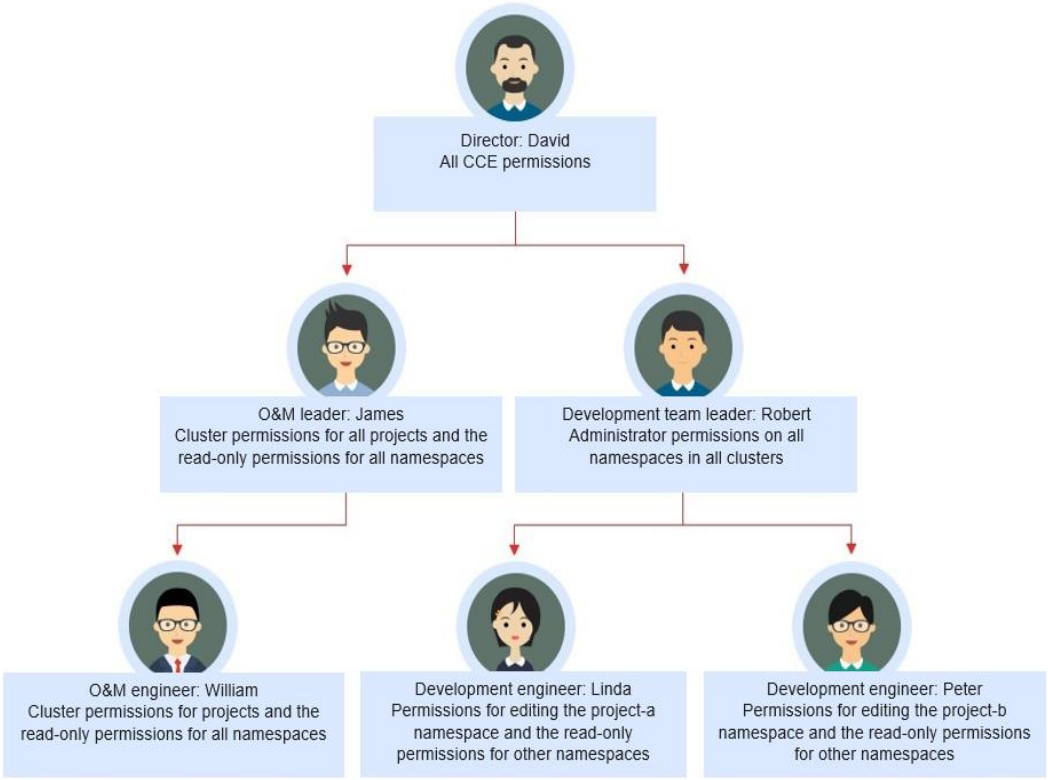
Cluster permissions and namespace permissions are independent of each other but must be used together. The permissions set for a user group apply to all users in the user group. When multiple permissions are added to a user or user group, they take effect at the same time (the union set is used).

Permission Design

The following uses company X as an example.

Generally, a company has multiple departments or projects, and each department has multiple members. Design how permissions are to be assigned to different groups and projects, and set a user name for each member to facilitate subsequent user group and permissions configuration.

The following figure shows the organizational structure of a department in a company and the permissions to be assigned to each member:



Director: David

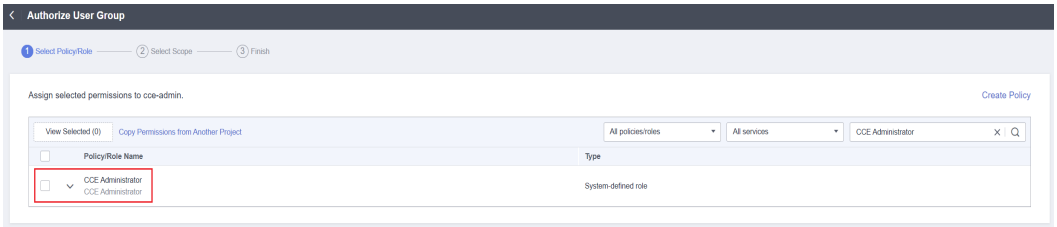
David is a department director of company X. To assign him all CCE permissions (both cluster and namespace permissions), create the **cce-admin** user group for David on the IAM console and assign the CCE Administrator role.

NOTE

CCE Administrator: This role has all CCE permissions. You do not need to assign other permissions.

CCE FullAccess and CCE ReadOnlyAccess: These policies are related to cluster management permissions and configured only for cluster-related resources (such as clusters and nodes). You must also configure namespace permissions to perform operations on Kubernetes resources (such as workloads and Services).

Figure 16-11 Assigning permissions to the user group to which David belongs

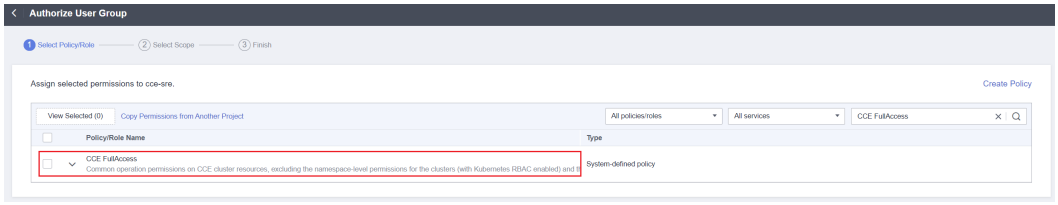


O&M Leader: James

James is the O&M team leader of the department. He needs the cluster permissions for all projects and the read-only permissions for all namespaces.

To assign the permissions, create a user group named **cce-sre** on the IAM console and add James to this user group. Then, assign CCE FullAccess to the user group **cce-sre** to allow it to perform operations on clusters in all projects.

Figure 16-12 Assigning permissions to the user group to which James belongs



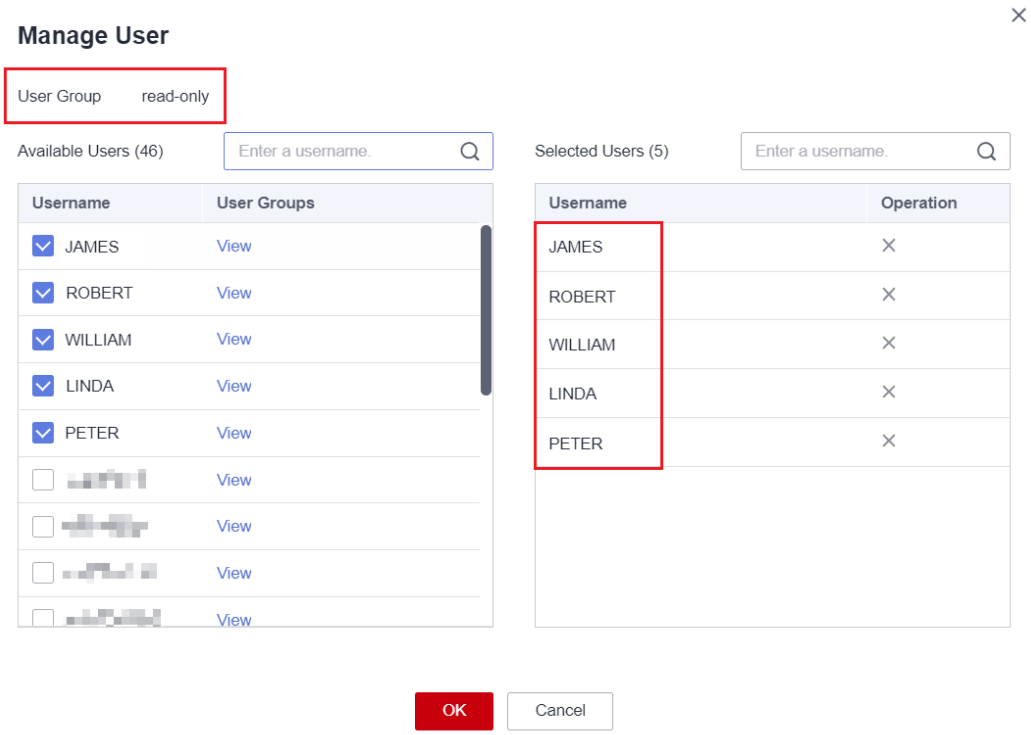
Assigning Read-only Permissions on All Clusters and Namespaces to All Team Leaders and Engineers

You can create a read-only user group named **read_only** on the IAM console and add users to the user group.

- Although the development engineers Linda and Peter do not require cluster management permissions, they still need to view data on the CCE console. Therefore, the read-only cluster permission is required.
- For the O&M engineer William, assign the read-only permission on clusters to him in this step.
- The O&M team leader James already has the management permissions on all clusters. You can add him to the **read_only** user group to assign the read-only permission on clusters to him.

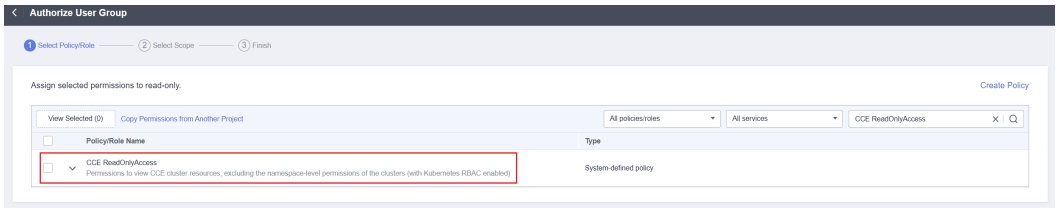
Users James, Robert, William, Linda, and Peter are added to the **read_only** user group.

Figure 16-13 Adding users to the read_only user group



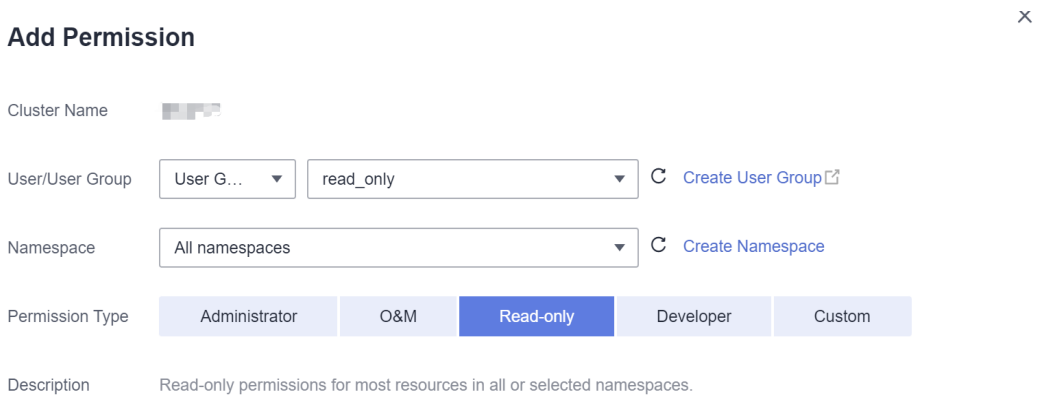
Assign the read-only permission on clusters to the user group **read_only**.

Figure 16-14 Assigning the read-only permission on clusters to the user group



Return to the CCE console, and add the read-only permission on namespaces to the user group **read_only** to which the five users belong. Choose **Permissions** on the CCE console, and assign the read-only policy to the user group **read_only** for each cluster.

Figure 16-15 Assigning the read-only permission on namespaces to the user group



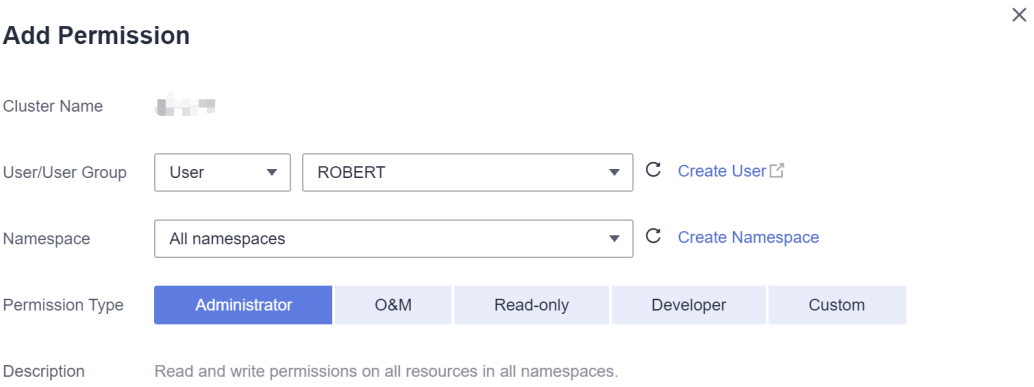
After the setting is complete, James has the cluster management permissions for all projects and the read-only permissions on all namespaces, and the Robert, William, Linda, and Peter have the read-only permission on all clusters and namespaces.

Development Team Leader: Robert

In the previous steps, Robert has been assigned the read-only permission on all clusters and namespaces. Now, assign the administrator permissions on all namespaces to Robert.

Therefore, assign the administrator permissions on all namespaces in all clusters to Robert.

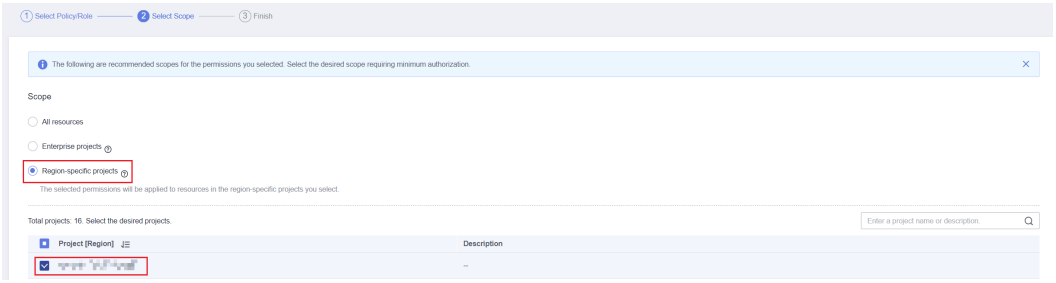
Figure 16-16 Assigning the administrator permissions on namespaces to Robert



O&M Engineer: William

In the previous steps, William has been assigned the read-only permission on all clusters and namespaces. He also requires the cluster management permissions in his region. Therefore, you can log in to the IAM console, create a user group named **cce-sre-b4** and assign CCE FullAccess to William for his region.

Figure 16-17 Assigning the cluster management permissions for Beijing4 region to the user group to which WILLIAM belongs

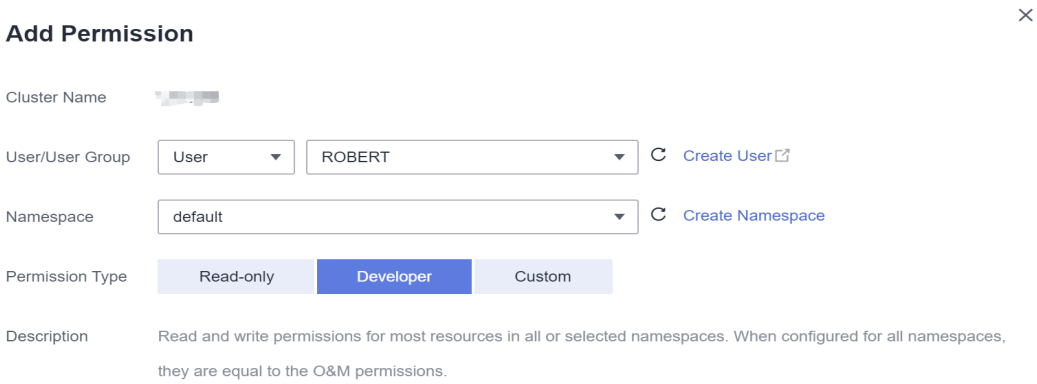


Now, William has the cluster management permissions for his region and the read-only permission on all namespaces.

Development Engineers: Linda and Peter

In the previous steps, Linda and Peter have been assigned the read-only permission on clusters and namespaces. Therefore, you only need to assign the edit policy to them.

Figure 16-18 Assigning the edit policy on namespaces



By now, all the required permissions are assigned to the department members.

16.5 Permission Dependency of the CCE Console

Some CCE permissions policies depend on the policies of other cloud services. To view or use other cloud resources on the CCE console, enable the access control feature of IAM and assign dependency policies for the other cloud services.

- Dependency policies are assigned based on the CCE FullAccess or CCE ReadOnlyAccess policy you configure. For details, see **Granting Cluster Permissions to an IAM User**.
- Only users and user groups with namespace permissions can gain the view access to resources in clusters.
  - If a user is granted the view access to all namespaces of a cluster, the user can view all namespace resources (except secrets) in the cluster. To view secrets in the cluster, the user must gain the **admin** or **edit** role in all namespaces of the cluster.

- The **view** role within a single namespace allows users to view resources only in the specified namespace.

Dependency Policy Configuration

To grant an IAM user the permissions to view or use resources of other cloud services on the CCE console, you must first grant the CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess policy to the user group to which the user belongs and then grant the dependency policies listed in **Table 16-14** to the user. These dependency policies will allow the IAM user to access resources of other cloud services.

NOTE

Enterprise projects can group and manage resources across different projects of an enterprise. Resources are thereby isolated. IAM allows you to implement fine-grained authorization. It is strongly recommended that you use IAM for permissions management.

If you use an enterprise project to set permissions for IAM users, the following restrictions apply:

- On the CCE console, enterprise projects cannot call the API used to obtain AOM monitoring data for cluster monitoring. Therefore, IAM users in these enterprise projects cannot query monitoring data.
- On the CCE console, enterprise projects cannot call the API to query the key pair created during node creation. Therefore, IAM users in these enterprise projects cannot use the key pair login mode. Only the password login mode is supported.
- On the CCE console, enterprise projects are not supported during template creation. Therefore, enterprise project sub-users cannot use template management.
- On the CCE console, the EVS disk query API does not support enterprise projects. Therefore, enterprise project IAM users cannot use existing EVS disks to create PVs. To use this function, add the fine-grained permissions such as evs:volumes:get to the IAM users.

CCE supports fine-grained permissions configuration, but has the following restrictions:

- AOM does not support resource-level monitoring. After operation permissions on specific resources are configured using IAM's fine-grained cluster resource management function, IAM users can view cluster monitoring information on the **Dashboard** page of the CCE console, but cannot view the data on non-fine-grained metrics.

Table 16-14 Dependency policies

Console Function	Dependent Service	Role or Policy Required
Cluster overview	Application Operations Management (AOM)	• An IAM user with the CCE Administrator permission assigned can use this function only after the AOM FullAccess permission is assigned. • IAM users with IAM ReadOnlyAccess, CCE FullAccess, or CCE ReadOnlyAccess assigned can directly use this function.

Console Function	Dependent Service	Role or Policy Required
Workload management	Elastic Load Balance (ELB) Application Performance Management (APM) Application Operations Management (AOM) NAT Gateway Object Storage Service (OBS) Scalable File Service (SFS)	Except in the following cases, the user does not require any additional role to create workloads. • To create a Service using ELB, you must have the ELB FullAccess or ELB Administrator plus VPC Administrator permissions assigned. • To use a Java probe, you must have the AOM FullAccess and APM FullAccess permissions assigned. • To create a Service using NAT Gateway, you must have the NAT Gateway Administrator permission assigned. • To use OBS, you must have the OBS Administrator permission globally assigned. **NOTE** Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users, enterprise projects, and user groups. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect. • To use SFS, you must have the SFS FullAccess permission assigned.
Cluster management	Application Operations Management (AOM) Billing Center (BSS)	• Auto scale-out or scale-up requires the AOM FullAccess policy. • Changing the billing mode to yearly/monthly requires the BSS Administrator role.
Node management	Elastic Cloud Server (ECS)	If the permission assigned to an IAM user is CCE Administrator, creating or deleting a node requires the ECS FullAccess or ECS Administrator policy and the VPC Administrator policy.

Console Function	Dependent Service	Role or Policy Required
Service	Elastic Load Balance (ELB) NAT Gateway	Except in the following cases, the user does not require any additional role to create a Service. • To create a Service using ELB, you must have the ELB FullAccess or ELB Administrator plus VPC Administrator permissions assigned. • To create a Service using NAT Gateway, you must have the NAT Administrator permission assigned.
Storage	Object Storage Service (OBS) Scalable File Service (SFS) SFS Turbo	• To use OBS, you must have the OBS Administrator permission globally assigned. **NOTE** Because of the cache, it takes about 13 minutes for the RBAC policy to take effect after being granted to users, enterprise projects, and user groups. After an OBS-related system policy is granted, it takes about 5 minutes for the policy to take effect. • To use SFS, you must have the SFS FullAccess permission assigned. • To use SFS Turbo, you must have the SFS Turbo FullAccess permission. The CCE Administrator role is required for importing storage devices.
Namespace management	N/A	N/A
Chart management	N/A	Cloud accounts and the IAM users with CCE Administrator assigned can use this function.
Add-ons	N/A	Cloud accounts and the IAM users with CCE Administrator, CCE FullAccess, or CCE ReadOnlyAccess assigned can use this function.

Console Function	Dependent Service	Role or Policy Required
Permissions management	N/A	• For cloud accounts, no additional policy/role is required. • IAM users with the CCE Administrator or global Security Administrator permission assigned can use this function. • IAM users with the CCE FullAccess or CCE ReadOnlyAccess permission can use this function. In addition, the IAM users must have the **administrator permissions (cluster-admin)** on the namespace.
ConfigMaps and Secrets	N/A	• Creating ConfigMaps does not require any additional policy. • Viewing secrets requires that the cluster-admin, admin, or edit permission be configured for the namespace. The DEW KeypairFullAccess or DEW KeypairReadOnlyAccess policy must be assigned for dependent services.
Help center	N/A	N/A
Switching to other related services	Software Repository for Container (SWR) Log Tank Service (LTS) Multi-Cloud Container Platform (MCP)	The CCE console provides links to other related services. To view or use these services, an IAM user must be assigned required permissions for the services.

16.6 Service Account Token Security Improvement

In clusters earlier than v1.21, a token is obtained by mounting the secret of the service account to a pod. Tokens obtained this way are permanent. This approach is no longer recommended starting from version 1.21. Service accounts will stop auto creating secrets in clusters from version 1.25.

In clusters of version 1.21 or later, you can use the **TokenRequest** API to obtain the token and use the projected volume to mount the token to the pod. Such tokens are valid for a fixed period (one hour by default). Before expiration, kubelet refreshes the token to ensure that the pod always uses a valid token. When the mounting pod is deleted, the token automatically becomes invalid. This approach is implemented by the **BoundServiceAccountTokenVolume** feature to improve

the token security of the service account. Kubernetes clusters of v1.21 and later enable this approach by default.

For smooth transition, the community extends the token validity period to one year by default. After one year, the token becomes invalid, and clients that do not support certificate reloading cannot access the API server. It is recommended that clients of earlier versions be upgraded as soon as possible. Otherwise, service faults may occur.

If you use a Kubernetes client of a to-be-outdated version, the certificate reloading may fail. Versions of officially supported Kubernetes client libraries able to reload tokens are as follows:

- Go: >= v0.15.7
- Python: >= v12.0.0
- Java: >= v9.0.0
- Javascript: >= v0.10.3
- Ruby: master branch
- Haskell: v0.3.0.0
- C#: >= 7.0.5

For details, visit **https://github.com/kubernetes/enhancements/tree/master/keps/sig-auth/1205-bound-service-account-tokens**.

NOTE

If you need a token that never expires, you can also **manually manage secrets for service accounts**. Although a permanent service account token can be manually created, you are advised to use a short-lived token by calling the **TokenRequest** API for higher security.

Diagnosis

Perform the following steps to check your CCE clusters of v1.21 or later:

1. Check the add-on versions.
   - If you are using the Prometheus add-on v2.23.34 or earlier, upgrade it to v2.23.34 or later.
   - If you are using the NPD add-on v1.15.0 or earlier, upgrade it to the latest version.
2. Use kubectl to access the cluster and run the **kubectl get --raw "/metrics" | grep stale** command to obtain the metrics. Check the metric named **serviceaccount_stale_tokens_total**.

   If the value is greater than 0, some workloads in the cluster may be using an earlier client-go version. In this case, check whether this problem occurs in your deployed applications. If yes, upgrade client-go to the version specified by the community as soon as possible. The version must be at least two major versions of the CCE cluster. For example, if your cluster version is 1.23, the Kubernetes dependency library version must be at least 1.19.

```
[root@ ]# kubectl get --raw "/metrics" | grep stale
# HELP serviceaccount_stale_tokens_total [ALPHA] Cumulative stale projected service account tokens used
# TYPE serviceaccount_stale_tokens_total counter
serviceaccount_stale_tokens_total 52
```


16.7 System Agencies

CCE works closely with multiple cloud services to support compute, storage, networking, and monitoring functions. When you log in to the CCE console for the first time, CCE automatically requests permissions to access those cloud services in the region where you run your applications. Specifically:

- Compute services

  When you create a node in a cluster, a cloud server is created accordingly. The prerequisite is that CCE has obtained the permissions for accessing Elastic Cloud Service (ECS) and Bare Metal Server (BMS).

- Storage services

  CCE allows you to mount storage to nodes and containers in a cluster. The prerequisite is that CCE has obtained the permissions for accessing services such as Elastic Volume Service (EVS), Scalable File Service (SFS), and Object Storage Service (OBS).

- Networking services

  CCE allows containers in a cluster to be published as services that can be accessed by external systems. The prerequisite is that CCE has obtained the permissions for accessing services such as Virtual Private Cloud (VPC) and Elastic Load Balance (ELB).

- Container and monitoring services

  CCE supports functions such as container image pull, monitoring, and logging. The prerequisite is that CCE has obtained the permissions for accessing services such as SoftWare Repository for Container (SWR) and Application Operations Management (AOM).

After you agree to the entrustment, CCE automatically creates an agency in IAM to delegate other resource operation permissions in your account to Huawei Cloud CCE. For details, see **Account Delegation**.

The agencies automatically created by CCE are as follows:

- **cce_admin_trust** has permission to call all cloud services that CCE depends on, with the exception of IAM.
- **cce_cluster_agency** is authorized to operate only the cloud services required by CCE to generate temporary access credentials used by components in CCE clusters.

cce_admin_trust

The cce_admin_trust agency has the Tenant Administrator permissions. Tenant Administrator has permission to call all cloud services (with the exception of IAM) that CCE depends on, such as creating a cluster or node. This delegation of permissions only applies to the current region.

To use CCE in multiple regions, request for cloud resource permissions in each region. You can go to the IAM console > **Agencies** and click **cce_admin_trust** to view the permissions of each region.

CCE may fail to run as expected if the Tenant Administrator role is not assigned. Therefore, do not delete or modify the **cce_admin_trust** agency when using CCE.

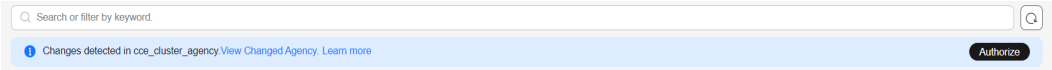
cce_cluster_agency

The **cce_cluster_agency** agency contains only the cloud service resource operation permissions required by CCE components. It generates temporary access credentials used by components in CCE clusters. This agency permission will be used when cloud service resources that CCE depends on are automatically created in a cluster, for example, when an ingress or dynamic storage volume is created.

NOTE

- The cce_cluster_agency agency supports clusters of v1.21 or later.
- When you create the cce_cluster_agency agency, a custom policy named **CCE cluster policies** is automatically created. Do not delete this policy.

If the permissions of the cce_cluster_agency agency are different from those expected by CCE, the console displays a message indicating that the permissions have changed and you need to re-authorize the agency.



The cce_cluster_agency agency may be re-authorized in the following scenarios:

- The permissions on which CCE components depend may change with versions. For example, if a new component depends on new permissions, CCE will update the expected permission list and you need to grant permissions to cce_cluster_agency again.
- When you manually modify the permissions of the cce_cluster_agency agency, the permissions of the agency are different from those expected by CCE. In this case, a message is displayed, asking you to re-authorize the agency. If you re-authorize the agency, the manually modified permissions may become invalid.

17 Settings

17.1 Dashboard

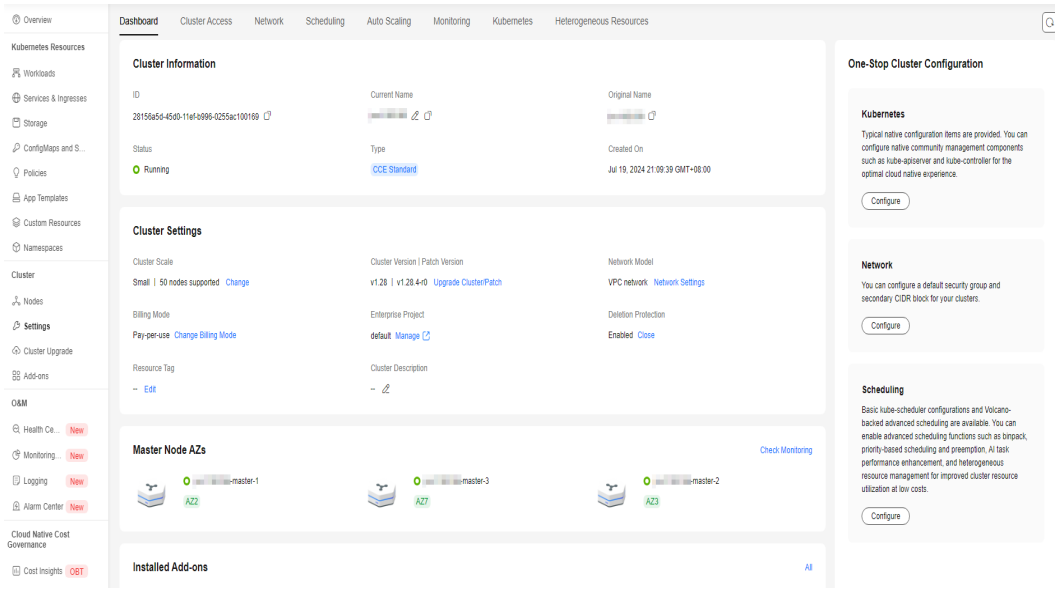
Settings offers you an entry to check and modify the basic settings of clusters. It includes information from dimension like **Cluster Information**, **Cluster Settings**, **Master Node AZs**, and **Installed Add-ons**.

Accessing the Settings

Step 1 Log in to the CCE console and click the name of the target cluster to access the cluster console.

Step 2 In the navigation pane, choose **Settings** and click the **Dashboard** tab.

Figure 17-1 Checking the cluster dashboard



----End

Cluster Information

It includes:

- **ID**: uniquely identifies a cluster resource. It is automatically generated after a cluster is created and can be used in scenarios such as API calling.
- **Current Name**: After a cluster is created, you can click  to change its name.
- **Original Name**: specifies a cluster's original name after its current name is changed. The current name of a cluster must be unique.
- **Status**: specifies the status of a cluster. For details, see **Cluster Lifecycle**.
- **Type**: specifies whether a cluster is a CCE standard or a CCE Turbo cluster. For details about the differences between CCE standard and CCE Turbo clusters, see Comparison Between Cluster Types.
- **Created On**: specifies the time when a cluster was created. You will be billed based on the creation time of clusters.

Cluster Settings

After a cluster is created, you can modify the following items for it:

- **Cluster Scale**: specifies how many nodes a cluster can manage at most. You can change the cluster scale as needed. For details, see **Changing Cluster Scale**.
- **Cluster Version | Patch Version**: specifies the Kubernetes version and CCE patch version of a cluster.
- **Network Model**: specifies the network model of a cluster, which cannot be changed. For details about network models, see **Container Network**.
- **Billing Mode**: specifies the billing mode of a cluster. A pay-per-use cluster can be changed to a yearly/monthly one. For details, see **Changing the Billing Mode of a Cluster from Pay-per-Use to Yearly/Monthly**.
- **Enterprise Project**: specifies the enterprise project to which a cluster belongs. For more details, see **Enterprise Management**.
- **Operation Protection**: After this function is enabled, you need to confirm operations by using the virtual MFA device, SMS messages, or emails. You can go to the IAM console, choose **Security Settings** > **Critical Operations**, and enable the protection functions.
- **Resource Tag**: You can add resource tags to classify resources.
- **Cluster Description**: specifies the description that you entered for a cluster. A maximum of 200 characters are allowed.
- **Cluster Deletion Protection**: A measure taken to prevent accidental deletion of clusters through the console or APIs. After this function is enabled, you will not be able to delete or unsubscribe from clusters.

Master Node AZs

You can check how many master nodes are supported in a cluster. To check data such as the resource usage of the master nodes, click **View Monitoring** in the upper right corner to go to the **Monitoring** (**Monitoring Center**) page.

Installed Add-ons

You can check the installed add-ons in a cluster. If there is an add-on to be upgraded in the cluster, click **Go to Upgrade** to go to the **Add-ons** page and see more details.

17.2 Cluster Access

Access Mode

- **kubectl**: You need to download and configure the kubectl and kubeconfig configuration files first, and then use kubectl to access a Kubernetes cluster. For details, see **Connecting to a Cluster Using kubectl**.
- **EIP**: You can bind an EIP to the cluster API server. Then, the cluster API server can access the public network.

  NOTE

  - Binding an EIP to a cluster could bring the cluster potential access risks from the public network. You are advised to harden the inbound rule of port 5443 for the master node security group. For details, see **How Can I Configure a Security Group Rule in a Cluster?**
  - This operation will restart kube-apiserver and update the cluster access certificate (kubeconfig). Do not perform any operations on the cluster during this time.
- **Custom SAN**: The Subject Alternative Name (SAN) allows multiple values (including IP addresses and domain names) to be associated with certificates. After the custom SAN is configured in the cluster access certificate, the cluster can be accessed using the domain names or IP addresses specified by the SAN. For details, see **Accessing a Cluster Using a Custom Domain Name**.

  NOTE

  This operation will restart kube-apiserver and update the cluster access certificate (kubeconfig). Do not perform any operations on the cluster during this time.

Authentication

CCE allows you to download the X509 certificate, which contains the **client.key**, **client.crt**, and **ca.crt** files. Keep your certificate secure.

For details about how to use a certificate to access clusters, see **Accessing a Cluster Using an X.509 Certificate**.

CCE supports X.509 certificate revocation. For details about how to revoke a cluster credential, see **Revoking a Cluster Access Credential**.

Server Request Settings

Table 17-1 Parameters

Item	Parameter	Description	Value
Maximum Number of Concurrent Modification API Calls	max-mutating-requests-inflight	Maximum number of concurrent modification API calls. Any requests that exceeding the specified value will be rejected by the server. The value **0** specifies that there is no limitation on the maximum number of concurrent modification calls. This parameter is related to the cluster scale. You are advised not to change the value.	Manual configuration is no longer supported since clusters of v1.21. The value is automatically specified based on the cluster scale: • **200** for clusters with 50 or 200 worker nodes • **500** for clusters with 1000 worker nodes • **1000** for clusters with 2000 worker nodes

Item	Parameter	Description	Value
Maximum Number of Concurrent Non-Modification API Calls	max-requests-inflight	Maximum number of concurrent non-modification API calls. Any requests that exceeding the specified value will be rejected by the server. The value **0** specifies that there is no limitation on the maximum number of concurrent non-modification calls. This parameter is related to the cluster scale. You are advised not to change the value.	Manual configuration is no longer supported since clusters of v1.21. The value is automatically specified based on the cluster scale: • **400** for clusters with 50 or 200 worker nodes • **1000** for clusters with 1000 worker nodes • **2000** for clusters with 2000 worker nodes
Request Timeout	request-timeout	Default request timeout interval of kube-apiserver. Exercise caution when changing the value of this parameter. Ensure that the changed value is proper to prevent frequent API timeout or other errors. This parameter is available only in clusters of v1.19.16-r30, v1.21.10-r10, v1.23.8-r10, v1.25.3-r10, and later versions.	Default: 1m0s Options: Min ≥ 1s Max ≤ 1 hour

Item	Parameter	Description	Value
Overload Control	support-overload	Cluster overload control. If enabled, concurrent requests will be dynamically controlled based on the resource demands received by master nodes to ensure the stable running of the master nodes and the cluster. For details, see **Enabling Overload Control for a Cluster**. This parameter is available only in clusters of v1.23 or later. **NOTE** In particular scenarios, such as request burst over a short period of time, a cluster could still be overloaded even though overload control is enabled for it. In such cases, you are advised to manage and control access to the cluster in a timely manner.	None

17.3 Network

You can configure a default security group and secondary CIDR block for your clusters.

Cluster Network

Table 17-2 Parameters

Parameter	Description
VPC	VPC where a cluster resides VPC enables you to provision logically isolated, configurable, manageable virtual networks for cloud servers, cloud containers, and cloud databases. The VPC gives you complete control over your virtual network, allowing you to select your own IP address range, create subnets, configure security groups, and even assign EIPs and allocate bandwidth in your network, enabling secure and easy access to your business system.

Parameter	Description
VPC CIDR Block	VPC CIDR Block of a cluster For details about how to expand the VPC CIDR block, see **Adding a Secondary VPC CIDR Block for a Cluster**.
Default Node Subnet	Node subnet of a cluster A subnet is a network that manages ECS network planes. It supports IP address management and DNS. ECSs in a subnet are assigned with its IP addresses. By default, ECSs in all subnets of the same VPC can communicate with one another, but ECSs in different VPCs cannot. You can create a VPC peering connection to enable ECSs in different VPCs to communicate with each other.
Default Node Subnet | IPv4 CIDR Block/IPv6 CIDR Block	Node subnet CIDR block of a cluster
IPv4/IPv6 Dual Stack	Whether to enable IPv6 dual stack.
Network Model	Network model of a cluster After a cluster is created, its network model cannot be changed. For details about the comparison between different network models, see **Overview**.
Default Node Security Group	Default security group of the worker nodes in a cluster You can select a custom security group as the default node security group for a cluster, and you need to allow traffic from specified ports in the security group to ensure normal communications in the cluster. If the custom security group needs some modifications, the modified security group applies only to newly created or accepted nodes. For existing nodes, you need to manually modify the security group rules.

Parameter	Description
Non-Translated CIDR Blocks for External Communication (available only in clusters using the VPC network model)	In a cluster using a VPC network, 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 are regarded as private CIDR blocks of the cluster by default. If the VPC to which the cluster resides uses a secondary CIDR block, operations such as creating or resetting a node will also add the secondary CIDR block to the private CIDR blocks. If a pod tries to access a private CIDR block, the source node will not perform NAT on the pod IP address. Instead, the upper-layer VPC can directly send the pod data packet to the destination, which means, the pod IP address is directly used to communicate with the private CIDR block in the cluster. This function is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions. For details, see **Retaining the Original IP Address of a Pod**. **NOTE** To enable a node to access a pod in another node, the node CIDR block must be added to this parameter. Similarly, to enable an ECS to access the IP address of a pod in a cluster that is in the same VPC as the ECS, the ECS CIDR block must be added to this parameter.
Pod's Access to Metadata (available only in CCE Turbo clusters)	Whether to allow pods in a cluster to access the node metadata, such as AZs and enterprise project IDs. For details, see **ECS Metadata Types**. This function is available only in clusters of v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, v1.28.3-r0, or later versions. • If a pod is created when the function is enabled, whether it can access metadata depends on the function status. • If a pod is created when the function is disabled or in a cluster of an earlier version, it cannot access metadata regardless of the function status. To grant a pod access to metadata, it must be rebuilt while the function is enabled.
Network Policies (supported by clusters using the container tunnel network model)	Whether to enable **network policies**. This configuration is supported by clusters of v1.25.16-r10, v1.27.16-r10, v1.28.15-r0, v1.29.10-r0, v1.30.6-r0, v1.31.1-r0, and later versions. • Disabled: The network policy capability is not supported, and the created policies do not take effect. • Enabled: If the CIDR blocks of a customer's service conflict with the on-premises CIDR blocks, the link to a newly added gateway may not be established. For example, when a cluster accesses an external address through a Direct Connect connection, the switch outside the cloud does not support **ip-option**. If the network policy is enabled, the network access may fail.

Service Settings

Table 17-3 Parameters

Parameter	Description
Request Forwarding	Forwarding mode of a cluster After a cluster is created, the service forwarding mode cannot be changed. IPVS and iptables are supported. For details, see **Comparing iptables and IPVS**.
IPv4 Service CIDR Block/IPv6 Service CIDR Block	Each Service in a cluster has its own IP address. When creating a CCE cluster, you can specify the Service address range (Service CIDR block). The Service CIDR block cannot overlap with the subnet or the container CIDR block. The Service CIDR block can be used only within a cluster.
Service Port Range	NodePort port range The default port range is 30000 to 32767. The port range can be changed to 20106 to 32767. After changing the value, go to the security group page and change the TCP/UDP port range of node security groups 30000 to 32767. Otherwise, ports other than the default port cannot be accessed from external networks. **NOTE** If the port number is smaller than 20106, a conflict may occur between the port and the system health check port, which may further lead to unavailable cluster. If the port number is greater than 32767, a conflict may occur between the port and the random port of the OS, which may further affect the performance.

Container CIDR Blocks (Available only in Clusters Using the VPC Network Model)

If a container CIDR block configured during cluster creation cannot meet service expansion requirements, you can add more container CIDR blocks. For details, see **Adding a Container CIDR Block for a Cluster**.

NOTE

- This function is available only for clusters of v1.19 or later using a VPC network.
- An added container CIDR block cannot be deleted.

Container Network (Available only in CCE Turbo Clusters)

If you want different namespaces or workloads to use different subnet CIDR blocks or security groups, you can create a policy to associate subnets or security groups with namespaces or workloads. For details, see **Binding a Subnet and Security Group to a Namespace or Workload Using a Container Network Configuration**.

- **Pod Subnet**: Pod IP addresses are allocated from this subnet. Only the pods in the same namespace or for running the same workload can communicate with each other.
- **Associate Security Group**: You can configure security group rules for pods in the same namespace or for running the same workload to customize access policies.

NOTE

This configuration is supported only by CCE Turbo clusters. Pod subnets can be deleted from clusters of v1.23.17-r0, v1.25.12-r0, v1.27.9-r0, v1.28.7-r0, v1.29.3-r0, or later versions.

Container Network Pre-binding Settings (Available only in CCE Turbo Clusters)

A CCE Turbo cluster requests for and binds an ENI or sub-ENI to each pod. Pods support fast scaling. However, it takes some time to create and bind an ENI to a pod, which slows down the pod startup speed if large-scale ENIs are to be created in batches. Dynamic container ENI pre-binding is enabled by default to speed up pod startup while improving the IP address usage. Cluster pre-binding policies take effect globally. Cluster nodes will pre-bind container ENIs based on the configured policies. To configure a separate pre-binding policy for a group of nodes, enable node pool pre-binding.

NOTE

This configuration is supported only by CCE Turbo clusters.

All Container ENI Pre-binding

- After this function is enabled, your cluster nodes will request for and bind the maximum number of ENIs supported by the node flavor. For example, if the maximum number of sub-ENIs supported by s7.large.2 nodes is 16, CCE will dynamically pre-bind 16 sub-ENIs to each node of this flavor.
- After this function is disabled, you can customize the pre-binding parameters on the console.

Table 17-4 Parameters of the dynamic ENI pre-binding policy

Parameter	Default Value	Description	Suggestion
Minimum Number of Container ENIs Bound to a Node	10	Minimum number of container ENIs bound to a node. The parameter value must be a positive integer. The value **10** indicates that there are at least 10 container ENIs bound to a node. If the number you entered exceeds the container ENI quota of the node, the ENI quota will be used.	Configure these parameters based on the number of pods.

Parameter	Default Value	Description	Suggestion
Upper Limit of Pre-bound Container ENIs	0	If the number of ENIs bound to a node exceeds the value of **nic-maximum-target**, the system does not proactively pre-bind ENIs. If the value of this parameter is greater than or equal to the value of **nic-minimum-target**, the check on the maximum number of the pre-bound ENIs is enabled. Otherwise, the check is disabled. The parameter value must be a positive integer. The value **0** indicates that the check on the upper limit of pre-bound container ENIs is disabled. If the number you entered exceeds the container ENI quota of the node, the ENI quota will be used.	Configure these parameters based on the number of pods.
Container ENIs Dynamically Pre-bound to a Node	2	Minimum number of pre-bound ENIs on a node. The value must be a number. When the value of **nic-warm-target** + the number of bound ENIs is greater than the value of **nic-maximum-target**, the system will pre-bind ENIs based on the difference between the value of **nic-maximum-target** and the number of bound ENIs.	Set this parameter to the number of pods that can be scaled out instantaneously within 10 seconds.

Parameter	Default Value	Description	Suggestion
Threshold for Unbinding Pre-bound Container ENIs	2	Only when the number of idle ENIs on a node minus the value of **nic-warm-target** is greater than the threshold, the pre-bound ENIs will be unbound and reclaimed. The value can only be a number. • Setting a larger value of this parameter slows down the recycling of idle ENIs and accelerates pod startup. However, the IP address usage decreases, especially when IP addresses are insufficient. Therefore, **exercise caution when increasing the value of this parameter**. • Setting a smaller value of this parameter accelerates the recycling of idle ENIs and improves the IP address usage. However, when a large number of pods increase instantaneously, the startup of some pods slows down.	Set this parameter based on the difference between the number of pods that are frequently scaled on most nodes within minutes and the number of pods that are instantly scaled out on most nodes within 10 seconds.

17.4 Scheduling

Basic kube-scheduler configurations and Volcano-backed advanced scheduling are available. You can enable advanced scheduling functions such as bin packing, priority-based scheduling and preemption, AI task performance enhancement, and heterogeneous resource management for improved cluster resource utilization at low costs.

Default Cluster Scheduler Configuration

Default cluster scheduler (default-scheduler)

The Kubernetes scheduler discovers newly created pods that have not been accepted by nodes and assigns them to proper nodes. You are allowed to use multiple schedulers in the same cluster. kube-scheduler is the default cluster scheduler provided by the Kubernetes community. CCE also supports the enhanced Volcano scheduler, which offers general computing capabilities like high-performance job scheduling engine, heterogeneous chip management, and task management.

You can use kube-scheduler together with the Volcano scheduler, or use kube-scheduler or Volcano scheduler separately.

Table 17-5 Default cluster scheduler

Scheduler Name	Description	Configuration
kube-scheduler	kube-scheduler provides the community native, standard scheduling capabilities. If kube-scheduler is set as the default scheduler in a cluster and the Volcano Scheduler add-on (**Volcano Scheduler**) is also installed in the cluster, the enhanced Volcano capabilities are enabled by default. This ensures that you have access to advanced scheduling capabilities, including resource utilization optimization, AI job performance enhancement, and heterogeneous resource management. They help reduce costs while improving resource utilization. In this case, kube-scheduler schedules common workloads, while the Volcano scheduler schedules some specified workloads. For details, see **Scheduling Workloads**.	kube-scheduler configurations: • **Scheduler Performance Configuration** • **Priority-based Scheduling** Enhanced configurations after the Volcano scheduler is installed: • **Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)** • **AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)** • **Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)**

Scheduler Name	Description	Configuration
Volcano scheduler (available in clusters v1.27 or later)	If the Volcano scheduler is set as the default scheduler in a cluster, kube-scheduler will no longer work. The Volcano scheduler schedules all workload tasks in the cluster. Volcano provides enhanced scheduling capabilities in addition to the capabilities provided by kube-scheduler. Before using this scheduler, you need to install the Volcano Scheduler add-on first. For details, see **Volcano Scheduler**.	Enhanced configurations of the Volcano scheduler: • **Priority-based Scheduling** • **Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)** • **AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)** • **Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)**

kube-scheduler

kube-scheduler provides the community native, standard scheduling capabilities.

Before **enabling volcano enhanced capabilities**, install Volcano Scheduler. Enabling this function will provide advanced scheduling capabilities, including optimizing resource utilization, enhancing AI job performance, and managing heterogeneous resources. This will ultimately improve cluster resource utilization and reduce costs.

Enhanced configurations of the Volcano scheduler:

- **Priority-based Scheduling**
- **Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)**
- **AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)**
- **Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)**

Scheduler Performance Configuration

NOTE

Only kube-scheduler supports this configuration.

Table 17-6 Parameters

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communication with kube-apiserver	• If the number of nodes in a cluster is less than 1,000, the default value is **100**. • If the number of nodes in a cluster is 1,000 or more, the default value is **200**.
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communication with kube-apiserver	• If the number of nodes in a cluster is less than 1,000, the default value is **100**. • If the number of nodes in a cluster is 1,000 or more, the default value is **200**.

Priority-based Scheduling

Scheduling based on priority

This is a basic scheduling capability and cannot be disabled. The scheduler preferentially guarantees the running of high-priority pods, and will not evict low-priority pods that are running. For details, see **Priority-based Scheduling and Preemption**.

Whether to enable preemption (supported by the Volcano scheduler)

After this function is enabled, when cluster resources are insufficient, the scheduler will proactively evict low-priority pods to make it possible to schedule pending high-priority pods. For details, see **Priority-based Scheduling and Preemption**.

NOTE

- This configuration is supported when Volcano is selected as the default scheduler.
- Enabling both priority-based preemption scheduling and delayed pod creation simultaneously is not possible.

Resource Utilization Optimization Scheduling (Supported by the Volcano Scheduler)

Bin packing

With this option enabled, the cluster scheduler schedules pods to nodes that have the most requested resources. This reduces resource fragments on each node and improves the resource utilization of the cluster. For details, see **Bin Packing**.

For details about the bin packing weight and weights for each resource to score nodes, see **Table 17-7**.

Table 17-7 Bin packing weight

Item	Description	Default Value
Binpack Scheduling Strategy Weight	A larger value indicates a higher weight of the bin packing policy in overall scheduling.	10
CPU Weight	A larger value indicates a higher cluster CPU usage.	1
Memory Weight	A larger value indicates a higher cluster memory usage.	1
Custom Resource Type	Other custom resource types requested by pods, for example, **nvidia.com/gpu**. A larger value indicates a higher usage of the specified cluster resource.	None

Load-aware scheduling (usage)

This function uses the Cloud Native Cluster Monitoring (kube-prometheus-stack) add-on to obtain the actual CPU and memory load of each node, calculates the average load of each node based on the specified period, and preferentially schedules jobs to the node with the lightest load to balance load. For details, see **Load-aware Scheduling**.

AI Job Performance Enhancement Scheduling (Supported by the Volcano Scheduler)

Fair Scheduling Policy (DRF)

Dominant Resource Fairness (DRF) is a scheduling algorithm based on the dominant resource of a container group. It supports fair allocation of multiple types of resources and is suitable for batch AI training and big data jobs. DRF is suitable for batch process small scale services like single AI model training and single big data computing and query, because it preferentially considers the throughput of services in clusters.

DRF helps you enhance the service throughput of clusters and improve service performance. For details, see **DRF**.

Workload Group Scheduling Policy (Gang)

Gang scheduling meets the scheduling requirements of "All or nothing" in the scheduling process and avoids the waste of cluster resources caused by arbitrary scheduling of pods. It is mainly used in scenarios that require multi-process collaboration, such as AI and big data scenarios.

Gang scheduling effectively resolves pain points such as resource waiting or deadlocks in distributed training jobs, thereby significantly improving the utilization of cluster resources. For details, see **Gang**.

Heterogeneous Resource Scheduling (Supported by the Volcano Scheduler)

Support GPU resource scheduling

To use this capability, the CCE AI Suite (NVIDIA GPU) add-on (**CCE AI Suite (NVIDIA GPU)**) must be installed. With this option enabled, GPUs can be used for AI training jobs, and the scheduler provides full GPU dispatch and GPU sharing to improve resource utilization.

Support NPU resource scheduling

To use this capability, the CCE AI Suite (Ascend NPU) add-on (**CCE AI Suite (Ascend NPU)**) must be installed. With this option enabled, NPUs can be used for AI training jobs, and the scheduler provides NPU topology-aware scheduling to improve training job efficiency.

17.5 Auto Scaling

Auto Scale-out Configuration

CCE Cluster Autoscaler comprehensively checks the resource statuses of an entire cluster. When the load of a microservice is high (for example, the CPU or memory usage is too high), it will add more pods to reduce the load.

Node Capacity Expansion Conditions

- Auto scale-out when the workload cannot be scheduled: If a workload pod cannot be scheduled, the system automatically scales out the node pool with auto scaling enabled. If the pod has been configured to be affinity for a node, the system will not automatically add more nodes.

  Such auto scaling works with an HPA policy. For details, see **Using HPA and CA for Auto Scaling of Workloads and Nodes**.
- User-defined policy switch: specifies whether to automatically scale out a node pool based on the **node scaling policies**. This function is enabled by default.

Upper limit of resources to be expanded

- **Total Nodes**: specifies how many nodes can be present in a cluster during scale-out. If there are more nodes in the cluster than the specified value, the cluster will not add nodes. The default value is determined by how many nodes a cluster can manage at most.
- **Total Cores**: specifies how many cores on all nodes can be present in a cluster during scale-out. If there are more cores in the cluster than the specified value, the cluster will not add nodes. By default, the number is not limited.

- **Total Memory** (GiB): specifies the upper limit of the total memory of all nodes in a cluster during scale-out. If the total memory exceeds the specified value, the cluster will not add nodes. By default, the number is not limited.

NOTE

When the total number of nodes, CPUs, and memory is collected, unavailable nodes in custom node pools are included but unavailable nodes in the default node pool are not included.

Scale-out Priority

You can drag and drop node pools in the list to adjust their scale-out priorities.

Auto Scale-in Configuration

CCE Cluster Autoscaler comprehensively checks the resource statuses of an entire cluster. Once it confirms that workload pods can be scheduled and run properly, it automatically obtains nodes for scale-in.

Node Scale-in Conditions

- **Node Resource Condition**: When the requested cluster node resources (both CPU and memory) are lower than a certain percentage (50% by default) for a period of time (10 minutes by default), a cluster scale-in is triggered.
- **Node Status Condition**: If a node is unavailable for a specified period of time, the node will be automatically reclaimed. The default value is 20 minutes.
- **Scale-in Exception Scenarios**: When a node meets the following exception scenarios, CCE will not scale in the node even if the node resources or status meets scale-in conditions:
  a. Resources on other nodes in the cluster are insufficient.
  b. Scale-in protection is enabled on the node. To enable or disable node scale-in protection, choose **Nodes** in the navigation pane and then click the **Nodes** tab. Locate the target node, choose **More**, and then enable or disable node scale-in protection in the **Operation** column.
  c. There is a pod with the non-scale label on the node.
  d. Policies such as reliability have been configured on some containers on the node.
  e. There are non-DaemonSet containers in the **kube-system** namespace on the node.
  f. (Optional) A container managed by a third-party pod controller is running on a node. Third-party pod controllers are for custom workloads except Kubernetes-native workloads such as Deployments and StatefulSets. Such controllers can be created using **CustomResourceDefinitions**.

Node Scale-in Policy

Table 17-8 Node scale-in policy configurations

Item	Description	Default Value
Number of Concurrent Scale-In Requests	Maximum number of idle nodes that can be deleted concurrently. Only idle nodes can be concurrently scaled in. Nodes that are not idle can only be scaled in one by one. **NOTE** During a node scale-in, if the pods on the node do not need to be evicted (such as DaemonSet pods), the node is idle. Otherwise, the node is not idle.	10
Node Recheck Timeout	Interval at which a node can be checked again after it is determined that the node cannot be scaled in	5 minutes
Cooldown Time	Cooldown period for starting scale-in evaluation again after auto scale-in is triggered in a cluster **NOTE** If both auto scale-out and scale-in exist in a cluster, set this parameter to 0 minutes. This prevents the node scale-in from being blocked due to continuous scale-out of some node pools or retries upon a scale-out failure, which results in unexpected waste of node resources.	10 minutes
	Cooldown period for starting scale-in evaluation again after auto scale-out is triggered in a cluster	10 minutes
	Cooldown period for starting scale-in evaluation again after auto scale-in triggered in a cluster failed	3 minutes

17.6 Monitoring

CCE monitors applications and resources and collects metrics and events to analyze application health status. You can choose **Settings** from the navigation pane, click the **Monitoring** tab, and change monitoring parameters on the console.

You will need to enable cluster monitoring (**Enabling Monitoring Center**) to use all monitoring functions.

Monitoring Configuration

Collection configuration

- **Preset Policies**: manage monitoring tasks of the Cloud Native Cluster Monitoring add-on in a visualized manner. For details, see **Managing Collection Tasks**.

- **ServiceMonitor Policies**: define the custom metric collection policies for Services. For details, see **Managing Collection Tasks**.

  For details about how to create a ServiceMonitor, see **Method 4: Configuring ServiceMonitor**.
- **PodMonitor Policies**: define the custom metric collection policies for pods. For details, see **Managing Collection Tasks**

  For details about how to create a PodMonitor, see **Method 3: Configuring PodMonitor**.
- **Targets**: statuses of the metric collection targets. For details, see **Managing Collection Tasks**.
- **Default Collection Period**: specifies the metric collection period of the Cloud Native Cluster Monitoring add-on. The default value is 15 seconds.
- **Customized indicator collection strategy**: After this function is enabled, Cloud Native Cluster Monitoring will collect custom metrics of applications. You need to provide custom metrics of applications based on Prometheus specifications, expose APIs for these metrics, and use the application images to deploy workloads in clusters. Prometheus will collect these metrics through the collection configuration. For details, see **Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.

Connect to AOM monitoring service

AOM uses Prometheus for container monitoring. To enable monitoring, you need to select an AOM instance first. With this function enabled, metrics will be reported to the selected AOM instance. Basic container metrics can be monitored for free, but other metrics are billed on a pay-per-use basis. For details about free metrics, see **Basic Metrics: Container Metrics**.

Access third-party monitoring platforms

After this function is enabled, Prometheus data can be reported to a third-party monitoring platform. You need to obtain the data reporting address and identity authentication credential of the third-party platform beforehand. For details, see **Reporting Prometheus Monitoring Data to a Third-Party Monitoring Platform**.

Log Configuration

Collection configuration

CCE collects, obtains, and analyzes Kubernetes container logs, including standard container output logs, container log files, node logs, and Kubernetes events.

NOTICE

When logs are reported to LTS, a default log group named **k8s-logs-***{Cluster ID}* will be created and will generate costs. For details about LTS pricing, see **Price Calculator**.

Log Type	View Log	LTS Log Group Name	Status	Reference
Container log	Container standard output	stdout-{Cluster ID}	To enable service log collection, you need to install the Cloud Native Log Collection add-on (**Cloud Native Log Collection**).	**Collecting Container Logs Using Cloud Native Log Collection**
Kubernetes event	Kubernetes event	event-{Cluster ID}	To enable service log collection, you need to install the Cloud Native Log Collection add-on (**Cloud Native Log Collection**).	**Collecting Kubernetes Events**
Kubernetes audit log	Kubernetes audit log	audit-{Cluster ID}	The option can be enabled separately.	**Collecting Kubernetes Audit Logs**
Control plane component log	kube-apiserver log	kube-apiserver-{Cluster ID}	The option can be enabled separately.	**Collecting Control Plane Component Logs**
	kube-controller-manager log	kube-controller-manager-{Cluster ID}	The option can be enabled separately.	
	kube-scheduler log	kube-scheduler-{Cluster ID}	The option can be enabled separately.	

Send stdout Logs to AOM 1.0 (**No more evolution**): Logging in AOM 1.0 has not been improved, so you are advised to disable this function and use LTS logging instead. Sending logs to AOM results in storage fees.

Kubernetes Events Reported to AOM

Once the Cluster Native Logging add-on (**Cloud Native Log Collection**) is installed in a cluster, Kubernetes events are reported to LTS by default, while this feature can be used to report Kubernetes events to AOM.

- **Abnormal events**: This option is enabled by default. All abnormal events are reported to AOM. You can click **Configure Blocklist** to add events that do not need to be reported to the blocklist. You can obtain event names in **CCE Events**.
- **Normal events**: If this option is enabled, normal events will be reported to AOM. The system is pre-configured to report some normal events. If you need to customize the events to be reported, click **Configure Trustlist** to add the events to the trustlist. You can obtain event names in **CCE Events**.

17.7 Kubernetes

Typical native configuration items are provided. You can configure native community management components such as kube-apiserver and kube-controller for the best cloud native experience.

API Server Configuration (kube-apiserver)

Container eviction configuration

By default, the default tolerance time applies to all containers in a cluster. You can also configure different tolerance times for pods. In this case, your custom settings take effect.

NOTE

It is recommended that you properly configure the tolerance times for pods, or certain problems may occur.

- If the parameter is set to a low value, containers may be moved frequently during brief fault scenarios like network jitter, which can impact services.
- If the parameter is set to a high value, containers may not be moved for a long period of time in the event of a node failure, which can impact services.

Table 17-9 Parameters

Item	Parameter	Description	Value
Toleration time for nodes in NotReady state	default-not-ready-toleration-seconds	Tolerance time when a node is not ready. If a node becomes unavailable, pods running on the node are evicted automatically after the tolerance time elapses. The default value is **300s**.	Default: 300s

Item	Parameter	Description	Value
Toleration time for nodes in unreachable state	default-unreachable-toleration-seconds	Tolerance time when a node is unreachable. If the environment is abnormal, for example, a node cannot be accessed (due to reasons such as abnormal node network), pods running on the node are evicted automatically after the tolerance time elapses. The default value is **300s**.	Default: 300s

Admission Controller Add-on Configurations

With Kubernetes, you can enable admission add-ons to limit and manage Kubernetes API objects (like pods, Services, and Deployments) prior to modifying them within a cluster.

NOTE

This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Table 17-10 Parameters

Item	Parameter	Description	Value
Node Restriction Add-on	enable-admission-plugin-node-restriction	This add-on allows the kubelet of a node to operate only the objects of the current node for enhanced isolation in multi-tenant scenarios or the scenarios with high security requirements. For details, see the **official documentation**.	Enable/ Disable
Pod Node Selector Add-on	enable-admission-plugin-pod-node-selector	This add-on allows cluster administrators to configure the default node selector through namespace annotations. In this way, pods run only on specific nodes and configurations are simplified.	Enable/ Disable

Item	Parameter	Description	Value
Pod Toleration Limit Add-on	enable-admission-plugin-pod-toleration-restriction	This add-on allows cluster administrators to configure the default value and limits of pod tolerations through namespaces for fine-grained control over pod scheduling and key resource protection.	Enable/ Disable

Service Account Token Volume Projection

Kubelet can project a service account token into a pod. You can specify the desired properties of the token, such as the API audiences. The token will become invalid against the API when either the pod or the service account is deleted. For details, see the **official documentation**.

NOTE

This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.

Table 17-11 Parameters

Item	Parameter	Description	Value
API Audience Settings	api-audiences	Audiences for a service account token. The Kubernetes component for authenticating service account tokens checks whether the token used in an API request specifies authorized audiences. Configuration suggestion: Accurately configure audiences according to the communication needs among cluster services. By doing so, the service account token is used for authentication only between authorized services, which enhances security. **NOTE** An incorrect configuration may lead to an authentication communication failure between services or an error during token verification.	Default value: "https://kubernetes.default.svc.cluster.local" Multiple values can be configured, which are separated by commas (,).

Item	Parameter	Description	Value
Service Account Token Issuer Identity	service-account-issuer	Entity identifier for issuing a service account token, which is the value identified by the **iss** field in the payload of the service account token. Configuration suggestion: Ensure the configured issuer URL can be accessed in the cluster and trusted by the authentication system in the cluster. **NOTE** If your specified issuer URL is untrusted or inaccessible, the authentication process based on the service account may fail.	Default value: "https://kubernetes.default.svc.cluster.local" Multiple values can be configured, which are separated by commas (,).

Controller Configuration (kube-controller-manager)

Common Configurations of the Controller

- **Controller performance configuration**: used to configure performance parameters for the controller to access kube-apiserver.

  NOTE

  It is recommended that you properly configure the controller performance settings, or certain problems may occur.

  - If a parameter is set to a small value, client traffic limiting may be triggered, affecting controller performance.
  - If a parameter is set to a large value, kube-apiserver may be overloaded.

Table 17-12 Parameters

Item	Parameter	Description	Value
QPS for communicating with kube-apiserver	kube-api-qps	QPS for communication with kube-apiserver	• If the number of nodes in a cluster is less than 1,000, the default value is **100**. • If the number of nodes in a cluster is 1,000 or more, the default value is **200**.
Burst for communicating with kube-apiserver	kube-api-burst	Burst for communication with kube-apiserver	• If the number of nodes in a cluster is less than 1,000, the default value is **100**. • If the number of nodes in a cluster is 1,000 or more, the default value is **200**.

- **Cluster controller concurrent configuration**: specifies the number of resource objects that are allowed to synchronize simultaneously. A larger value indicates a quicker response and higher CPU (and network) load.

NOTE

It is recommended that you properly configure the controller concurrency, or certain problems may occur.

- If a parameter is set to a small value, the controller may respond slowly.
- If a parameter is set to a large value, the cluster management plane will be overloaded.

Table 17-13 Parameters

Item	Parameter	Description	Value
Number of concurrent processing of deployment	concurrent-deployment-syncs	Number of Deployment objects that can be synchronized concurrently. A larger value indicates a quicker response to Deployments and higher CPU (and network bandwidth) pressure.	Default: 5
Concurrent processing number of endpoint	concurrent-endpoint-syncs	Number of endpoints that can be concurrently synchronized. A larger value indicates faster update of endpoints and higher CPU (and network) pressure.	Default: 5
Concurrent number of garbage collectors	concurrent-gc-syncs	Number of garbage collector workers that are allowed to synchronize concurrently.	Default: 20
Number of job objects allowed to sync simultaneously	concurrent-job-syncs	Number of job objects that can be synchronized concurrently. A larger value indicates a quicker response to jobs and higher CPU (and network) usage.	Default: 5
CronJob	concurrent-cron-job-syncs	Number of CronJob objects that can be synchronized concurrently. A larger value indicates a quicker response to CronJobs and higher CPU (and network) usage.	Default: 5

Item	Parameter	Description	Value
Number of concurrent processing of namespace	concurrent-namespace-syncs	Number of namespace objects that can be synchronized concurrently. A larger value indicates a quicker response to namespaces and higher CPU (and network) usage.	Default: 10
Concurrent processing number of replicaset	concurrent-replicaset-syncs	Number of ReplicaSet objects that can be synchronized concurrently. A larger value indicates a quicker response to ReplicaSet management and higher CPU (and network) usage.	Default: 5
Number of concurrent processing of resource quota	concurrent-resource-quota-syncs	Number of ResourceQuota objects that can be synchronized concurrently. A larger value indicates a faster response to quota management and higher CPU (and network) usage.	Default: 5
Concurrent processing number of service	concurrent-service-syncs	Number of Service objects that can be synchronized concurrently. A larger value indicates a faster response to Service management and higher CPU (and network) usage.	Default: 10
Concurrent processing number of serviceaccount-token	concurrent-serviceaccount-token-syncs	Number of service account token objects that can be synchronized concurrently. A larger value indicates faster token generation and higher CPU (and network) usage.	Default: 5
Concurrent processing of ttl-after-finished	concurrent-ttl-after-finished-syncs	Number of **ttl-after-finished-controller** workers that can be synchronized concurrently.	Default: 5

Item	Parameter	Description	Value
RC	concurrent_rc_syncs	Number of replication controllers that can be synchronized concurrently. A larger value indicates faster replica management operations and higher CPU (and network) usage. **NOTE** This parameter is used only in clusters of v1.19 or earlier.	Default: 5
RC	concurrent-rc-syncs	Number of replication controllers that can be synchronized concurrently. A larger value indicates faster replica management operations and higher CPU (and network) usage. **NOTE** This parameter is used only in clusters of v1.21 to v1.23. In clusters of v1.25 and later, this parameter is deprecated (officially deprecated from v1.25.3-r0 on).	Default: 5
HPA	concurrent-horizontal-pod-autoscaler-syncs	Maximum number of HPA auto scaling requests that can be processed concurrently. A larger value indicates a faster HPA auto scaling and higher CPU (and network) usage. This parameter is available only in clusters of v1.27 or later.	Default: 5 Value range: 1 to 50

Node lifecycle controller (node-lifecycle-controller) configuration

NOTE

This parameter is available only in clusters of v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later versions.

Table 17-14 Parameters

Item	Parameter	Description	Value
Unhealthy AZ Threshold	unhealthy-zone-threshold	When more than a certain proportion of pods in an AZ are unhealthy, the AZ itself will be considered unhealthy, and scheduling pods to nodes in that AZ will be restricted to limit the impacts of the unhealthy AZ. **NOTE** If the parameter is set to a large value, pods in unhealthy AZs will be migrated in a large scale, which may lead to risks such as overloaded clusters.	Default: 0.55
Node Eviction Rate	node-eviction-rate	This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is healthy. The default value is **0.1**, indicating that pods can be evicted from at most one node every 10 seconds. **NOTE** Configure this parameter based on the size of the cluster. The number of pods to be evicted in each batch should not exceed 300. If the parameter is set to a large value, the cluster may be overloaded. Additionally, if too many pods are evicted, they cannot be rescheduled, which will slow down fault recovery.	Default: 0.1

Item	Parameter	Description	Value
Secondary Node Eviction Rate	secondary-node-eviction-rate	This parameter specifies the number of nodes that pods are deleted from per second in a cluster when the AZ is unhealthy. The default value is **0.01**, indicating that pods can be evicted from at most one node every 100 seconds. **NOTE** Configure this parameter with **node-eviction-rate** and set it to one-tenth of **node-eviction-rate**. There is no need to set the parameter to a large value for nodes in an unhealthy AZ, and this configuration may result in overloaded clusters.	Default: 0.01
Large Cluster Threshold	large-cluster-size-threshold	If the number of nodes in a cluster is greater than the value of this parameter, this is a large cluster. Configuration suggestion: For the clusters with a large number of nodes, configure a relatively larger value than the default one for higher performance and faster responses of controllers. Retain the default value for small clusters. Before adjusting the value of this parameter in a production environment, check the impact of the change on cluster performance in a test environment. **NOTE** kube-controller-manager automatically adjusts configurations for large clusters to optimize the cluster performance. Therefore, an excessively small threshold for small clusters will deteriorate the cluster performance.	Default: 50

Load elastic scaling synchronization cycle

Table 17-15 Parameters

Item	Parameter	Description	Value
Cluster elastic computing period	horizontal-pod-autoscaler-sync-period	Period for the horizontal pod autoscaler to perform elastic scaling on pods. A smaller value will result in a faster auto scaling response and higher CPU load. **NOTE** Make sure to configure this parameter properly as a lengthy period can cause the controller to respond slowly, while a short period may overload the cluster control plane.	Default: 15s
Horizontal Pod Scaling Tolerance	horizontal-pod-autoscaler-tolerance	The configuration determines how quickly the horizontal pod autoscaler will act to auto scaling policies. If the parameter is set to **0**, auto scaling will be triggered immediately when the related metrics are met. Configuration suggestion: If the service resource usage increases sharply over time, retain a certain tolerance to prevent auto scaling which is beyond expectation in high resource usage scenarios.	Default: 0.1

Item	Parameter	Description	Value
HPA CPU Initialization Period	horizontal-pod-autoscaler-cpu-initialization-period	During the period specified by this parameter, the CPU usage data used in HPA calculation is limited to pods that are both ready and have recently had their metrics collected. You can use this parameter to filter out unstable CPU usage data during the early stage of pod startup. This helps prevent incorrect scaling decisions based on momentary peak values. Configuration suggestion: If you find that HPA is making incorrect scaling decisions due to CPU usage fluctuations during pod startup, increase the value of this parameter to allow for a buffer period of stable CPU usage. **NOTE** Make sure to configure this parameter properly as a small value may trigger unnecessary scaling based on peak CPU usage, while a large value may cause scaling to be delayed. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default: 5 minutes

Item	Parameter	Description	Value
HPA Initial Readiness Delay	horizontal-pod-autoscaler-initial-readiness-delay	After CPU initialization, this period allows HPA to use a less strict criterion for getting CPU metrics. During this period, HPA will gather data on the CPU usage of the pod for scaling, regardless of any changes in the pod's readiness status. This parameter ensures continuous tracking of CPU usage, even when the pod status changes frequently. Configuration suggestion: If the readiness status of pods fluctuates after startup and you want to prevent HPA misjudgment caused by the fluctuation, increase the value of this parameter to allow HPA to gather more comprehensive CPU usage data. **NOTE** Configure this parameter properly. If it is set to a small value, an unnecessary scale-out may occur due to CPU data fluctuations when the pod is just ready. If it is set to a large value, HPA may not be able to make a quick decision when a rapid response is needed. This parameter is available only in clusters of v1.23.16-r0, v1.25.11-r0, v1.27.8-r0, v1.28.6-r0, v1.29.2-r0, or later versions.	Default: 30s

Threshold configuration of the number of terminal state pods that trigger recycling

Table 17-16 Parameters

Item	Parameter	Description	Value
The maximum number of terminated pods that can be kept before the Pod GC deletes the terminated pod	terminated-pod-gc-threshold	Number of terminated pods that can exist before the terminated pod garbage collector starts deleting terminated pods **NOTE** It is recommended that you properly configure this parameter. If the value is too large, there may be a large number of terminated pods in the cluster, which will further affect the performance of list queries and result in an overloaded cluster.	Default: 1000 Value range: 0 to 100000

Resource quota controller (resource-quota-controller) configuration

NOTE

In high-concurrency scenarios (for example, creating pods in batches), the resource quota management may cause some requests to fail due to conflicts. Do not enable this function unless necessary. To enable this function, ensure that there is a retry mechanism in the request client.

Table 17-17 Parameters

Item	Parameter	Description	Value
Enable resource quota management	enable-resource-quota	With resource quota management, you are allowed to control the number of workloads (such as Deployments and pods) and the upper limits of resources (such as CPUs and memory) in namespaces or related dimensions. Namespaces control quotas through the ResourceQuota objects. • **false**: Auto creation is disabled. • **true**: Auto creation is enabled. For details about the resource quota defaults, see **Configuring Resource Quotas**.	Default: false

17.8 Heterogeneous Resources

GPU Settings

- **GPU Virtualization**: CCE uses proprietary xGPU virtualization to dynamically divide the GPU memory and compute. A single GPU can be virtualized into up to 20 virtual GPUs. Virtualization is more flexible than static allocation. You can specify the number of GPUs on the basis of stable services to improve GPU utilization. For details, see **Overview**.
- **Default Cluster Driver**: specifies the default GPU driver version used by the GPU nodes in a cluster. To use a custom driver, enter the download link of the NVIDIA driver. For details, see **Obtaining the Driver Link from Public Network**.
- **Node Pool Configurations**: If you do not want all GPU nodes in a cluster to use the same driver, CCE allows you to install a different GPU driver for each node pool. After you customize a GPU driver for a node pool, nodes in the node pool will preferentially use the custom driver. Nodes for which no driver is specified will use the cluster's default driver.

  NOTE

  - The system installs the driver of the version specified for the node pool. The driver applies only to new nodes in the node pool.
  - After the driver version is updated, it takes effect on the nodes newly added to the node pool. Existing nodes must restart to apply the changes.
  - When installing the CCE AI Suite (NVIDIA GPU) add-on of v2.7.2 or later, you can configure xGPU virtualization for node pools.

NPU Settings

If driver selection is disabled, you cannot specify driver versions or maintain drivers using the CCE AI Suite (Ascend NPU) add-on. When you add an NPU node on the console, the system adds the command to install an NPU driver (version and type decided by the system) and automatically restarts the node after the driver installation is complete. Adding an NPU node in another way, such as using an API, requires you to add the driver installation command to the post-installation command.

Once driver selection is enabled, CCE AI Suite (Ascend NPU) will automatically install the driver during its startup based on the driver configuration of the corresponding model. This makes driver maintenance more flexible. You are advised to use the default driver version. You can also enter the complete driver address to use custom drivers.

The following table lists what NPUs and OS specifications are supported.

NPU Type	Supported OS
D310	EulerOS 2.5 x86, CentOS 7.6 x86, EulerOS 2.9 x86, and EulerOS 2.8 arm

18 Storage Management: FlexVolume (Deprecated)

18.1 FlexVolume Overview

In container storage, you can use different types of volumes and mount them to containers in pods as many as you want.

In CCE, container storage is backed both by Kubernetes-native objects, such as emptyDir, hostPath, secret, and ConfigMap, and by cloud storage services.

CCE clusters of **1.13 and earlier versions** use the **storage-driver** add-on to connect to cloud storage services to support Kubernetes FlexVolume driver for container storage. The FlexVolume driver has been deprecated in favor of the Container Storage Interface (CSI). **The CSI add-on Everest is installed in CCE clusters of 1.15 and later versions by default.** For details, see **Overview**.

NOTE

- In CCE clusters earlier than Kubernetes 1.13, end-to-end capacity expansion of container storage is not supported, and the PVC capacity is inconsistent with the storage capacity.
- **In a cluster of v1.13 or earlier**, when an upgrade or bug fix is available for storage functionalities, you only need to install or upgrade the storage-driver add-on. Upgrading the cluster or creating a cluster is not required.

Notes and Constraints

- For clusters created in CCE, Kubernetes v1.15.11 is a transitional version in which the FlexVolume plugin (**storage-driver**) is compatible with the CSI plugin (**Everest**). Clusters of v1.17 and later versions do not support FlexVolume anymore. Use the Everest add-on.
- The FlexVolume plugin storage-driver will be maintained by Kubernetes developers, but new functionality will only be added to CSI **CCE Container Storage (Everest)**. Do not create storage that access the FlexVolume storage-driver in CCE anymore. Otherwise, the storage resources may not function properly.

Checking Storage Add-ons

Step 1 Log in to the CCE console.

Step 2 In the navigation tree on the left, click **Add-ons**.

Step 3 Click the **Add-on Instance** tab.

Step 4 Select a cluster in the upper right corner. The default storage add-on installed during cluster creation is displayed.

----End

Differences Between CSI and FlexVolume Plugins

Table 18-1 CSI and FlexVolume

Kubernetes Solution	CCE Add-on	Feature	Recommendation
CSI	everest	CSI was developed as a standard for exposing arbitrary block and file storage systems to containerized workloads. Using CSI, third-party storage providers can deploy plugins exposing new storage systems in Kubernetes without having to touch the core Kubernetes code. In CCE, the Everest add-on is installed by default in clusters of Kubernetes v1.15 or later to access storage services such as EVS, SFS, OBS, and SFS Turbo. The Everest add-on consists of: • **everest-csi-controller** for storage volume creation, deletion, capacity expansion, and cloud disk snapshots • **everest-csi-driver** for mounting, unmounting, and formatting storage volumes on nodes For details, see **Everest**.	The **Everest** add-on is installed by default in clusters of **v1.15 or later**. CCE will mirror the Kubernetes community by providing continuous support for updated CSI capabilities.

Kubernetes Solution	CCE Add-on	Feature	Recommendation
FlexVolume	storage-driver	FlexVolume is an out-of-tree plugin interface that has existed in Kubernetes since version 1.2 (before CSI). CCE provided FlexVolume volumes through the storage-driver add-on installed in clusters of Kubernetes v1.13 and earlier versions. This add-on connects clusters to storage services such as EVS, SFS, OBS, and SFS Turbo. For details, see **storage-driver**.	For the created clusters of **v1.13 or earlier**, the installed FlexVolume plugin (CCE add-on **storage-driver**) can still be used. CCE has stopped providing update support for this add-on. **Upgrade these clusters**.

NOTE

- Either CSI or FlexVolume can be used in one cluster.
- The FlexVolume plugin cannot be replaced by the CSI plugin in clusters of v1.13 or earlier. To change the FlexVolume plugin to CSI, you can only upgrade these clusters. For details, see **Cluster Upgrade Path**.

18.2 Changing the Storage Class Used by a Cluster of v1.15 from FlexVolume to CSI Everest

In clusters later than v1.15.11-r1, CSI (the everest add-on) has taken over all functions of fuxi FlexVolume (the storage-driver add-on) for managing container storage. You are advised to use CSI Everest.

To migrate your storage volumes, create a static PV to associate with the original underlying storage, and then create a PVC to associate with this static PV. When you upgrade your application, mount the new PVC to the original mounting path to migrate the storage volumes.

WARNING

Services will be interrupted during the migration. Therefore, properly plan the migration and back up data.

Procedure

Step 1 (Optional) Back up data to prevent data loss in case of exceptions.

Step 2 Configure a YAML file of the PV in the CSI format according to the PV in the FlexVolume format and associate the PV with the existing storage.

To be specific, run the following commands to configure the pv-example.yaml file, which is used to create a PV.

touch pv-example.yaml

vi pv-example.yaml

Configuration example of **a PV for an EVS volume**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeAttributes:
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SAS
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-2 EVS volume configuration parameters

Parameter	Description
failure-domain.beta.kubernetes.io/region	Region where the EVS disk is located. Use the same value as that of the FlexVolume PV.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS disk is located. Use the same value as that of the FlexVolume PV.
name	Name of the PV, which must be unique in the cluster.
storage	EVS volume capacity in the unit of Gi. Use the value of **spec.capacity.storage** of the FlexVolume PV.
driver	Storage driver used to attach the volume. Set the driver to **disk.csi.everest.io** for the EVS volume.
volumeHandle	Volume ID of the EVS disk. Use the value of **spec.flexVolume.options.volumeID** of the FlexVolume PV.

Parameter	Description
everest.io/disk-mode	EVS disk mode. Use the value of **spec.flexVolume.options.disk-mode** of the FlexVolume PV.
everest.io/disk-volume-type	EVS disk type. Currently, high I/O (SAS) and ultra-high I/O (SSD) are supported. Use the value of **kubernetes.io/volumetype** in the storage class corresponding to **spec.storageClassName** of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class associated with the storage volume. Set this field to **csi-disk** for EVS disks.

Configuration example of **a PV for an SFS volume**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: nas.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: sfs-nas01.ap-southeast-1.myhuaweicloud.com:/share-436304e8
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 682f00bb-ace0-41d8-9b3e-913c9aa6b695
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-nas
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-3 SFS volume configuration parameters

Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	File storage size in the unit of Gi. Use the value of **spec.capacity.storage** of the FlexVolume PV.
driver	Storage driver used to attach the volume. Set the driver to **nas.csi.everest.io** for the file system.
everest.io/share-export-location	Shared path of the file system. Use the value of **spec.flexVolume.options.deviceMountPath** of the FlexVolume PV.

Parameter	Description
volumeHandle	File system ID. Use the value of **spec.flexVolume.options.volumeID** of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-nas**.

Configuration example of **a PV for an OBS volume**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    fsType: s3fs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: ap-southeast-1
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-4 OBS volume configuration parameters

Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	Storage capacity, in the unit of Gi. Set this parameter to the fixed value **1Gi**.
driver	Storage driver used to attach the volume. Set the driver to **obs.csi.everest.io** for the OBS volume.
fsType	File type. Value options are **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. Set this parameter according to the value of **spec.flexVolume.options.posix** of the FlexVolume PV. If the value of **spec.flexVolume.options.posix** is **true**, set this parameter to **obsfs**. If the value is **false**, set this parameter to **s3fs**.

Parameter	Description
everest.io/obs-volume-type	Storage class, including **STANDARD** (standard bucket) and **WARM** (infrequent access bucket). Set this parameter according to the value of **spec.flexVolume.options.storage_class** of the FlexVolume PV. If the value of **spec.flexVolume.options.storage_class** is **standard**, set this parameter to **STANDARD**. If the value is **standard_ia**, set this parameter to **WARM**.
everest.io/region	Region where the OBS bucket is located. Use the value of **spec.flexVolume.options.region** of the FlexVolume PV.
volumeHandle	OBS bucket name. Use the value of **spec.flexVolume.options.volumeID** of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-obs**.

Configuration example of **a PV for an SFS Turbo volume**:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: sfsturbo.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: 192.168.0.169:/
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-sfsturbo
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-5 SFS Turbo volume configuration parameters

Parameter	Description
name	Name of the PV, which must be unique in the cluster.
storage	File system size. Use the value of **spec.capacity.storage** of the FlexVolume PV.
driver	Storage driver used to attach the volume. Set it to **sfsturbo.csi.everest.io**.

Parameter	Description
everest.io/share-export-location	Shared path of the SFS Turbo volume. Use the value of **spec.flexVolume.options.deviceMountPath** of the FlexVolume PV.
volumeHandle	SFS Turbo volume ID. Use the value of **spec.flexVolume.options.volumeID** of the FlexVolume PV.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-sfsturbo** for SFS Turbo volumes.

Step 3 Configure a YAML file of the PVC in the CSI format according to the PVC in the FlexVolume format and associate the PVC with the PV created in **Step 2**.

To be specific, run the following commands to configure the pvc-example.yaml file, which is used to create a PVC.

touch pvc-example.yaml

vi pvc-example.yaml

Configuration example of **a PVC for an EVS volume**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: <zone name>
  annotations:
    everest.io/disk-volume-type: SAS
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName:  pv-evs-example
  storageClassName: csi-disk
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-6 PVC configuration parameters for an EVS volume

Parameter	Description
failure-domain.beta.kubernetes.io/region	Region where the cluster is located. Use the same value as that of the FlexVolume PVC.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS disk is deployed. Use the same value as that of the FlexVolume PVC.

Parameter	Description
everest.io/disk-volume-type	Storage class of the EVS disk. The value can be **SAS** or **SSD**. Set this parameter to the same value as that of the PV created in **Step 2**.
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storage	Requested capacity of the PVC, which must be the same as the storage size of the existing PV.
volumeName	Name of the PV. Set this parameter to the name of the static PV in **Step 2**.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-disk** for EVS disks.

Configuration example of **a PVC for an SFS volume**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-nas
  volumeName: pv-sfs-example
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-7 PVC configuration parameters for an SFS volume

Parameter	Description
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)

Parameter	Description
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
storageClassName	Set this field to **csi-nas**.
volumeName	Name of the PV. Set this parameter to the name of the static PV in **Step 2**.

Configuration example of **a PVC for an OBS volume**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-8 PVC configuration parameters for an OBS volume

Parameter	Description
everest.io/obs-volume-type	OBS volume type, which can be **STANDARD** (standard bucket) and **WARM** (infrequent access bucket). Set this parameter to the same value as that of the PV created in **Step 2**.
csi.storage.k8s.io/fstype	File type, which can be **obsfs** or **s3fs**. The value must be the same as that of **fsType** of the static OBS volume PV.
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.

Parameter	Description
storage	Storage capacity, in the unit of Gi. Set this parameter to the fixed value **1Gi**.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-obs**.
volumeName	Name of the PV. Set this parameter to the name of the static PV created in **Step 2**.

Configuration example of **a PVC for an SFS Turbo volume**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-sfsturbo
  volumeName: pv-efs-example
```

Pay attention to the fields in bold and red. The parameters are described as follows:

Table 18-9 PVC configuration parameters for an SFS Turbo volume

Parameter	Description
name	PVC name, which must be unique in the namespace. The value must be unique in the namespace. (If the PVC is dynamically created by a stateful application, the value of this parameter must be the same as the name of the FlexVolume PVC.)
namespace	Namespace to which the PVC belongs. Use the same value as that of the FlexVolume PVC.
storageClassName	Name of the Kubernetes storage class. Set this field to **csi-sfsturbo**.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV. Set this parameter to the name of the static PV created in **Step 2**.

Step 4 Upgrade the workload to use a new PVC.

For Deployments

1. Run the **kubectl create -f** commands to create a PV and PVC.

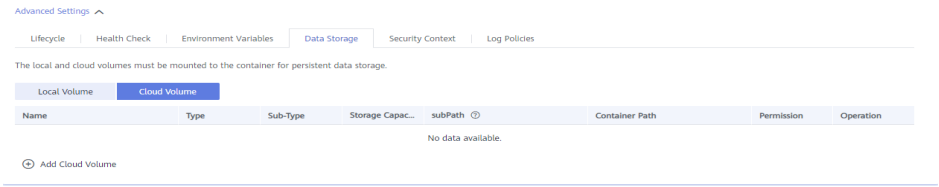
   kubectl create -f pv-example.yaml

   kubectl create -f pvc-example.yaml

   NOTE

   Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.

2. Go to the CCE console. On the workload upgrade page, click **Upgrade** > **Advanced Settings** > **Data Storage** > **Cloud Storage**.



3. Uninstall the old storage and add the PVC in the CSI format. Retain the original mounting path in the container.
4. Click **Submit**.
5. Wait until the pods are running.

For StatefulSets that use existing storage

1. Run the **kubectl create -f** commands to create a PV and PVC.

   kubectl create -f pv-example.yaml

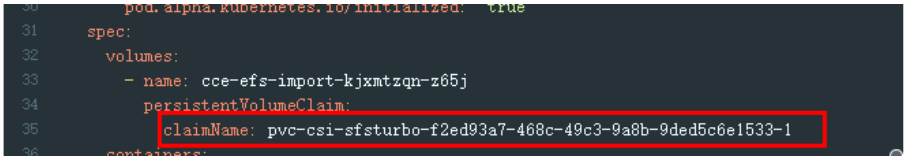
   kubectl create -f pvc-example.yaml

   NOTE

   Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.

2. Run the **kubectl edit** command to edit the StatefulSet and use the newly created PVC.

   kubectl edit sts sts-example -n xxx



   NOTE

   Replace **sts-example** in the preceding command with the actual name of the StatefulSet to upgrade. **xxx** indicates the namespace to which the StatefulSet belongs.

3. Wait until the pods are running.

NOTE

The current console does not support the operation of adding new cloud storage for StatefulSets. Use the kubectl commands to replace the storage with the newly created PVC.

For StatefulSets that use dynamically allocated storage

1. Back up the PV and PVC in the flexVolume format used by the StatefulSet.
   kubectl get pvc xxx -n {namespaces} -oyaml > pvc-backup.yaml
   kubectl get pv xxx -n {namespaces} -oyaml > pv-backup.yaml
2. Change the number of pods to **0**.
3. On the storage page, disassociate the flexVolume PVC used by the StatefulSet.
4. Run the **kubectl create -f** commands to create a PV and PVC.
   kubectl create -f pv-example.yaml
   kubectl create -f pvc-example.yaml

   NOTE

   Replace the example file name **pvc-example.yaml** in the preceding commands with the names of the YAML files configured in **Step 2** and **Step 3**.
5. Change the number of pods back to the original value and wait until the pods are running.

NOTE

The dynamic allocation of storage for StatefulSets is achieved by using **volumeClaimTemplates**. This field cannot be modified by Kubernetes. Therefore, data cannot be migrated by using a new PVC.

The PVC naming rule of the **volumeClaimTemplates** is fixed. When a PVC that meets the naming rule exists, this PVC is used.

Therefore, disassociate the original PVC first and then create a PVC with the same name in the CSI format.

6. (Optional) Recreate the stateful application to ensure that a CSI PVC is used when the application is scaled out. Otherwise, FlexVolume PVCs are used in scaling out.

- Run the following command to obtain the YAML file of the StatefulSet:

kubectl get sts xxx -n {namespaces} -oyaml > sts.yaml

- Run the following command to back up the YAML file of the StatefulSet:

cp sts.yaml sts-backup.yaml

- Modify the definition of **volumeClaimTemplates** in the YAML file of the StatefulSet.

vi sts.yaml

Configuration example of **volumeClaimTemplates for an EVS volume**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161070049798261342
      namespace: default
      creationTimestamp: null
      annotations:
        everest.io/disk-volume-type: SAS
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-disk
```


The parameter value must be the same as the PVC of the EVS volume created in **Step 3**.

Configuration example of **volumeClaimTemplates for an SFS volume**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161063441560279697
      namespace: default
      creationTimestamp: null
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 10Gi
      storageClassName: csi-nas
```

The parameter value must be the same as the PVC of the SFS volume created in **Step 3**.

Configuration example of **volumeClaimTemplates for an OBS volume**:

```
volumeClaimTemplates:
  - metadata:
      name: pvc-161070100417416148
      namespace: default
      creationTimestamp: null
      annotations:
        csi.storage.k8s.io/fstype: s3fs
        everest.io/obs-volume-type: STANDARD
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 1Gi
      storageClassName: csi-obs
```

The parameter value must be the same as the PVC of the OBS volume created in **Step 3**.

- Delete the StatefulSet.

kubectl delete sts xxx -n {namespaces}

- Create the StatefulSet.

kubectl create -f sts.yaml

Step 5 Check service functions.

1. Check whether the application is running properly.
2. Checking whether the data storage is normal.

NOTE

If a rollback is required, perform **Step 4**. Select the PVC in FlexVolume format and upgrade the application.

Step 6 Uninstall the PVC in the FlexVolume format.

If the application functions normally, unbind the PVC in the FlexVolume format on the storage management page.

You can also run the kubectl command to delete the PVC and PV of the FlexVolume format.

CAUTION

Before deleting a PV, change the persistentVolumeReclaimPolicy of the PV to **Retain**. Otherwise, the underlying storage will be reclaimed after the PV is deleted.

If the cluster has been upgraded before the storage migration, PVs may fail to be deleted. You can remove the PV protection field **finalizers** to delete PVs.

kubectl patch pv {pv_name} -p '{"metadata":{"finalizers":null}}'

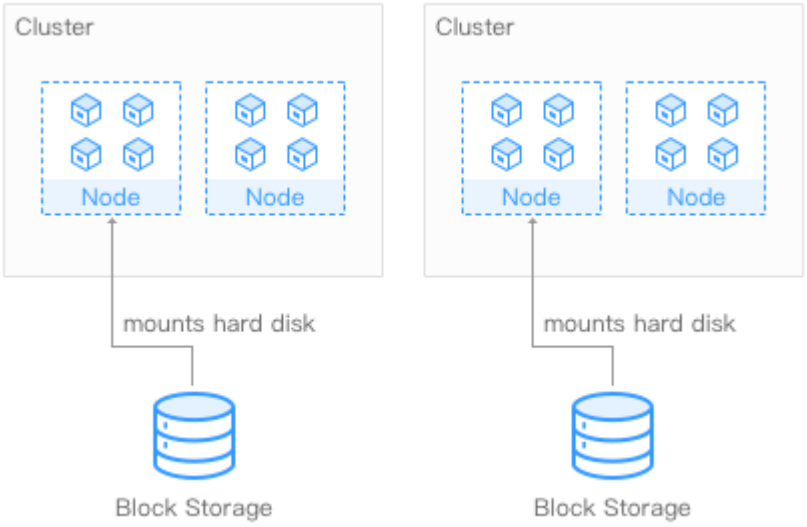
----End

18.3 Using EVS Disks as Storage Volumes

18.3.1 Overview

To achieve persistent storage, CCE allows you to mount the storage volumes created from Elastic Volume Service (EVS) disks to a path of a container. When the container is migrated, the mounted EVS volumes are also migrated. By using EVS volumes, you can mount the remote file directory of storage system into a container so that data in the data volume is permanently preserved even when the container is deleted.

Figure 18-1 Mounting EVS volumes to CCE



Description

- **User-friendly**: Similar to formatting disks for on-site servers in traditional layouts, you can format block storage (disks) mounted to cloud servers, and create file systems on them.

- **Data isolation**: Each server uses an independent block storage device (disk).
- **Private network**: User can access data only in private networks of data centers.
- **Capacity and performance**: The capacity of a single volume is limited (TB-level), but the performance is excellent (ms-level read/write I/O latency).
- **Restriction**: EVS disks that have partitions or have non-ext4 file systems cannot be imported.
- **Applications**: HPC, enterprise core applications running in clusters, enterprise application systems, and development and testing. These volumes are often used by single-pod Deployments and jobs, or exclusively by each pod in a StatefulSet. EVS disks are non-shared storage and cannot be attached to multiple nodes at the same time. If two pods are configured to use the same EVS disk and the two pods are scheduled to different nodes, one pod cannot be started because the EVS disk cannot be attached to it.

18.3.2 (kubectl) Automatically Creating an EVS Disk

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **pvc-evs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-evs-auto-example.yaml

vi pvc-evs-auto-example.yaml

Example YAML file for clusters of v1.9, v1.11, and v1.13:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto-example
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```


Table 18-10 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	EVS disk type. The value is in lowercase.
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	Storage capacity in the unit of Gi.
accessModes	Read/write mode of the volume. You can set this parameter to **ReadWriteMany** (shared volume) and **ReadWriteOnce** (non-shared volume).

Step 3 Run the following command to create a PVC.

kubectl create -f pvc-evs-auto-example.yaml

After the command is executed, an EVS disk is created in the partition where the cluster is located. Choose **Storage** > **EVS** to view the EVS disk. Alternatively, you can view the EVS disk based on the volume name on the EVS console.

----End

18.3.3 (kubectl) Creating a PV from an Existing EVS Disk

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Log in to the EVS console, create an EVS disk, and record the volume ID, capacity, and disk type of the EVS disk.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create two YAML files for creating the PersistentVolume (PV) and PersistentVolumeClaim (PVC). Assume that the file names are **pv-evs-example.yaml** and **pvc-evs-example.yaml**.

touch pv-evs-example.yaml pvc-evs-example.yaml

Kubernetes Cluster Version	Description	YAML Example
1.11.7 ≤ K8s version ≤ 1.13	Clusters from v1.11.7 to v1.13	Example YAML
1.11 ≤ K8s version < 1.11.7	Clusters from v1.11 to v1.11.7	Example YAML
K8s version = 1.9	Clusters of v1.9	Example YAML

Clusters from v1.11.7 to v1.13

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:  ap-southeast-1a
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxivol
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-evs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      disk-mode: SCSI
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Table 18-11 Key parameters

Parameter	Description
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	EVS volume capacity in the unit of Gi.
storageClassName	EVS disk type. Supported values: High I/O (SAS) and Ultra-high I/O (SSD)

Parameter	Description
driver	Storage driver. For EVS disks, set this parameter to **huawei.com/fuxivol**.
volumeID	Volume ID of the EVS disk. To obtain the volume ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **EVS** tab page, and copy the PVC ID on the PVC details page.
disk-mode	Device type of the EVS disk. The value is **VBD** or **SCSI**. For CCE clusters earlier than v1.11.7, you do not need to set this field. The value defaults to **VBD**. This field is mandatory for CCE clusters from v1.11.7 to v1.13 that use Linux x86. As the EVS volumes dynamically provisioned by a PVC are created from SCSI EVS disks, you are advised to choose **SCSI** when manually creating volumes (static PVs). Volumes in the VBD mode can still be used after cluster upgrades.
spec.claimRef.apiVersion	The value is fixed at **v1**.
spec.claimRef.kind	The value is fixed at **PersistentVolumeClaim**.
spec.claimRef.name	PVC name. The value is the same as the name of the PVC created in the next step.
spec.claimRef.namespace	Namespace of the PVC. The value is the same as the namespace of the PVC created in the next step.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
```



```
    storage: 10Gi
volumeName: pv-evs-example
```

Table 18-12 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Storage class, which must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	The field must be set to **flexvolume-huawei.com/fuxivol**.
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

Clusters from v1.11 to v1.11.7

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Table 18-13 Key parameters

Parameter	Description
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.

Parameter	Description
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	EVS volume capacity in the unit of Gi.
storageClassName	EVS disk type. Supported values: High I/O (SAS) and Ultra-high I/O (SSD)
driver	Storage driver. For EVS disks, set this parameter to **huawei.com/fuxivol**.
volumeID	Volume ID of the EVS disk. To obtain the volume ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **EVS** tab page, and copy the PVC ID on the PVC details page.
disk-mode	Device type of the EVS disk. The value is **VBD** or **SCSI**. For CCE clusters earlier than v1.11.7, you do not need to set this field. The default value is **VBD**. This field is mandatory for CCE clusters from v1.11.7 to v1.13 that use Linux x86. As the EVS volumes dynamically provisioned by a PVC are created from SCSI EVS disks, you are advised to choose **SCSI** when manually creating volumes (static PVs). Volumes in the VBD mode can still be used after cluster upgrades.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
```


Table 18-14 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Storage class. The value can be **sas** or **ssd**. The value must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	The field must be set to **flexvolume-huawei.com/fuxivol**.
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

Clusters of v1.9

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:
  name: pv-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      kubernetes.io/namespace: default
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sas
```

Table 18-15 Key parameters

Parameter	Description
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.

Parameter	Description
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	EVS volume capacity in the unit of Gi.
storageClassName	EVS disk type. Supported values: High I/O (SAS) and Ultra-high I/O (SSD)
driver	Storage driver. For EVS disks, set this parameter to **huawei.com/fuxivol**.
volumeID	Volume ID of the EVS disk. To obtain the volume ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **EVS** tab page, and copy the PVC ID on the PVC details page.
disk-mode	Device type of the EVS disk. The value is **VBD** or **SCSI**. For CCE clusters earlier than v1.11.7, you do not need to set this field. The default value is **VBD**. This field is mandatory for CCE clusters from v1.11.7 to v1.13 that use Linux x86. As the EVS volumes dynamically provisioned by a PVC are created from SCSI EVS disks, you are advised to choose **SCSI** when manually creating volumes (static PVs). Volumes in the VBD mode can still be used after cluster upgrades.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sas
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1
    failure-domain.beta.kubernetes.io/zone:
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
  volumeNamespace: default
```


Table 18-16 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Storage class, which must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	The field must be set to **flexvolume-huawei.com/fuxivol**.
failure-domain.beta.kubernetes.io/region	Region where the cluster is located.
failure-domain.beta.kubernetes.io/zone	AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
storage	Requested capacity in the PVC, in Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

Step 4 Create a PV.

kubectl create -f pv-evs-example.yaml

Step 5 Create a PVC.

kubectl create -f pvc-evs-example.yaml

After the operation is successful, choose **Resource Management** > **Storage** to view the created PVC. You can also view the EVS disk by name on the EVS console.

Step 6 (Optional) Add the metadata associated with the cluster to ensure that the EVS disk associated with the mounted static PV is not deleted when the node or cluster is deleted.

CAUTION

If you skip this step in this example or when creating a static PV or PVC, ensure that the EVS disk associated with the static PV has been unbound from the node before you delete the node.

1. Obtain the tenant token. For details, see **Obtaining a User Token**.
2. Obtain the EVS access address **EVS_ENDPOINT**. For details, see **Regions and Endpoints**.
3. Add the metadata associated with the cluster to the EVS disk backing the static PV.

```
curl -X POST ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --insecure \
  -d '{"metadata":{"cluster_id": "${cluster_id}", "namespace": "${pvc_namespace}"}}' \
  -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' \
  -H 'X-Auth-Token:${TOKEN}'
```


Table 18-17 Key parameters

Parameter	Description
EVS_ENDPOINT	EVS access address. Set this parameter to the value obtained in **Step 6.2**.
project_id	Project ID.
volume_id	ID of the associated EVS disk. Set this parameter to **volume_id** of the static PV to be created. You can also log in to the EVS console, click the name of the EVS disk to be imported, and obtain the ID from **Summary** on the disk details page, as shown in **Figure 18-2**.
cluster_id	ID of the cluster where the EVS PV is to be created. On the CCE console, choose **Resource Management** > **Clusters**. Click the name of the cluster to be associated. On the cluster details page, obtain the cluster ID, as shown in **Figure 18-3**.
pvc_namespace	Namespace where the PVC is to be bound.
TOKEN	User token. Set this parameter to the value obtained in **Step 6.1**.

Figure 18-2 Obtaining the disk ID

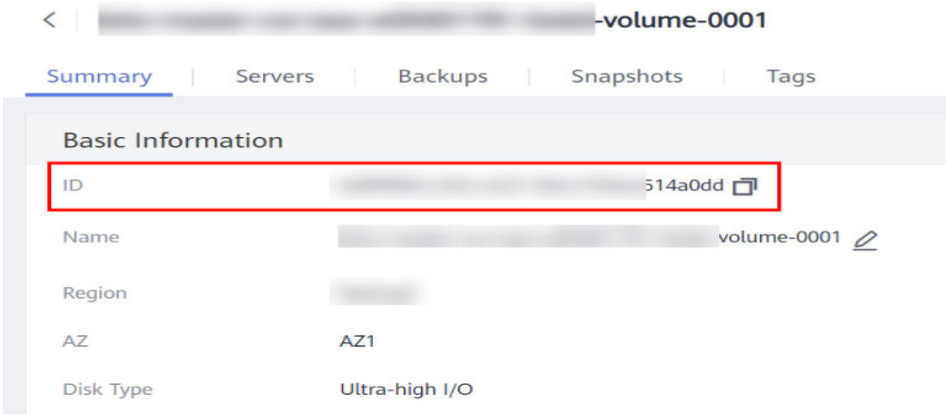
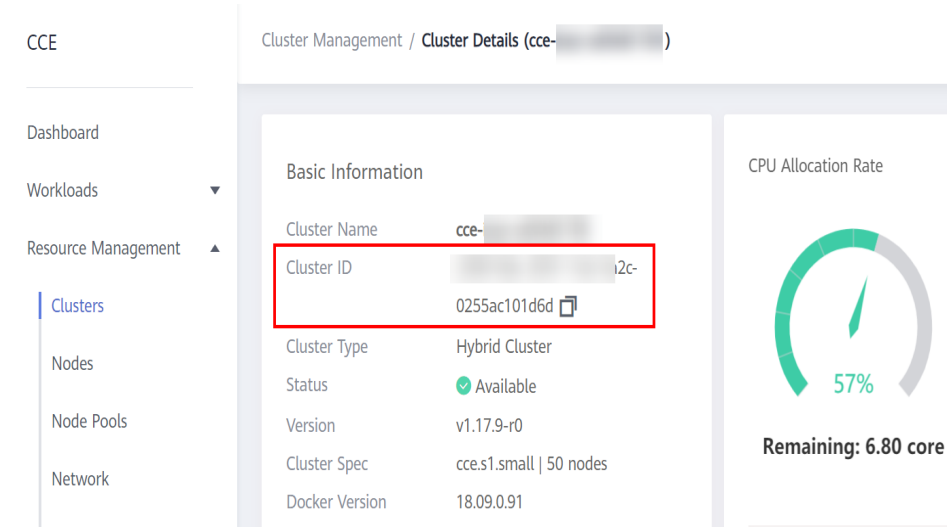


Figure 18-3 Obtaining the cluster ID



For example, run the following commands:

```
curl -X POST https://evs.ap-
southeast-1.myhuaweicloud.com:443/v2/060576866680d5762f52c0150e726aa7/volumes/
69c9619d-174c-4c41-837e-31b892604e14/metadata --insecure \
   -d '{"metadata":{"cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442", "namespace":
"default"}}' \
   -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' \
   -H 'X-Auth-Token:MIIPe******IsIm1ldG
```

After the request is executed, run the following commands to check whether the EVS disk has been associated with the metadata of the cluster:

```
curl -X GET ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --insecure \
   -H 'X-Auth-Token:${TOKEN}'
```

For example, run the following commands:

```
curl -X GET https://evs.ap-southeast-1.myhuaweicloud.com/v2/060576866680d5762f52c0150e726aa7/
volumes/69c9619d-174c-4c41-837e-31b892604e14/metadata --insecure \
   -H 'X-Auth-Token:MIIPeAYJ***9t1c31ASaQ=='
```

The command output displays the current metadata of the EVS disk.

```
{
   "metadata": {
      "namespace": "default",
      "cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442",
      "hw:passthrough": "true"
   }
}
```

----End

18.3.4 (kubectl) Creating a Pod Mounted with an EVS Volume

Scenarios

After an EVS volume is created or imported to CCE, you can mount it to a workload.

NOTICE

EVS volumes cannot be mounted across AZs. Before mounting a volume, you can run the **kubectl get pvc** command to obtain the available PVCs in the AZ where the current cluster is located.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **evs-deployment-example.yaml** file, which is used to create a Deployment.

touch evs-deployment-example.yaml

vi evs-deployment-example.yaml

Example of mounting an EVS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: evs-deployment-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: evs-deployment-example
  template:
    metadata:
      labels:
        app: evs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-evs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
      - name: pvc-evs-example
        persistentVolumeClaim:
          claimName: pvc-evs-auto-example
```

Table 18-18 Key parameters

Parent Parameter	Parameter	Description
spec.template.spec.containers.volumeMounts	name	Name of the volume mounted to the container.

Parent Parameter	Parameter	Description
spec.template.spec.containers.volumeMounts	mountPath	Mount path of the container. In this example, the volume is mounted to the **/tmp** directory.
spec.template.spec.volumes	name	Name of the volume.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Name of an existing PVC.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Mounting an EVS volume to a StatefulSet (PVC template-based, non-shared volume):

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-evs-sas-in
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-evs-sata-in
  template:
    metadata:
      labels:
        app: deploy-evs-sata-in
        failure-domain.beta.kubernetes.io/region: ap-southeast-1
        failure-domain.beta.kubernetes.io/zone: ap-southeast-1a
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          volumeMounts:
            - name: bs-sas-mountoptionpvc
              mountPath: /tmp
      imagePullSecrets:
        - name: default-secret
  volumeClaimTemplates:
    - metadata:
        name: bs-sas-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: sas
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
  serviceName: wwww
```


Table 18-19 Key parameters

Parent Parameter	Parameter	Description
metadata	name	Name of the created workload.
spec.template.spec.containers	image	Image of the workload.
spec.template.spec.containers.volumeMount	mountPath	Mount path of the container. In this example, the volume is mounted to the **/tmp** directory.
spec	serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 3 Run the following command to create the pod:

kubectl create -f evs-deployment-example.yaml

After the creation is complete, log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage** > **EVS**. Then, click the PVC name. On the PVC details page, you can view the binding relationship between the EVS volume and the PVC.

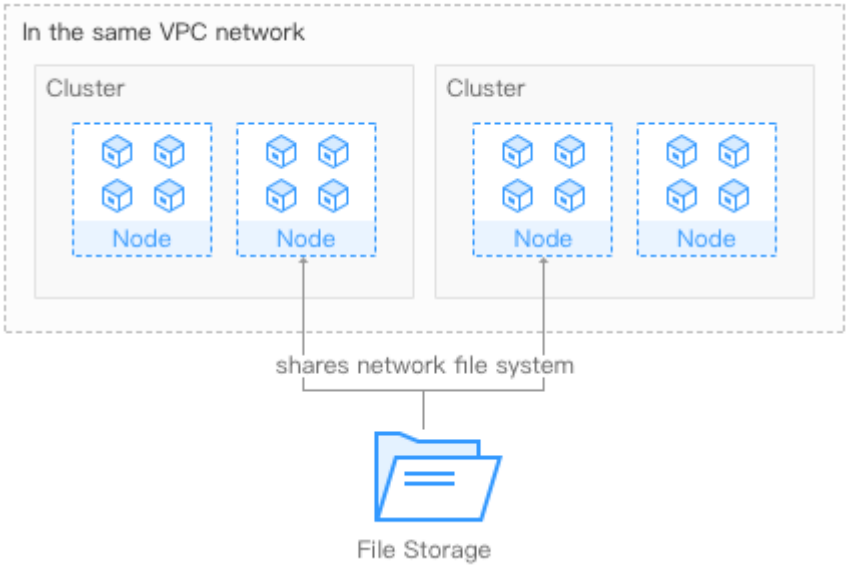
----End

18.4 Using SFS Turbo File Systems as Storage Volumes

18.4.1 Overview

CCE allows you to mount a volume created from an SFS Turbo file system to a container to store data persistently. Provisioned on demand and fast, SFS Turbo is suitable for DevOps, container microservices, and enterprise OA scenarios.

Figure 18-4 Mounting SFS Turbo volumes to CCE



Description

- **Standard file protocols**: You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing**: The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network**: User can access data only in private networks of data centers.
- **Data isolation**: The on-cloud storage service provides exclusive cloud file storage, which delivers data isolation and ensures IOPS performance.
- **Use cases**: Deployments/StatefulSets in the ReadWriteMany mode, DaemonSets, and jobs created for high-traffic websites, log storage, DevOps, and enterprise OA applications

18.4.2 (kubectl) Creating a PV from an Existing SFS Turbo File System

Scenario

CCE allows you to use an existing SFS Turbo file system to create a PersistentVolume (PV). After the creation is successful, you can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Log in to the SFS console, create a file system, and record the file system ID, shared path, and capacity.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create two YAML files for creating the PV and PVC. Assume that the file names are **pv-efs-example.yaml** and **pvc-efs-example.yaml**.

touch pv-efs-example.yaml pvc-efs-example.yaml

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiefs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 100Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-efs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxiefs
    fsType: efs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your SFS Turbo file.
      fsType: efs
      volumeID: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: efs-standard
```

Table 18-20 Key parameters

Parameter	Description
driver	Storage driver used to mount the volume. Set it to **huawei.com/fuxiefs**.
deviceMountPath	Shared path of the SFS Turbo volume.
volumeID	SFS Turbo volume ID. To obtain the ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **SFS Turbo** tab page, and copy the PVC ID on the PVC details page.
storage	File system size.

Parameter	Description
storageClassName	Volume type supported by SFS Turbo. The value can be **efs-standard** and **efs-performance**. Currently, SFS Turbo does not support dynamic creation; therefore, this parameter is not used for now.
spec.claimRef.apiVersion	The value is fixed at **v1**.
spec.claimRef.kind	The value is fixed at **PersistentVolumeClaim**.
spec.claimRef.name	The value is the same as the name of the PVC created in the next step.
spec.claimRef.namespace	The value is the same as the namespace of the PVC created in the next step.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: efs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiefs
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  volumeName: pv-efs-example
```

Table 18-21 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Read/write mode supported by SFS Turbo. The value can be **efs-standard** or **efs-performance**. The value must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	The field must be set to **flexvolume-huawei.com/fuxiefs**.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

NOTE

The VPC to which the SFS Turbo file system belongs must be the same as the VPC of the ECS VM planned for the workload. Ports 111, 445, 2049, 2051, and 20048 must be enabled in the security groups.

Step 4 Create the PV.

kubectl create -f pv-efs-example.yaml

Step 5 Create the PVC.

kubectl create -f pvc-efs-example.yaml

----End

18.4.3 (kubectl) Creating a Deployment Mounted with an SFS Turbo Volume

Scenario

After an SFS Turbo volume is created or imported to CCE, you can mount the volume to a workload.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **efs-deployment-example.yaml** file, which is used to create a Deployment:

touch efs-deployment-example.yaml

vi efs-deployment-example.yaml

Example of mounting an SFS Turbo volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: efs-deployment-example                  # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-deployment-example
  template:
    metadata:
      labels:
        app: efs-deployment-example
    spec:
      containers:
```



```
- image: nginx
  name: container-0
  volumeMounts:
  - mountPath: /tmp                          # Mount path
    name: pvc-efs-example
restartPolicy: Always
imagePullSecrets:
  - name: default-secret
volumes:
- name: pvc-efs-example
  persistentVolumeClaim:
    claimName: pvc-sfs-auto-example              # PVC name
```

Table 18-22 Key parameters

Parameter	Description
name	Name of the created Deployment.
app	Name of the application running in the Deployment.
mountPath	Mount path in the container. In this example, the mount path is **/tmp**.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Step 3 Run the following command to create the pod:

kubectl create -f efs-deployment-example.yaml

After the creation is complete, choose **Storage** > **SFS Turbo** on the CCE console and click the PVC name. On the PVC details page, you can view the binding relationship between SFS Turbo and PVC.

----End

18.4.4 (kubectl) Creating a StatefulSet Mounted with an SFS Turbo Volume

Scenario

CCE allows you to use an existing SFS Turbo volume to create a StatefulSet.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Create an SFS Turbo volume and record the volume name.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create a YAML file for creating the workload. Assume that the file name is **efs-statefulset-example.yaml**.

touch efs-statefulset-example.yaml

vi efs-statefulset-example.yaml

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: efs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-statefulset-example
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
      labels:
        app: efs-statefulset-example
    spec:
      containers:
        - image: 'nginx:1.0.0'
          name: container-0
          resources:
            requests: {}
            limits: {}
          env:
            - name: PAAS_APP_NAME
              value: efs-statefulset-example
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: b18296881cc34f929baa8b9e95abf88b
          volumeMounts:
            - name: efs-statefulset-example
              mountPath: /tmp
              readOnly: false
              subPath: ''
      imagePullSecrets:
        - name: default-secret
      terminationGracePeriodSeconds: 30
      volumes:
        - persistentVolumeClaim:
            claimName: cce-efs-import-jnr481gm-3y5o
          name: efs-statefulset-example
      affinity: {}
      tolerations:
        - key: node.kubernetes.io/not-ready
          operator: Exists
          effect: NoExecute
          tolerationSeconds: 300
        - key: node.kubernetes.io/unreachable
          operator: Exists
          effect: NoExecute
          tolerationSeconds: 300
  podManagementPolicy: OrderedReady
  serviceName: test
  updateStrategy:
    type: RollingUpdate
```


Table 18-23 Key parameters

Parameter	Description
replicas	Number of pods.
name	Name of the created workload.
image	Image used by the workload.
mountPath	Mount path in the container.
serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
claimName	Name of an existing PVC.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Step 4 Create the StatefulSet.

kubectl create -f efs-statefulset-example.yaml

----End

18.5 Using OBS Buckets as Storage Volumes

18.5.1 Overview

CCE allows you to mount a volume created from an Object Storage Service (OBS) bucket to a container to store data persistently. Object storage is commonly used in cloud workloads, data analysis, content analysis, and hotspot objects.

Figure 18-5 Mounting OBS volumes to CCE

Notes and Constraints

Secure containers do not support OBS volumes.

A single user can create a maximum of 100 OBS buckets on the console. If you have a large number of CCE workloads and you want to mount an OBS bucket to every workload, you may easily run out of buckets. In this scenario, you are advised to use OBS through the OBS API or SDK and do not mount OBS buckets to the workload on the console.

Storage Class

Object storage offers three storage classes, Standard, Infrequent Access, and Archive, to satisfy different requirements for storage performance and costs.

- The Standard storage class features low access latency and high throughput. It is therefore applicable to storing a large number of hot files (frequently accessed every month) or small files (less than 1 MB). The application scenarios include big data analytics, mobile apps, hot videos, and picture processing on social media.
- The Infrequent Access storage class is ideal for storing data that is semi-frequently accessed (less than 12 times a year), with requirements for quick response. The application scenarios include file synchronization or sharing, and enterprise-level backup. It provides the same durability, access latency, and throughput as the Standard storage class but at a lower cost. However, the Infrequent Access storage class has lower availability than the Standard storage class.
- The Archive storage class is suitable for archiving data that is rarely-accessed (averagely once a year). The application scenarios include data archiving and long-term data backup. The Archive storage class is secure and durable at an affordable low cost, which can be used to replace tape libraries. However, it may take hours to restore data from the Archive storage class.

Description

- **Standard APIs**: With HTTP RESTful APIs, OBS allows you to use client tools or third-party tools to access object storage.
- **Data sharing**: Servers, embedded devices, and IoT devices can use the same path to access shared object data in OBS.
- **Public/Private networks**: OBS allows data to be accessed from public networks to meet Internet application requirements.
- **Capacity and performance**: No capacity limit; high performance (read/write I/O latency within 10 ms).
- **Use cases**: Deployments/StatefulSets in the ReadOnlyMany mode and jobs created for big data analysis, static website hosting, online video on demand (VoD), gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks). You can create object storage by using the OBS console, tools, and SDKs.

Reference

CCE clusters can also be mounted with OBS buckets of third-party tenants, including OBS parallel file systems (preferred) and OBS object buckets. For details, see **Mounting an Object Storage Bucket of a Third-Party Tenant**.

18.5.2 Automatically Creating an OBS Volume Through kubectl

Scenario

During the use of OBS, expected OBS buckets can be automatically created and mounted as volumes. Currently, standard and infrequent access OBS buckets are supported, which correspond to **obs-standard** and **obs-standard-ia**, respectively.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **pvc-obs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-obs-auto-example.yaml

vi pvc-obs-auto-example.yaml

Example YAML:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
```



```
    volume.beta.kubernetes.io/storage-class: obs-standard  # OBS bucket type. The value can be obs-standard (standard) or obs-standard-ia (infrequent access).
  name: pvc-obs-auto-example  # PVC name
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi   # Storage capacity in the unit of Gi. For OBS buckets, this parameter is used only for verification (fixed to 1, cannot be empty or 0). Any value you set does not take effect for OBS buckets.
```

Table 18-24 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Bucket type. Currently, **obs-standard** and **obs-standard-ia** are supported.
name	Name of the PVC to be created.
accessModes	Only **ReadWriteMany** is supported. **ReadWriteOnce** is not supported.
storage	Storage capacity in the unit of Gi. For OBS buckets, this field is used only for verification (cannot be empty or 0). Its value is fixed at **1**, and any value you set does not take effect for OBS buckets.

Step 3 Run the following command to create a PVC:

kubectl create -f pvc-obs-auto-example.yaml

After the command is executed, an OBS bucket is created in the VPC to which the cluster belongs. You can click the bucket name in **Storage** > **OBS** to view the bucket or view it on the OBS console.

----End

18.5.3 (kubectl) Creating a PV from an Existing OBS Bucket

Scenario

CCE allows you to use an existing OBS bucket to create a PersistentVolume (PV). You can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create two YAML files for creating the PV and PVC. Assume that the file names are **pv-obs-example.yaml** and **pvc-obs-example.yaml**.

touch pv-obs-example.yaml pvc-obs-example.yaml

Kubernetes Cluster Version	Description	YAML Example
1.11 ≤ K8s version ≤ 1.13	Clusters from v1.11 to v1.13	**Example YAML**
K8s version = 1.9	Clusters of v1.9	**Example YAML**

Clusters from v1.11 to v1.13

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiobs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-obs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      region: ap-southeast-1
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

Table 18-25 Key parameters

Parameter	Description
driver	Storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
storage_class	Storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).
region	Region where the cluster is located.

Parameter	Description
volumeID	OBS bucket name. To obtain the name, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **OBS** tab page, and copy the PV name on the **PV Details** tab page.
storage	Storage capacity, in Gi. The value is fixed at **1Gi**.
storageClassName	Storage class supported by OBS, including **obs-standard** (standard bucket) and **obs-standard-ia** (infrequent access bucket).
spec.claimRef.apiVersion	The value is fixed at **v1**.
spec.claimRef.kind	The value is fixed at **PersistentVolumeClaim**.
spec.claimRef.name	The value is the same as the name of the PVC created in the next step.
spec.claimRef.namespace	The value is the same as the namespace of the PVC created in the next step.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
```

Table 18-26 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Storage class supported by OBS, including **obs-standard** and **obs-standard-ia**.
volume.beta.kubernetes.io/storage-provisioner	Must be set to **flexvolume-huawei.com/fuxiobs**.
volumeName	Name of the PV.
storage	Storage capacity, in Gi. The value is fixed at **1Gi**.

Clusters of v1.9

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      kubernetes.io/namespace: default
      region: ap-southeast-1
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

Table 18-27 Key parameters

Parameter	Description
driver	Storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
storage_class	Storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).
region	Region where the cluster is located.
volumeID	OBS bucket name. To obtain the name, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **OBS** tab page, and copy the PV name on the **PV Details** tab page.
storage	Storage capacity, in Gi. The value is fixed at **1Gi**.
storageClassName	Storage class supported by OBS, including **obs-standard** (standard bucket) and **obs-standard-ia** (infrequent access bucket).

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```



```
volumeName: pv-obs-example
volumeNamespace: default
```

Table 18-28 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Storage class supported by OBS, including **obs-standard** and **obs-standard-ia**.
volume.beta.kubernetes.io/storage-provisioner	Must be set to **flexvolume-huawei.com/fuxiobs**.
volumeName	Name of the PV.
storage	Storage capacity, in Gi. The value is fixed at **1Gi**.

Step 4 Create a PV.

kubectl create -f pv-obs-example.yaml

Step 5 Create a PVC.

kubectl create -f pvc-obs-example.yaml

----End

18.5.4 (kubectl) Creating a Deployment Mounted with an OBS Volume

Scenario

After an OBS volume is created or imported to CCE, you can mount the volume to a workload.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **obs-deployment-example.yaml** file, which is used to create a pod.

touch obs-deployment-example.yaml

vi obs-deployment-example.yaml

Example of mounting an OBS volume to a Deployment (PVC-based, shared volume):


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: obs-deployment-example                  # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-deployment-example
  template:
    metadata:
      labels:
        app: obs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                       # Mount path
          name: pvc-obs-example
      restartPolicy: Always
      imagePullSecrets:
        - name: default-secret
      volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: pvc-obs-auto-example       # PVC name
```

Table 18-29 Key parameters

Parameter	Description
name	Name of the pod to be created.
app	Name of the application running in the pod.
mountPath	Mount path in the container.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Example of mounting an OBS volume to a StatefulSet (PVC template-based, dedicated volume):

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-obs-standard-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-obs-standard-in
  template:
```



```
  metadata:
    labels:
      app: deploy-obs-standard-in
    annotations:
      metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
      pod.alpha.kubernetes.io/initialized: 'true'
  spec:
    containers:
      - name: container-0
        image: 'nginx:1.12-alpine-perl'
        env:
          - name: PAAS_APP_NAME
            value: deploy-obs-standard-in
          - name: PAAS_NAMESPACE
            value: default
          - name: PAAS_PROJECT_ID
            value: a2cd8e998dca42e98a41f596c636dbda
        resources: {}
        volumeMounts:
          - name: obs-bs-standard-mountoptionpvc
            mountPath: /tmp
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        imagePullPolicy: IfNotPresent
    restartPolicy: Always
    terminationGracePeriodSeconds: 30
    dnsPolicy: ClusterFirst
    securityContext: {}
    imagePullSecrets:
      - name: default-secret
    affinity: {}
    schedulerName: default-scheduler
volumeClaimTemplates:
  - metadata:
      name: obs-bs-standard-mountoptionpvc
      annotations:
        volume.beta.kubernetes.io/storage-class: obs-standard
        volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
    spec:
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 1Gi
serviceName: wwww
podManagementPolicy: OrderedReady
updateStrategy:
  type: RollingUpdate
revisionHistoryLimit: 10
```

Table 18-30 Key parameters

Parameter	Description
name	Name of the created workload.
image	Image of the workload.
mountPath	Mount path in the container. In this example, the volume is mounted to the **/tmp** directory.
serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 3 Run the following command to create the pod:

kubectl create -f obs-deployment-example.yaml

After the creation is complete, choose **Storage** > **OBS** on the CCE console and click the PVC name. On the PVC details page, you can view the binding relationship between the OBS service and the PVC.

----End

18.5.5 (kubectl) Creating a StatefulSet Mounted with an OBS Volume

Scenario

CCE allows you to use an existing OBS volume to create a StatefulSet through a PersistentVolumeClaim (PVC).

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Create an OBS volume by referring to **Automatically Creating an OBS Volume Through kubectl** and obtain the PVC name.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create a YAML file for creating the workload. Assume that the file name is **obs-statefulset-example.yaml**.

touch obs-statefulset-example.yaml

vi obs-statefulset-example.yaml

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: obs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
```



```
    annotations:
      metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
      pod.alpha.kubernetes.io/initialized: "true"
    creationTimestamp: null
    labels:
      app: obs-statefulset-example
  spec:
    affinity: {}
    containers:
      image: nginx:latest
      imagePullPolicy: Always
      name: container-0
      volumeMounts:
      - mountPath: /tmp
        name: pvc-obs-example
    imagePullSecrets:
    - name: default-secret
    volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: cce-obs-demo
```

Table 18-31 Key parameters

Parameter	Description
replicas	Number of pods.
name	Name of the created workload.
image	Image used by the workload.
mountPath	Mount path in the container.
serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
claimName	Name of an existing PVC.

Step 4 Create the StatefulSet.

kubectl create -f obs-statefulset-example.yaml

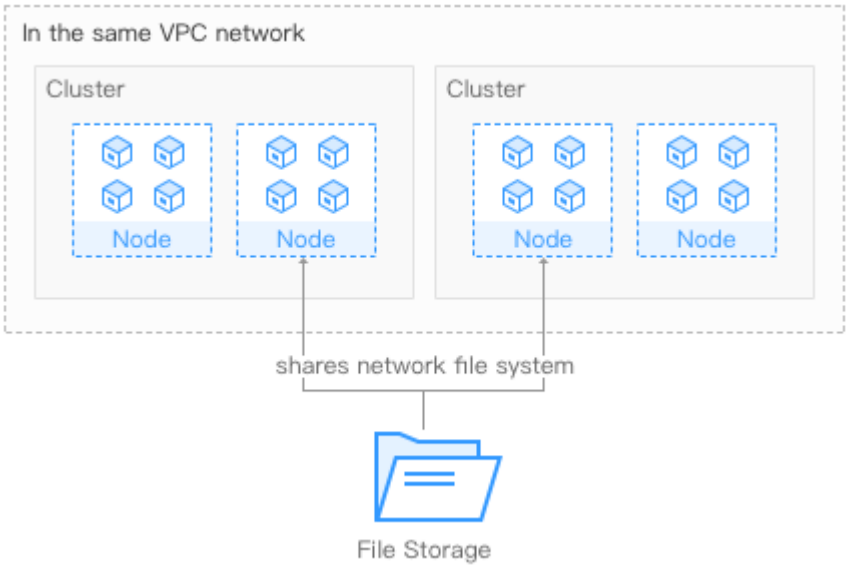
----End

18.6 Using SFS File Systems as Storage Volumes

18.6.1 Overview

CCE allows you to mount a volume created from a Scalable File Service (SFS) file system to a container to store data persistently. SFS volumes are commonly used in ReadWriteMany scenarios, such as media processing, content management, big data analysis, and workload process analysis.

Figure 18-6 Mounting SFS volumes to CCE



Description

- **Standard file protocols**: You can mount file systems as volumes to servers, the same as using local directories.
- **Data sharing**: The same file system can be mounted to multiple servers, so that data can be shared.
- **Private network**: User can access data only in private networks of data centers.
- **Capacity and performance**: The capacity of a single file system is high (PB level) and the performance is excellent (ms-level read/write I/O latency).
- **Use cases**: Deployments/StatefulSets in the ReadWriteMany mode and jobs created for high-performance computing (HPC), media processing, content management, web services, big data analysis, and workload process analysis

For details, see **SFS Service Overview**.

18.6.2 (kubectl) Automatically Creating an SFS Volume

NOTE

Currently, SFS file systems are sold out and PVCs cannot be automatically created using the storage class.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **pvc-sfs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-sfs-auto-example.yaml

vi pvc-sfs-auto-example.yaml

Example YAML file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
  name: pvc-sfs-auto-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

Table 18-32 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	File storage class. Currently, the standard file protocol type (nfs-rw) is supported.
name	Name of the PVC to be created.
accessModes	Only **ReadWriteMany** is supported. **ReadWriteOnce** is not supported.
storage	Storage capacity in Gi.

Step 3 Run the following command to create a PVC:

kubectl create -f pvc-sfs-auto-example.yaml

After the command is executed, a file system is created in the VPC to which the cluster belongs. Choose **Storage** > **SFS** on the CCE console or log in to the SFS console to view the file system.

----End

18.6.3 (kubectl) Creating a PV from an Existing SFS File System

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Log in to the SFS console, create a file system, and record the file system ID, shared path, and capacity.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create two YAML files for creating the PV and PVC. Assume that the file names are **pv-sfs-example.yaml** and **pvc-sfs-example.yaml**.

touch pv-sfs-example.yaml pvc-sfs-example.yaml

Kubernetes Cluster Version	Description	YAML Example
1.11 ≤ K8s version < 1.13	Clusters from v1.11 to v1.13	**Example YAML**
K8s version = 1.9	Clusters of v1.9	**Example YAML**

Clusters from v1.11 to v1.13

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxinfs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-sfs-example
    namespace: default
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your file.
      fsType: nfs
      volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
  persistentVolumeReclaimPolicy: Delete
  storageClassName: nfs-rw
```

Table 18-33 Key parameters

Parameter	Description
driver	Storage driver used to mount the volume. Set the driver to **huawei.com/fuxinfs** for the file system.
deviceMountPath	Shared path of the file system. On the management console, choose **Service List** > **Storage** > **Scalable File Service**. You can obtain the shared path of the file system from the **Mount Address** column, as shown in **Figure 18-7**.

Parameter	Description
volumeID	File system ID. To obtain the ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **SFS** tab page, and copy the PVC ID on the PVC details page.
storage	File system size.
storageClassName	Read/write mode supported by the file system. Currently, **nfs-rw** and **nfs-ro** are supported.
spec.claimRef.apiVersion	The value is fixed at **v1**.
spec.claimRef.kind	The value is fixed at **PersistentVolumeClaim**.
spec.claimRef.name	The value is the same as the name of the PVC created in the next step.
spec.claimRef.namespace	Namespace of the PVC. The value is the same as the namespace of the PVC created in the next step.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
```

Table 18-34 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Read/write mode supported by the file system. **nfs-rw** and **nfs-ro** are supported. The value must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	Must be set to **flexvolume-huawei.com/fuxinfs**.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

Clusters of v1.9

- **Example YAML file for the PV:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: <your_deviceMountPath>  # Shared storage path of your file.
      fsType: nfs
      kubernetes.io/namespace: default
      volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
  persistentVolumeReclaimPolicy: Delete
  storageClassName: nfs-rw
```

Table 18-35 Key parameters

Parameter	Description
driver	Storage driver used to mount the volume. Set the driver to **huawei.com/fuxinfs** for the file system.
deviceMountPath	Shared path of the file system. On the management console, choose **Service List** > **Storage** > **Scalable File Service**. You can obtain the shared path of the file system from the **Mount Address** column, as shown in **Figure 18-7**.
volumeID	File system ID. To obtain the ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name in the **SFS** tab page, and copy the PVC ID on the PVC details page.
storage	File system size.
storageClassName	Read/write mode supported by the file system. Currently, **nfs-rw** and **nfs-ro** are supported.

- **Example YAML file for the PVC:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
```

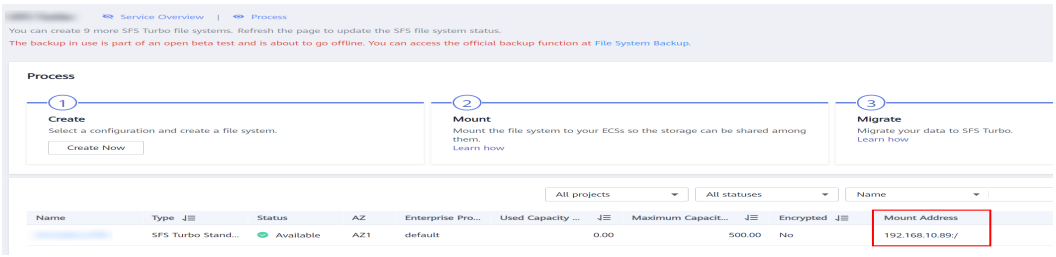


```
- ReadWriteMany
resources:
  requests:
    storage: 10Gi
volumeName: pv-sfs-example
volumeNamespace: default
```

Table 18-36 Key parameters

Parameter	Description
volume.beta.kubernetes.io/storage-class	Read/write mode supported by the file system. **nfs-rw** and **nfs-ro** are supported. The value must be the same as that of the existing PV.
volume.beta.kubernetes.io/storage-provisioner	The field must be set to **flexvolume-huawei.com/fuxinfs**.
storage	Storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
volumeName	Name of the PV.

Figure 18-7 SFS - file system mount address



NOTE

The VPC to which the file system belongs must be the same as the VPC of the ECS VM to which the workload is planned.

Step 4 Create a PV.

kubectl create -f pv-sfs-example.yaml

Step 5 Create a PVC.

kubectl create -f pvc-sfs-example.yaml

----End

18.6.4 (kubectl) Creating a Deployment Mounted with an SFS Volume

Scenario

After an SFS volume is created or imported to CCE, you can mount the volume to a workload.

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following commands to configure the **sfs-deployment-example.yaml** file, which is used to create a pod.

touch sfs-deployment-example.yaml

vi sfs-deployment-example.yaml

Example of mounting an SFS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sfs-deployment-example                  # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-deployment-example
  template:
    metadata:
      labels:
        app: sfs-deployment-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                       # Mount path
          name: pvc-sfs-example
      imagePullSecrets:
        - name: default-secret
      restartPolicy: Always
      volumes:
      - name: pvc-sfs-example
        persistentVolumeClaim:
          claimName: pvc-sfs-auto-example       # PVC name
```


Table 18-37 Key parameters

Parent Parameter	Parameter	Description
metadata	name	Name of the pod to be created.
spec.template.spec.containers.volumeMounts	mountPath	Mount path in the container. In this example, the mount path is **/tmp**.
spec.template.spec.volumes.persistentVolumeClaim	claimName	Name of an existing PVC.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Example of mounting an SFS volume to a StatefulSet (PVC template-based, dedicated volume):

NOTE

Currently, SFS file systems are sold out and cannot be exclusively used by defining the PVC template.

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-sfs-nfs-rw-in
  namespace: default
  labels:
    appgroup: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-sfs-nfs-rw-in
  template:
    metadata:
      labels:
        app: deploy-sfs-nfs-rw-in
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          volumeMounts:
            - name: bs-nfs-rw-mountoptionpvc
              mountPath: /aaa
      imagePullSecrets:
        - name: default-secret
  volumeClaimTemplates:
    - metadata:
        name: bs-nfs-rw-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: nfs-rw
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
      spec:
        accessModes:
          - ReadWriteMany
```



```
      resources:
        requests:
          storage: 1Gi
serviceName: wwww
```

Table 18-38 Key parameters

Parent Parameter	Parameter	Description
metadata	name	Name of the created workload.
spec.template.spec.containers	image	Image of the workload.
spec.template.spec.containers.volumeMount	mountPath	Mount path in the container. In this example, the mount path is **/tmp**.
spec	serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 3 Run the following command to create the pod:

kubectl create -f sfs-deployment-example.yaml

After the creation is complete, log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage** > **SFS**. Click the PVC name. On the PVC details page, you can view the binding relationship between SFS and PVC.

----End

18.6.5 (kubectl) Creating a StatefulSet Mounted with an SFS Volume

Scenario

CCE allows you to use an existing SFS volume to create a StatefulSet through a PersistentVolumeClaim (PVC).

Notes and Constraints

The following configuration example applies to clusters of Kubernetes 1.13 or earlier.

Procedure

Step 1 Create an SFS volume by referring to **(kubectl) Automatically Creating an SFS Volume** and record the volume name.

Step 2 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 3 Create a YAML file for creating the workload. Assume that the file name is **sfs-statefulset-example.yaml**.

touch sfs-statefulset-example.yaml

vi sfs-statefulset-example.yaml

Example YAML:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfs-statefulset-example
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sfs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: "true"
      labels:
        app: sfs-statefulset-example
    spec:
      affinity: {}
      containers:
      - image: nginx:latest
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-sfs-example
      imagePullSecrets:
      - name: default-secret
      volumes:
        - name: pvc-sfs-example
          persistentVolumeClaim:
            claimName: cce-sfs-demo
```

Table 18-39 Key parameters

Parent Parameter	Parameter	Description
spec	replicas	Number of pods.
metadata	name	Name of the created workload.
spec.template.spec.containers	image	Image used by the workload.
spec.template.spec.containers.volumeMounts	mountPath	Mount path in the container.
spec	serviceName	Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.

Parent Parameter	Parameter	Description
spec.template.spec.volumes.persistentVolumeClaim	claimName	Name of an existing PVC.

NOTE

spec.template.spec.containers.volumeMounts.name and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Step 4 Create the StatefulSet.

kubectl create -f sfs-statefulset-example .yaml

----End

