Cloud Container Engine

User Guide

Issue	01
Date	2024-05-30

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 Operation Guides

CCE provides different types of clusters for you to select. Create clusters on the CCE console and deploy workloads in them. This section provides console operation guides for different types of CCE clusters.

Type	Reference
Standard and Turbo clusters	• **3.2 Clusters** • **3.3 Nodes** • **3.4 Node Pools** • **3.5 Workloads** • **3.7 Network** • **3.8 Storage** • **3.9 Observability** • **3.12 Auto Scaling** For more details, see **3 User Guide for Standard and Turbo Clusters**.
Autopilot clusters	• **4.3 Clusters** • **4.4 Workloads** • **4.5 Services and Ingresses** • **4.6 Storage** • **4.7 Observability** • **4.10 Auto Scaling** For more details, see **4 User Guide for Autopilot Clusters**.

2 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-gen container cluster designed for Cloud Native 2.0, with accelerated computing, networking, and scheduling	Serverless clusters without user nodes and billed by actual CPU and memory usage In such clusters, no node deployment, management, and security maintenance is needed.
Application scenario	-	For users who expect to use container clusters to manage applications, obtain elastic computing resources, and enable simplified management on computing, 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

Category	Subcategory	CCE Standard	CCE Turbo	CCE Autopilot
Specification difference	Network model	Cloud native 1.0 network: applies to common, smaller-scale scenarios. • Tunnel network • Virtual Private Cloud (VPC) network	Cloud Native 2.0 network: applies to large-scale and high-performance scenarios. Max networking scale: 2,000 nodes	Cloud Native 2.0 network: applies to large-scale and high-performance scenarios.
	Network performance	Overlays the VPC network with the container network, causing certain performance loss.	Flattens the VPC network and container network into one, achieving zero performance loss.	Flattens the VPC network and container network into one, achieving zero performance loss.
	Network isolation	• Tunnel network model: supports network policies for intra-cluster communications. • VPC network model: supports no isolation.	Associates pods with security groups. Unifies security isolation in and out the cluster via security groups' network policies.	Associates pods with security groups. Unifies security isolation in and out the cluster via security groups' network policies.
	Security isolation	Runs common containers, isolated by cgroups.	• Physical machine: runs Kuasar containers, allowing VM-level isolation. • Runs common containers, isolated by cgroups.	VM-level isolation
	Edge infrastructure management	None	Supports management of Intelligent EdgeSite (IES).	None

3 User Guide for Standard and Turbo Clusters

3.1 High-Risk Operations and Solutions

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 3-1 High-risk operations and solutions

Category	Operation	Impact	Solution
Master node	Modifying the security group of a node in a cluster **NOTE** Naming rule of a 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 **Configuring Cluster Security Group Rules**.
	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.

Category	Operation	Impact	Solution
	Upgrading components on the master or etcd node	The cluster may be unavailable.	Roll back to the original version.
	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 **3.2.5.1 Cluster Configuration Management**.
	Replacing the master or etcd certificate	The cluster may be unavailable.	This operation cannot be undone.
Worker node	Modifying the security group of a node in a cluster **NOTE** Naming rule of a security group: *Cluster name*-**cce-node**-*Random digits*	The node may be unavailable.	Restore the security group and allow traffic from the security group to pass through. For details, see **Configuring Cluster Security Group Rules**.
	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 **3.7.5.2 DNS Configuration**.	Restore the DNS configuration based on the DNS configuration of a new node.

Category	Operation	Impact	Solution
	Deleting the node	The node will become unavailable.	This operation cannot be undone.
	Reinstalling the OS	Node components are deleted, and the node becomes unavailable.	Reset the node. For details, see **3.3.7.3 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 **3.3.7.3 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 **3.4.3.3 Configuring a Node Pool**.
	Modifying OS configuration	The node may be unavailable.	Restore the configuration items or reset the node. For details, see **3.3.7.3 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 **3.3.7.3 Resetting a Node**.

Category	Operation	Impact	Solution
	Modifying the node directory permission and the container directory permission	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 **3.3.7.3 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 **3.3.7.3 Resetting a Node**.
	Modifying NetworkManager configurations	The node will become unavailable.	Reset the node. For details, see **3.3.7.3 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.
	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 3-2 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**.

Operation	Impact	Solution
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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster** and allow traffic from the security group to pass through.
Deleting CRD resources of network-attachment-definitions 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.
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**.

Load Balancing

Table 3-3 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 3-4 High-risk operations and solutions

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 3-5 High-risk operations and solutions

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

Add-ons

Table 3-6 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.

3.2 Clusters

3.2.1 Cluster Overview

3.2.1.1 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 **3.7.2 Container Network Models**.

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.

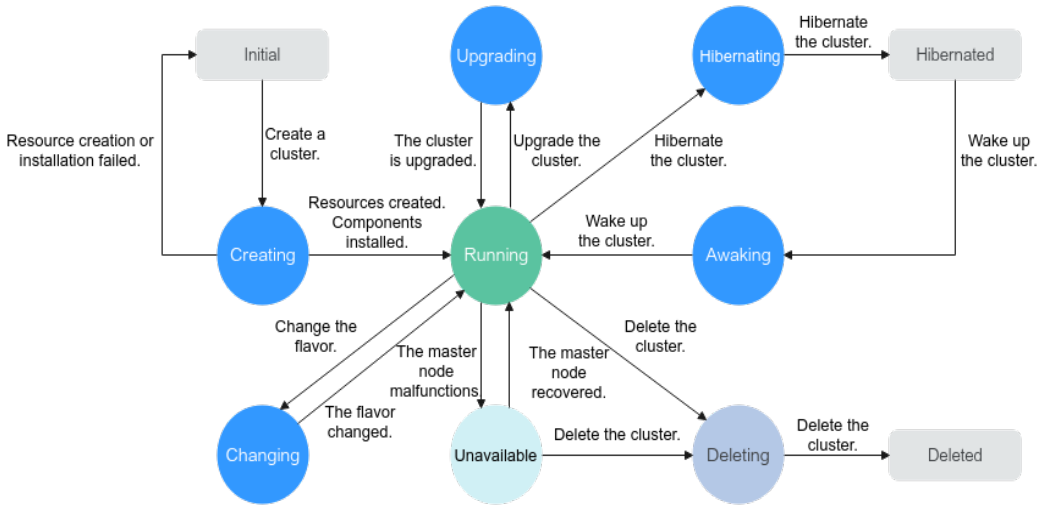
The master node specifications decide the number of nodes that can be managed by a cluster. You can select the cluster management scale, for example, 50 or 200 nodes.

Cluster Lifecycle

Table 3-7 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 3-1 Cluster status transition



3.2.1.2 Kubernetes Version Release Notes

3.2.1.2.1 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**
- **API Changes and Removals**

- **Enhanced Kubernetes 1.29 on CCE**
- **References**

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 before it starts or 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 computing 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 Common Expression Language (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**.

3.2.1.2.2 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

- 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 General Availability (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 PersistentVolume (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: **Pod replacement policy** and **Backoff limit per index**.

  - Pod replacement policy

    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 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. In this case, the backoff limit per index configuration is useful.

    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 Common Expression Language (CEL) related features are improved.

  CEL related capabilities are enhanced.

  - CEL used to validate CustomResourceDefinitions (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**.

3.2.1.2.3 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 retrained 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 General Availability (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. The 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 the **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**

3.2.1.2.4 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 **Common Expression Language (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**

3.2.1.2.5 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 1.23 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 1.22 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**

3.2.1.2.6 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 1.21 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 1.20 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. Update the **timeoutSeconds** field for the applications that use this feature during the upgrade. 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**

3.2.1.2.7 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**

3.2.1.2.8 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**

3.2.1.2.9 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

3.2.1.2.10 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 3-8 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

3.2.1.2.11 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 3-9 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.
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

3.2.1.2.12 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 3-10 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.
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.

Kubernetes (CCE Enhanced Version)	Description
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.
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

Kubernetes (CCE Enhanced Version)	Description
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.
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.

Kubernetes (CCE Enhanced Version)	Description
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.
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.

Kubernetes (CCE Enhanced Version)	Description
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.
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.

3.2.1.3 Patch Version Release Notes

Indexes

Version 1.29

Table 3-11 Release notes for the v1.29 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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 3-12 Release notes for the v1.28 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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.
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.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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
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 Kata containers in a CCE Turbo cluster. • Improved the stability when Kata containers are repeatedly created or deleted in a CCE Turbo cluster.	None

Version 1.27

NOTICE

Dockershim has been removed since Kubernetes v1.24, and Docker is not supported in v1.24 and later versions by default. Use containerd. To migrate nodes from Docker to containerd, follow the operations described in **Migrating Nodes from Docker to containerd**.

Table 3-13 Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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.
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.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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 Kata containers in a CCE Turbo cluster. • Improved the stability when Kata 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	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
			service and ingress ports, as well as the logic for checking the conflict of ingress paths in a cluster.	
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

All nodes in the CCE clusters of version 1.25, except the ones running EulerOS 2.5, use containerd by default.

Table 3-14 Release notes for the v1.25 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
v1.25.8-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.
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.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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 Kata containers in a CCE Turbo cluster. • Improved the stability when Kata 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 3-15 Release notes for the v1.23 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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.
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.

CCE Cluster Patch Version	Kubernetes Version	Feature Updates	Optimization	Vulnerability Fixing
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 Kata containers in a CCE Turbo cluster. • Improved the stability when Kata 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		• 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 Kata 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 3-16 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. • 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.	• 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		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 Kata 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 Kata 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 3-17 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	• 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.	• 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 2.0 Networks allow 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		• 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

3.2.2 Buying a Cluster

3.2.2.1 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.

Constraints

- During the node creation, software packages are downloaded from OBS using the domain name. A private DNS server must be used to resolve the OBS domain name. Therefore, the DNS server address of the subnet where the node resides must be set to the **private DNS server address** so that the node

can access the private DNS server. When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.

- You can create a maximum of 50 clusters in a single region. If more clusters are required, go to **Quotas** to increase your quota.
- 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** in the upper right corner.

----End

Step 2: Configure the Cluster

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

Basic Settings

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 (for example, **default**) is selected, the cluster, nodes in the cluster, cluster security groups, node security groups, and EIPs of the automatically created nodes will be created in this enterprise project. After the cluster is created, do not modify the enterprise projects of nodes, cluster security groups, and node security groups in the cluster. Enterprise projects facilitate project-level management and grouping of cloud resources and users. 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 newly created cluster only supports scaling out. For details, see **3.2.5.3 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. You can also select AZs for the master nodes. 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 **3.7.1 Overview**.

Table 3-18 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.
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.

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 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 **Configuring Cluster Security Group Rules**.
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**.

Table 3-19 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 **3.7.2.1 Overview**.
Container CIDR Block (configured for CCE standard clusters)	Configure the CIDR block used by containers. The value determines the maximum number of containers in your cluster. Multiple container CIDR blocks can be added to the VPC network after the cluster is created. For details, see **3.7.7.3 Adding a Container CIDR Block for a Cluster**.
Default Pod Subnet (configured for CCE Turbo clusters)	Select the subnet to which the pod belongs. If no subnet is available, click **Create Subnet** to create one. The pod subnet determines the maximum number of containers in a cluster. You can add pod subnets after a cluster is created.

Table 3-20 Service network

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 **3.2.2.2 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 applies to scenarios where the cluster scale is large or the number of Services is large.
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

Parameter	Description
Certificate Authentication	• If **Automatically generated** is selected, the X509-based authentication mode will be enabled by default. X509 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.
CPU Management	If enabled, exclusive CPU cores can be allocated to workload pods. For details, see **3.6.2.1 CPU Policy**.

Parameter	Description
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 **3.2.5.2 Cluster Overload Control**.
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. Key specifications: • Cannot be empty. Contains 1 to 128 single-byte characters. • Do not enter labels starting with _**sys**_, which are system labels. • Can contain UTF-8 letters, digits, spaces, and the following characters: _. : / = + - @ Recommended regular expression: ^((?!_sys_)[\\p{L}\\p{Z}\\p{N}_.:\\/=+\\-@]*)$ Value specifications: • Can contain up to 255 characters. • Can contain UTF-8 letters, digits, spaces, and the following characters: _. : / = + - @ Recommended regular expression: ^([\p{L}\p{Z}\p{N}_.:\/=+\-@]*)$ • The value can be empty or null. • The value of a predefined tag cannot be empty or null.
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.

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.

Add-on Name	Description
CCE Container Storage (Everest)	This add-on (**3.14.3 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 (**3.14.2 CoreDNS**) is installed by default. It provides DNS resolution for your cluster and can be used to access the in-cloud DNS server.

Observability

Add-on Name	Description
Cloud Native Cluster Monitoring	(Optional) If selected, this add-on (**3.14.17 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 **3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.
Cloud Native Logging	(Optional) If selected, this add-on (**3.14.18 Cloud Native Logging**) 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 **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.
CCE Node Problem Detector	(Optional) If selected, this add-on (**3.14.4 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.

Observability

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 **3.9.6.1 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 collecting 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 **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.
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 **3.2.3.1 Connecting to a Cluster Using kubectl**.
- Add nodes to the cluster. For details, see **3.3.4 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**.

3.2.2.2 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	iptables is a mature and stable kube-proxy mode, but its performance is average. It applies to scenarios where the number of services is small (less than 1000) or there are a large number of short concurrent connections on the client. For details, see **iptables**.	IPVS is a high-performance kube-proxy mode. It applies to scenarios where the cluster scale is large or the number of Services is large. 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.
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**.

Feature Difference	iptables	IPVS
Additional restrictions	When there are more than 1000 Services in the cluster, network delay may occur.	• If an ingress and a Service use the same load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer used by the ingress. Use different load balancers for the ingress and Service. • 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**.

3.2.3 Connecting to a Cluster

3.2.3.1 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.

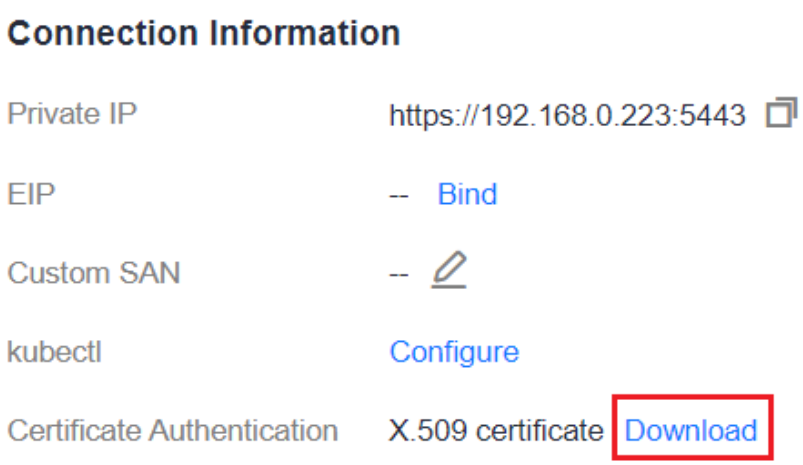
CCE allows you to access a cluster through a private network or a public network.

- 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 3-2**. 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 3-2 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. Procedure:

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.

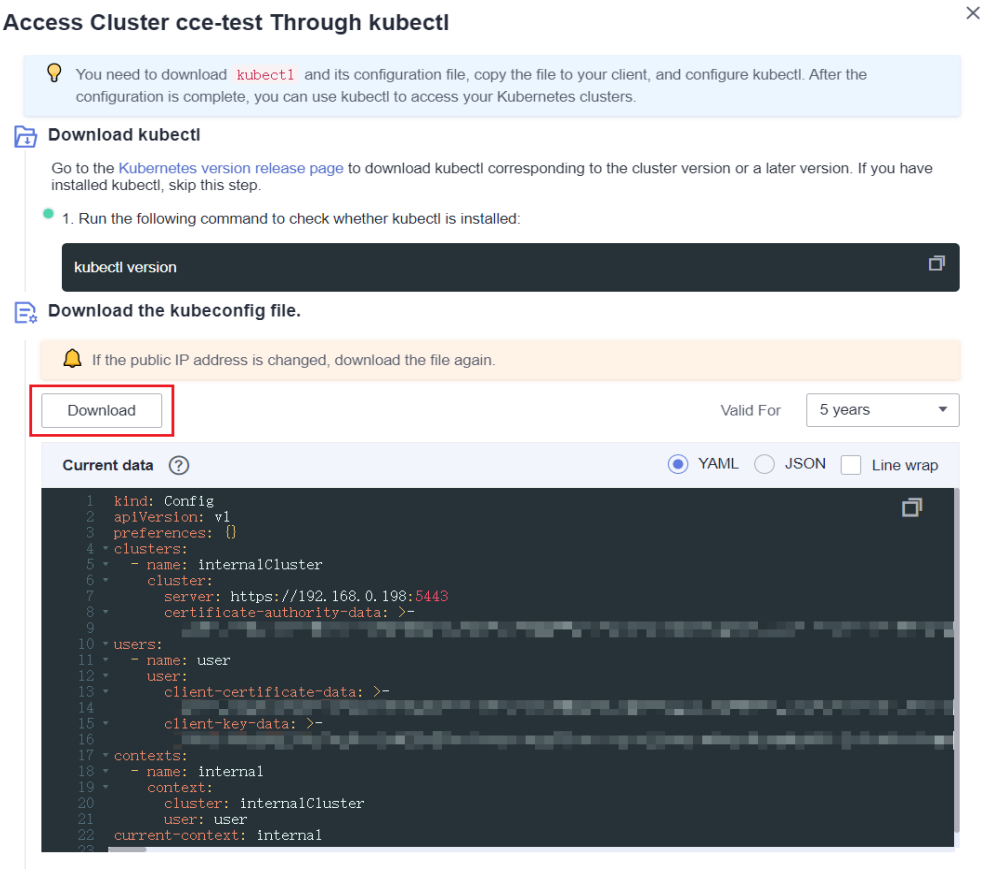
This section uses the Linux environment as an example to describe how to install and configure kubectl. For details, see **Installing kubectl**.

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 number. Replace it as required.
2. Install kubectl.
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

Step 2 Obtain the kubectl configuration file (kubeconfig).

On the **Overview** page, locate the **Connection Info** area, click **Configure** next to **kubectl**. On the page displayed, download the configuration file.

Figure 3-3 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 **kubeconfig.yaml** file 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**.

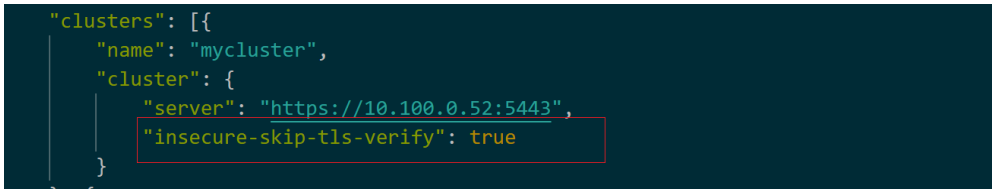
----End

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 3-4**. To use two-way authentication, download the **kubeconfig.yaml** file again and enable two-way authentication for the domain names.

Figure 3-4 Two-way authentication disabled for domain names



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 **3.16.3 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**

3.2.3.2 Connecting to a Cluster Using CloudShell

Scenario

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

Permissions

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

Using CloudShell

CloudShell is a web shell used to manage and maintain cloud resources. CCE allows you to use CloudShell to connect to clusters and use kubectl in CloudShell to access clusters (clicking the command line tool icon in **Figure 3-5**).

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.
- Currently, you can use CloudShell to log in to containers only in CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and CN North-Ulanqab1 regions.
- CloudShell does not support an account or sub-project agency.

Figure 3-5 CloudShell

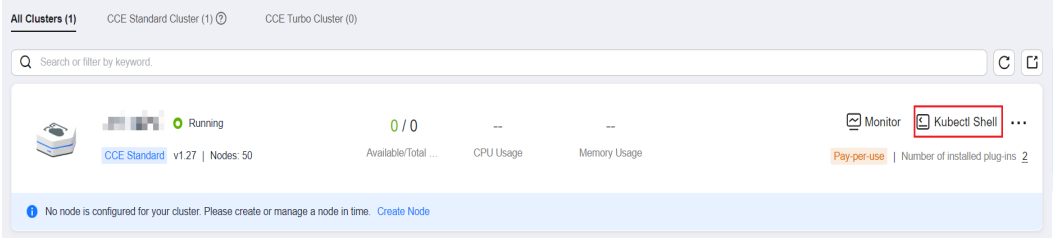
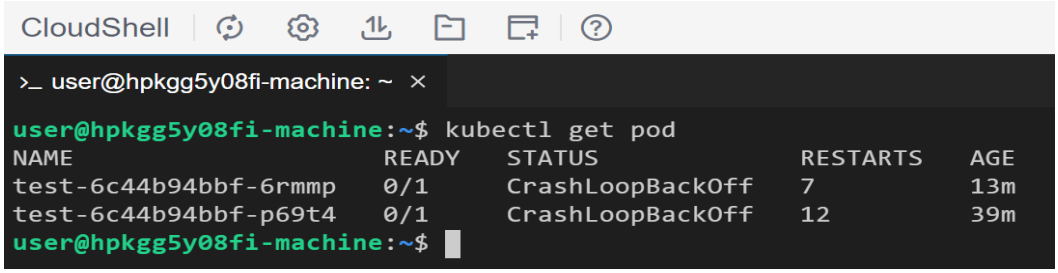


Figure 3-6 Using kubectl in CloudShell



3.2.3.3 Connecting to a Cluster Using an X.509 Certificate

Scenario

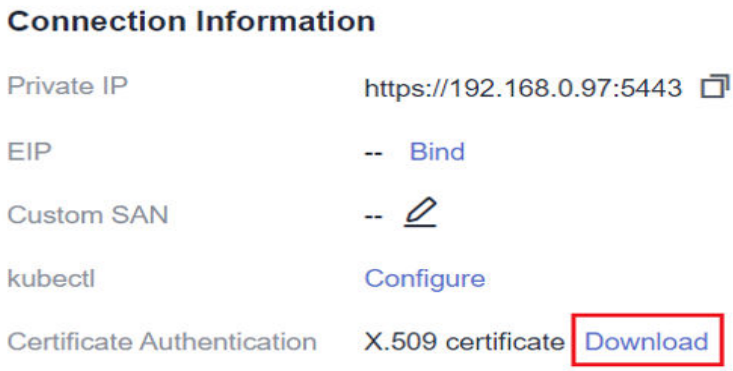
This section describes how to obtain the cluster certificate from the console and use it to access Kubernetes clusters.

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 3-7 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. In the following information, *192.168.0.18:5443* indicates the IP address of the API server in the cluster.

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

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

----End

3.2.3.4 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 response domain name mapping when specifying 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.

Constraints

This feature is available only to clusters of v1.19 and later.

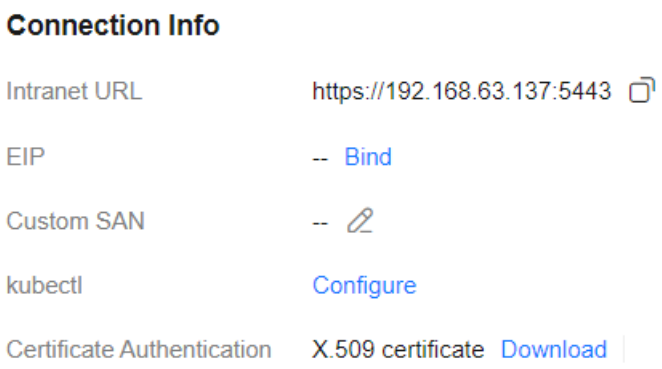
Customizing a SAN

Step 1 Log in to the CCE console.

Step 2 Click the target cluster in the cluster list to go to the cluster details 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 3-8 Custom SAN



NOTE

1. This operation will restart kube-apiserver and update the **kubeconfig.json** file for a short period of time. Do not perform operations on the cluster during this period.
2. A maximum of 128 domain names or IP addresses, separated by commas (,), are allowed.
3. If a custom domain name needs to be bound to an EIP, ensure that an EIP has been configured.

----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 X509 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 view 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

3.2.4 Upgrading a Cluster

3.2.4.1 Upgrade Overview

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:

1. 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.
2. 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**.
3. 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.
4. 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 or v1.25
v1.23	v1.25, v1.27, or v1.28
v1.25	v1.27 or v1.28
v1.27	v1.28
v1.28	N/A

  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

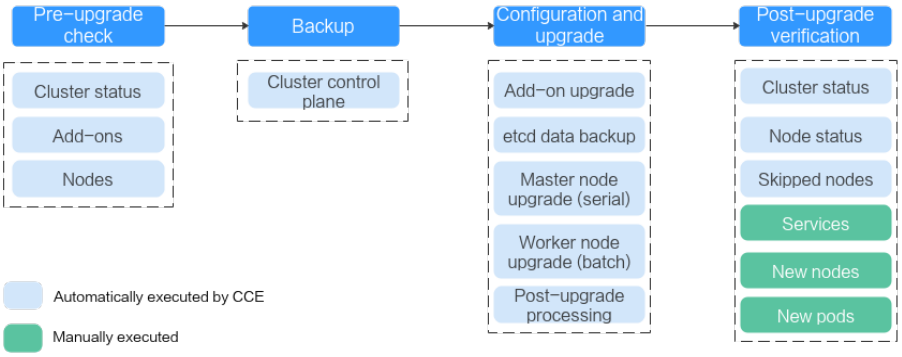
  Patch version management is available for CCE clusters of v1.19 or later to provide new features and fix bugs and vulnerability for in-maintenance clusters without requiring a major version upgrade.

  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 **3.2.1.3 Patch Version Release Notes**.

Cluster Upgrade Process

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

Figure 3-9 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, and nodes to ensure that the cluster meets the upgrade requirements. For more details, see **3.2.4.5.1 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.

Backup Type	Backup Object	Backup Mode	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.

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 60.

     - **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.

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 **3.2.4.3 Performing Post-Upgrade Verification**.

Upgrade Modes

Table 3-21 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, which 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

3.2.4.2 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 **3.2.4.1 Upgrade Overview**.

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. Do not **stop, restart, or delete nodes** during cluster upgrade. Otherwise, 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

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.
- 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 **3.3.2 Container Engine**.	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**.
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 1.19 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 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. 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 1.15 and 1.19 clusters, the Docker storage driver file system is switched from XFS to Ext4. As a result, the import package sequence in the pods of the upgraded Java application may be abnormal, causing 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 1.19 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 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. 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 1.15 to 1.19 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 1.16 and later versions, **QoS classes** are different from those in earlier versions. In kubelet 1.15 and earlier versions, only containers in **spec.containers** are counted. In kubelet 1.16 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 3-22**.
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 is allocated 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 3-22 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 old 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 v1.27**
- **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 3-23 APIs deprecated in Kubernetes v1.27

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

Table 3-24 APIs deprecated in Kubernetes v1.25

Resource Name	Deprecated API Version	Substitute API Version	Change Description
CronJob	batch/v1beta1	batch/v1 (This API is available since v1.21.)	None
EndpointSlice	discovery.k8s.io/v1beta1	discovery.k8s.io/v1 (This API is 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 Name	Deprecated API Version	Substitute API Version	Change Description
Event	events.k8s.io/v1beta1	events.k8s.io/v1 (This API is 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 Name	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 is available since v1.23.)	None
PodDisruptionBudget	policy/v1beta1	policy/v1 (This API is 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 is available since v1.20.)	None

Table 3-25 APIs deprecated in Kubernetes v1.22

Resource Name	Deprecated API Version	Substitute API Version	Change Description
MutatingWebhookConfiguration ValidatingWebhookConfiguration	admissionregistration.k8s.io/v1beta1	admissionregistration.k8s.io/v1 (This API is 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 Name	Deprecated API Version	Substitute API Version	Change Description
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	apiextensions/v1 (This API is 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 Name	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 is available since v1.10.)	None
TokenReview	authentication.k8s.io/v1beta1	authentication.k8s.io/v1 (This API is available since v1.6.)	None
LocalSubjectAccessReview SelfSubjectAccessReview SubjectAccessReview SelfSubjectRulesReview	authorization.k8s.io/v1beta1	authorization.k8s.io/v1 (This API is available since v1.16.)	**spec.group** was renamed **spec.groups** in v1 (patch **#32709**).

Resource Name	Deprecated API Version	Substitute API Version	Change Description
CertificateSigningRequest	certificates.k8s.io/v1beta1	certificates.k8s.io/v1 (This API is 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 is available since v1.14.)	None

Resource Name	Deprecated API Version	Substitute API Version	Change Description
Ingress	networking.k8s.io/v1beta1 extensions/v1beta1	networking.k8s.io/v1 (This API is 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 is available since v1.19.)	None
ClusterRole ClusterRoleBinding Role RoleBinding	rbac.authorization.k8s.io/v1beta1	rbac.authorization.k8s.io/v1 (This API is available since v1.8.)	None
PriorityClass	scheduling.k8s.io/v1beta1	scheduling.k8s.io/v1 (This API is available since v1.14.)	None

Resource Name	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 3-26 APIs deprecated in Kubernetes v1.16

Resource Name	Deprecated API Version	Substitute API Version	Change Description
NetworkPolicy	extensions/v1beta1	networking.k8s.io/v1 (This API is available since v1.8.)	None
DaemonSet	extensions/v1beta1 apps/v1beta2	apps/v1 (This API is 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 Name	Deprecated API Version	Substitute API Version	Change Description
Deployment	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API is 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 is 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 Name	Deprecated API Version	Substitute API Version	Change Description
ReplicaSet	extensions/v1beta1 apps/v1beta1 apps/v1beta2	apps/v1 (This API is 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 is 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.

3.2.4.3 Performing Post-Upgrade Verification

3.2.4.3.1 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.

3.2.4.3.2 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.

3.2.4.3.3 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.

3.2.4.3.4 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.

3.2.4.3.5 New Node Check

Check Items

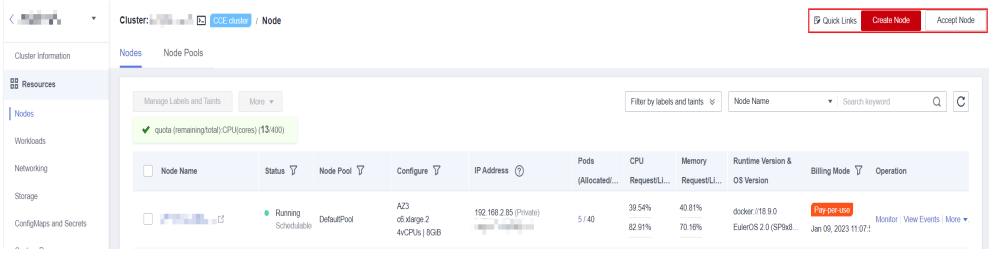
Check whether nodes can be created in the cluster.

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 3-10 Creating a node



----End

Solution

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

3.2.4.3.6 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.

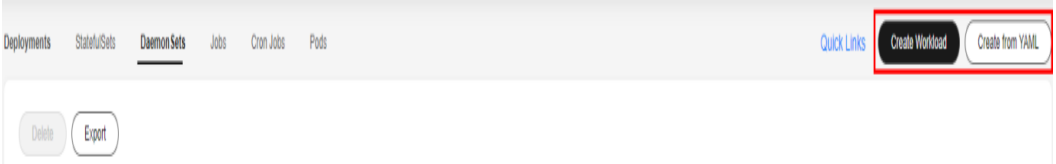
Procedure

After creating a node based on **3.2.4.3.5 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 3-11 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, YAML deploys DaemonSet in the default namespace, uses **ngxin:perl** as the base image, requests 10m vCPUs and 10 MiB memory, and limits 100 MB CPU and 50 MiB 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.

3.2.4.4 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 v1.7.* or v1.9.* to 1.17.*) and new clusters can be created for service migration.

Prerequisites

Table 3-27 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 whitelists.
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 **3.2.2.1 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 **3.8.5.2 Using an Existing SFS Turbo File System Through a Static PV**.

NOTE

Storage switching supports only OBS buckets and SFS Turbo file systems. If non-shared storage is used, suspend the workloads in the old cluster to switch the storage resources. As a result, 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 **3.8.5.2 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

3.2.4.5 Troubleshooting for Pre-upgrade Check Exceptions

3.2.4.5.1 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 3-28 Check items

No.	Check Item	Description
1	**3.2.4.5.2 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	**3.2.4.5.3 Upgrade Management**	Check whether the target cluster is under upgrade management.
3	**3.2.4.5.4 Add-ons**	• Check whether the add-on status is normal. • Check whether the add-on support the target version.
4	**3.2.4.5.5 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.

No.	Check Item	Description
5	3.2.4.5.6 SSH Connectivity of Master Nodes	Check whether CCE can connect to your master nodes.
6	3.2.4.5.8 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.
7	3.2.4.5.9 To-Be-Migrated Nodes	Check whether the node needs to be migrated.
8	3.2.4.5.10 Discarded Kubernetes Resources	Check whether there are discarded resources in the clusters.
9	3.2.4.5.11 Compatibility Risks	Read the version compatibility differences and ensure that they are not affected. The patch upgrade does not involve version compatibility differences.
10	3.2.4.5.12 CCE Agent Versions	Check whether cce-agent on the current node is of the latest version.
11	3.2.4.5.13 Node CPU Usage	Check whether the CPU usage of the node exceeds 90%.
12	3.2.4.5.14 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.
13	3.2.4.5.15 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.
14	3.2.4.5.16 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.
15	3.2.4.5.17 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**.
16	3.2.4.5.18 Kubelet	Check whether the kubelet on the node is running properly.

No.	Check Item	Description
17	3.2.4.5.19 Node Memory	Check whether the memory usage of the node exceeds 90%.
18	3.2.4.5.20 Node Clock Synchronization Server	Check whether the clock synchronization server ntpd or chronyd of the node is running properly.
19	3.2.4.5.21 Node OS	Check whether the OS kernel version of the node is supported by CCE.
20	3.2.4.5.22 Node CPUs	Check whether the number of CPUs on the master node is greater than 2.
21	3.2.4.5.23 Node Python Commands	Check whether the Python commands are available on a node.
22	3.2.4.5.24 ASM Version	• Check whether ASM is used by the cluster. • Check whether the current ASM version supports the target cluster version.
23	3.2.4.5.25 Node Readiness	Check whether the nodes in the cluster are ready.
24	3.2.4.5.26 Node journald	Check whether journald of a node is normal.
25	3.2.4.5.27 containerd.sock	Check whether the containerd.sock file exists on the node. This file affects the startup of container runtime in the Euler OS.
26	3.2.4.5.28 Internal Errors	Before the upgrade, check whether an internal error occurs.
27	3.2.4.5.29 Node Mount Points	Check whether inaccessible mount points exist on the node.
28	3.2.4.5.30 Kubernetes Node Taints	Check whether the taint needed for cluster upgrade exists on the node.
29	3.2.4.5.31 Everest Restrictions	Check whether there are any compatibility restrictions on the current Everest add-on.
30	3.2.4.5.32 cce-hpa-controller Restrictions	Check whether the current cce-controller-hpa add-on has compatibility restrictions.
31	3.2.4.5.33 Enhanced CPU Policies	Check whether the current cluster version and the target version support **enhanced CPU policy**.
32	3.2.4.5.34 Health of Worker Node Components	Check whether the container runtime and network components on the worker nodes are healthy.

No.	Check Item	Description
33	**3.2.4.5.35 Health of Master Node Components**	Check whether the Kubernetes, container runtime, and network components of the master nodes are healthy.
34	**3.2.4.5.36 Memory Resource Limit of Kubernetes Components**	Check whether the resources of Kubernetes components, such as etcd and kube-controller-manager, exceed the upper limit.
35	**3.2.4.5.37 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.
36	**3.2.4.5.38 Node NetworkManager**	Check whether NetworkManager of a node is normal.
37	**3.2.4.5.39 Node ID File**	Check the ID file format.
38	**3.2.4.5.40 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:
39	**3.2.4.5.41 Node Configuration File**	Check whether the configuration files of key components exist on the node.
40	**3.2.4.5.42 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**.
41	**3.2.4.5.43 sudo Commands of a Node**	Whether the sudo commands and sudo-related files of the node are working
42	**3.2.4.5.44 Key Commands of Nodes**	Whether some key commands that the node upgrade depends on are working

No.	Check Item	Description
43	3.2.4.5.45 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.
44	3.2.4.5.46 HTTPS Load Balancer Certificate Consistency	Check whether the certificate used by an HTTPS load balancer has been modified on ELB.
45	3.2.4.5.47 Node Mounting	Check whether the default mount directory and soft link on the node have been manually mounted or modified.
46	3.2.4.5.48 Login Permissions of User paas on a Node	Check whether user **paas** is allowed to log in to a node.
47	3.2.4.5.49 Private IPv4 Addresses of Load Balancers	Check whether the load balancer associated with a Service is allocated with a private IPv4 address.
48	3.2.4.5.50 Historical Upgrade Records	Check whether the source version of the cluster is earlier than v1.11 and the target version is later than v1.23.
49	3.2.4.5.51 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.
50	3.2.4.5.52 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.
51	3.2.4.5.53 Nodes' System Parameter Settings	Check whether the default system parameter settings on your nodes are modified.
52	3.2.4.5.54 Residual Package Versions	Check whether there are residual package version data in the current cluster.
53	3.2.4.5.55 Node Commands	Check whether the commands required for the upgrade are available on the node.
54	3.2.4.5.56 Node Swap	Check whether swap has been enabled on cluster nodes.

No.	Check Item	Description
55	3.2.4.5.57 nginx-ingress Upgrade	Check whether there are compatibility issues that may occur during nginx-ingress upgrade.
56	3.2.4.5.62 ELB Listener Access Control	Check whether the access control of the ELB listener has been configured for the Service in the current cluster using annotations and whether the configurations are correct.
57	3.2.4.5.63 Master Node Flavor	Check whether the flavor of the master nodes in the cluster is the same as the actual flavor of these nodes.
58	3.2.4.5.64 Subnet Quota of Master Nodes	Check whether the number of available IP addresses in the cluster subnet supports rolling upgrade.
59	3.2.4.5.65 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.
60	3.2.4.5.66 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.
61	3.2.4.5.67 Number of Node Images	Check the number of images on your node. If the number is greater than 1000, Docker startup may be slow.

3.2.4.5.2 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.

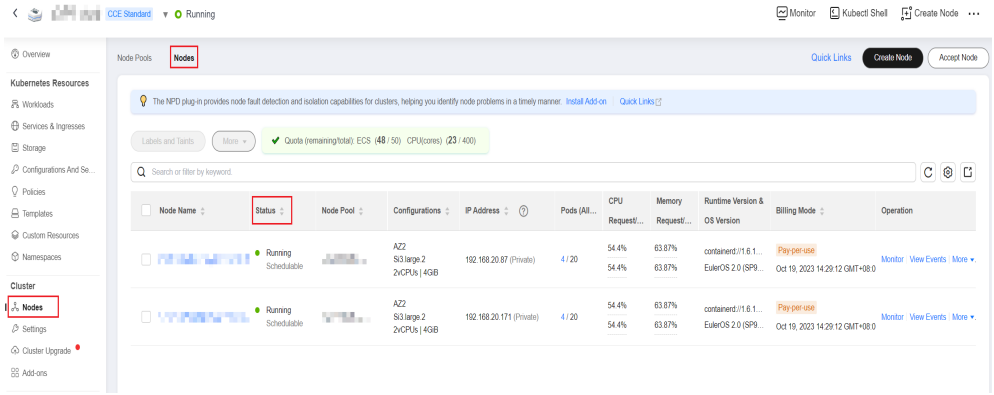
Solution

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 3-12 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 3-29 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 check result shows that the upgrade is not supported due to regional restrictions, contact technical support. **NOTE** If the target version is v1.27 or later, only Ubuntu 22.04 supports the upgrade.
Huawei Cloud EulerOS	If the check result shows that the upgrade is not supported due to regional restrictions, contact technical support.

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 Kubernetes node corresponding to the affected node does not exist.**

   It is possible that the node is being deleted. Check again later.

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 cluster to v1.25 or v1.28. If necessary, contact technical support.

3.2.4.5.3 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.

To resolve this issue, contact technical support.

3.2.4.5.4 Add-ons

Check Items

Check the following items:

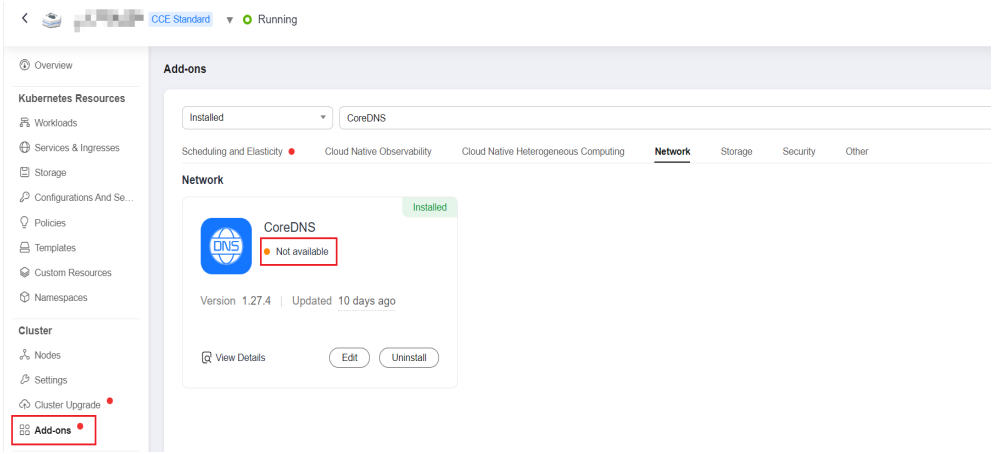
- Check whether the add-on status is normal.
- Check whether the add-on support the target version.

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.

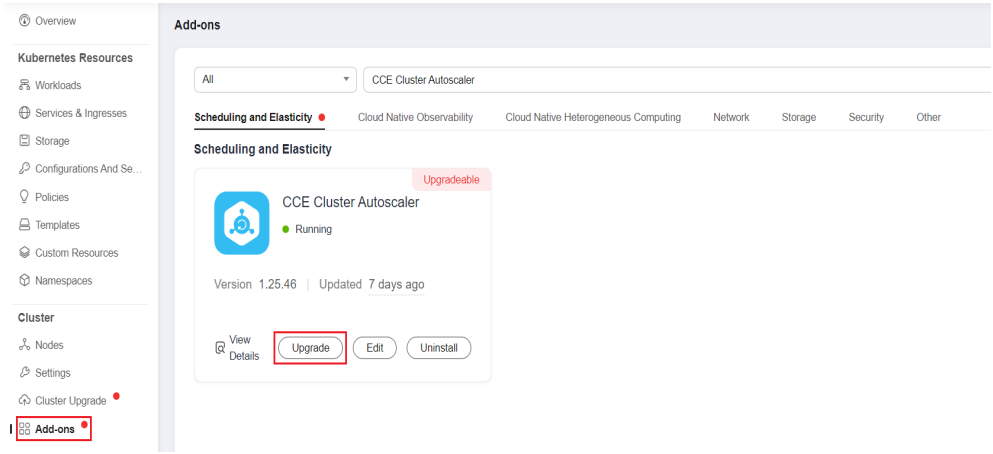
Figure 3-13 Checking add-on statuses



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

  The add-on cannot be automatically upgraded with the cluster due to compatibility issues. 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.

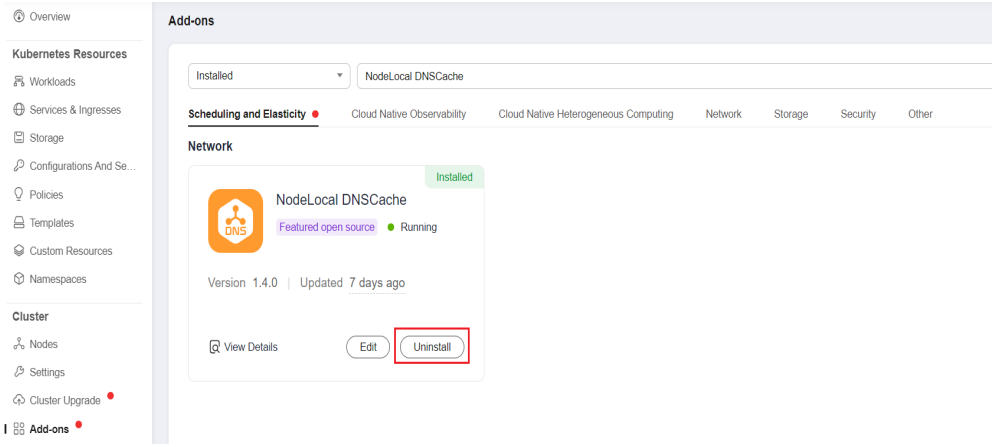
Figure 3-14 Upgrading an add-on



- **Scenario 3: After the add-on is upgraded to the latest version, it is still not supported by the target cluster version.**

  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.

Figure 3-15 Uninstalling an add-on



- **Scenario 4: 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.

3.2.4.5.5 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.

3.2.4.5.6 SSH Connectivity of Master Nodes

Check Items

Check whether CCE can connect to your master nodes.

Solution

Contact technical support.

3.2.4.5.7 Node Pools

Check Items

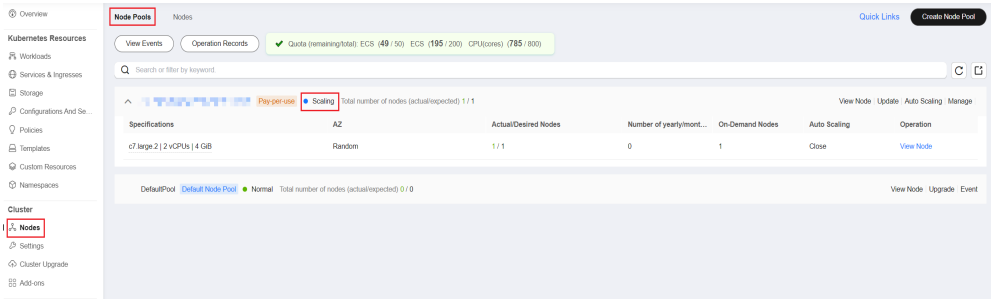
- Check the node pool status.
- Check whether the node pool OS or container runtime is supported after the upgrade.

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 view 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 3-16 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 **3.3.3 Node OS**.

  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 **3.4.3.5 Synchronizing Node Pools**.

3.2.4.5.8 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

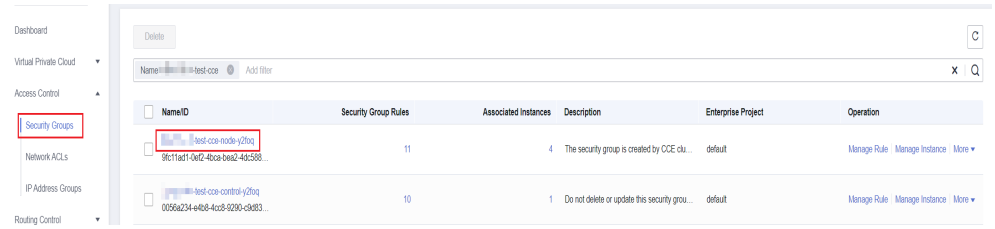
This check item is performed only for clusters using VPC networking. For clusters using other networking, skip this check item.

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:

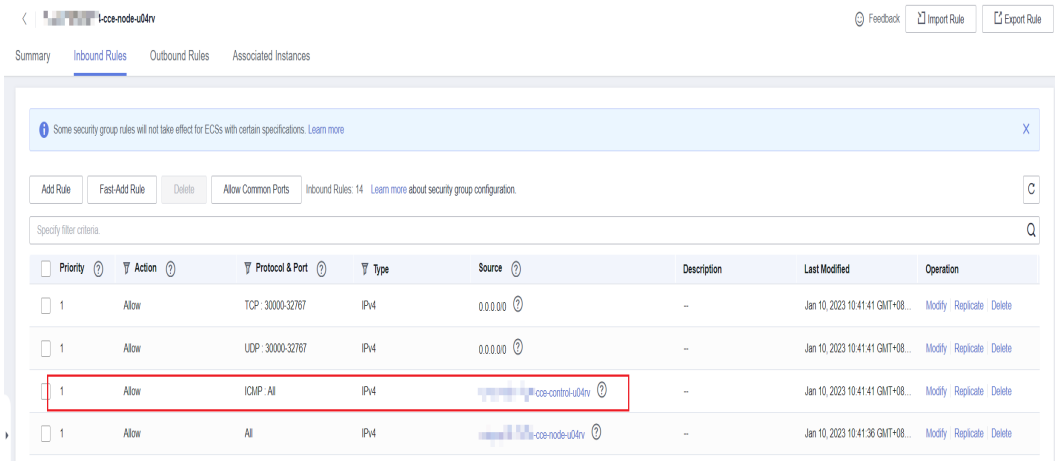
- The security group name is **cluster name-node-*xxx***. This security group is associated with the worker nodes.
- The security group name is **cluster name-control-*xxx***. This security group is associated with the master nodes.

Figure 3-17 Cluster security groups



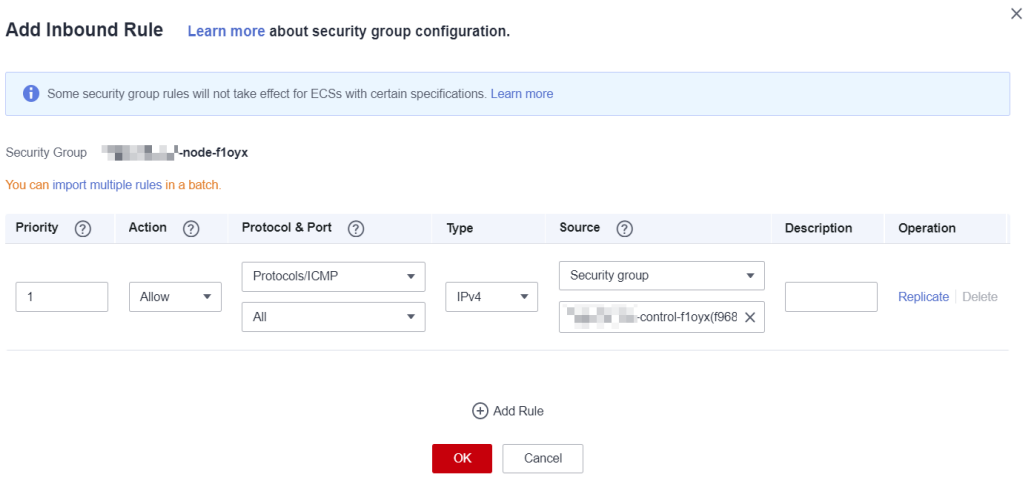
Click the node security group and ensure that the following rules are configured to allow the master node to access the node using **ICMP**.

Figure 3-18 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 3-19 Allowing ICMP for the master security group



3.2.4.5.9 To-Be-Migrated Nodes

Check Items

Check whether the node needs to be migrated.

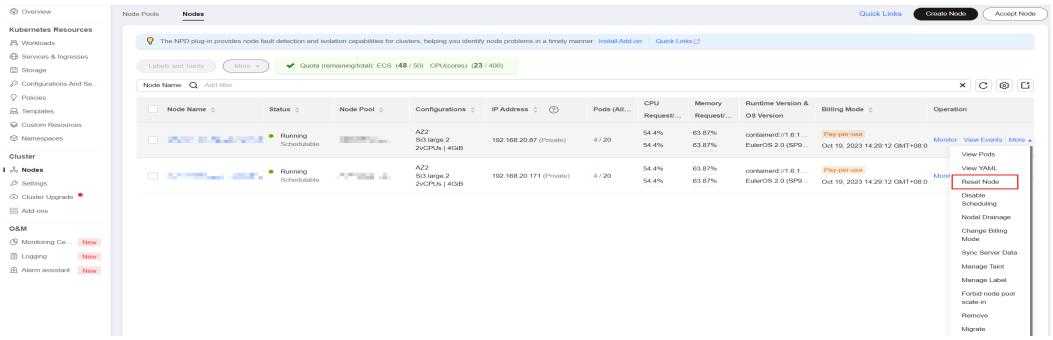
Solution

For the 1.15 cluster that is upgraded from 1.13 in rolling mode, migrate (reset or create and replace) all nodes before performing the upgrade again.

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 3-20 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.

3.2.4.5.10 Discarded Kubernetes Resources

Check Items

Check whether there are discarded resources in the clusters.

Solution

Scenario: 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 of **tolerate-unready-endpoints**. If yes, replace the annotation with the following fields:

```
publishNotReadyAddresses: true
```

3.2.4.5.11 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 **3.3.2 Container Engine**.	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**.
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 1.19 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 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. 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 1.15 and 1.19 clusters, the Docker storage driver file system is switched from XFS to Ext4. As a result, the import package sequence in the pods of the upgraded Java application may be abnormal, causing 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 1.19 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 1.15. kube-apiserver of CCE 1.19 is compiled using Go 1.15. 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 1.15 to 1.19 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 1.16 and later versions, **QoS classes** are different from those in earlier versions. In kubelet 1.15 and earlier versions, only containers in **spec.containers** are counted. In kubelet 1.16 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 3-22**.
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 is allocated 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 3-30 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

3.2.4.5.12 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.**

  cce-agent does not need to be updated but is not restarted. In this case, 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.**

  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.

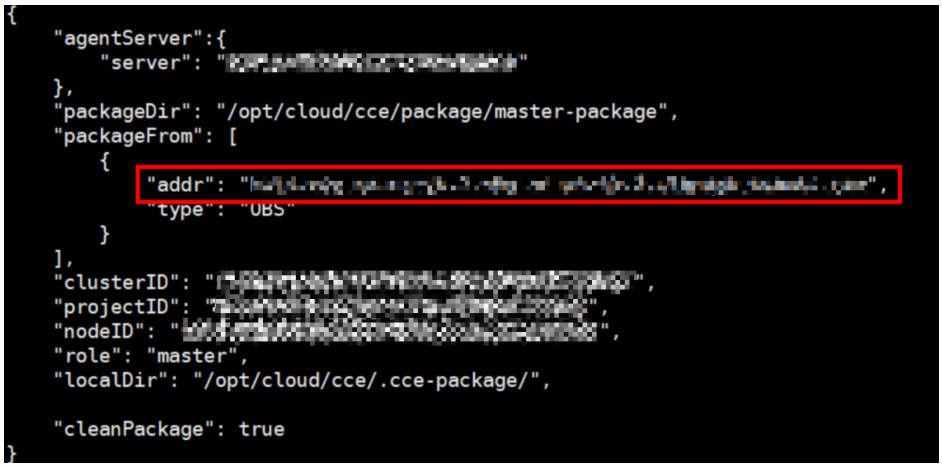
  Solution

  a. Log in to a node where the check succeeded, obtain the path of the cce-agent configuration file, and obtain the OBS address.


```
cat `ps aux | grep cce-agent | grep -v grep | awk -F '-f ' '{print $2}'`
```

The OBS configuration address field in the configuration file is **packageFrom.addr**.

Figure 3-21 OBS address



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.

3.2.4.5.13 Node CPU Usage

Check Items

Check whether the CPU usage of the node exceeds 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.

3.2.4.5.14 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.

3.2.4.5.15 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
```

3.2.4.5.16 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.

3.2.4.5.17 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.

3.2.4.5.18 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 is 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.

3.2.4.5.19 Node Memory

Check Items

Check whether the memory usage of the node exceeds 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.

3.2.4.5.20 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 3-22 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 3-23 Running status of chronyd

  ```
  root@[redacted]# systemctl status chronyd
  ● chrony.service - chrony, an NTP client/server
     Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-08-24 16:33:28 CST; 3 months 16 days ago
       Docs: man:chronyd(8)
             man:chronyc(1)
             man:chrony.conf(5)
    Process: 6492 ExecStartPost=/usr/lib/chrony/chrony-helper update-daemon (code=exited, status=0/SUCCESS)
    Process: 6461 ExecStart=/usr/lib/systemd/scripts/chronyd-starter.sh $DAEMON_OPTS (code=exited, status=0/SUCCESS)
  Main PID: 6488 (chronyd)
      Tasks: 1 (limit: 4915)
     CGroup: /system.slice/chrony.service
             └─6488 /usr/sbin/chronyd
  ```

  If the problem persists after chronyd is restarted, contact technical support.

3.2.4.5.21 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.

3.2.4.5.22 Node CPUs

Check Items

Check whether the number of CPUs on the master node is greater than 2.

Solution

If the number of CPUs on the master node is 2, contact technical support to expand the number to 4 or more.

3.2.4.5.23 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

Install Python before the upgrade.

3.2.4.5.24 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 3-31 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
1.13	v1.21 or v1.23
1.15	v1.21, v1.23, v1.27, or v1.25
1.18	v1.25, v1.27, or v1.28

- 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.

3.2.4.5.25 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.

3.2.4.5.26 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 3-24 Running status of journald

```
[root@[redacted] paas]# systemctl is-active systemd-journald
active
```

If the problem persists after journald is restarted, contact technical support.

3.2.4.5.27 containerd.sock

Check Items

Check whether the containerd.sock file exists 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 **ls /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

3.2.4.5.28 Internal Errors

Check Items

Before the upgrade, check whether an internal error occurs.

Solution

If this check fails, contact technical support.

3.2.4.5.29 Node Mount Points

Check Items

Check whether inaccessible mount points exist 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 Run the following commands on the node in sequence:

```
- df -h
- for dir in `df -h | grep -v "Mounted on" | awk "{print \\$NF}"`;do cd $dir; done && echo "ok"
```

Step 3 If **ok** is returned, no problem occurs.

Otherwise, start another terminal and run the following command to check whether the previous command is in the D state:

```
- ps aux | grep "D "
```

Step 4 If a process is in the D state, the problem occurs. You can restart the node to solve the problem. Restart the node and upgrade the cluster again.

NOTE

Workloads running on the node will be rescheduled after a node is restarted. Check whether services will be affected before restarting the node.

----End

3.2.4.5.30 Kubernetes Node Taints

Check Items

Check whether the taint needed for cluster upgrade exists on the node.

Table 3-32 Taint checklist

Taint Name	Impact
node.kubernetes.io/upgrade	NoSchedule

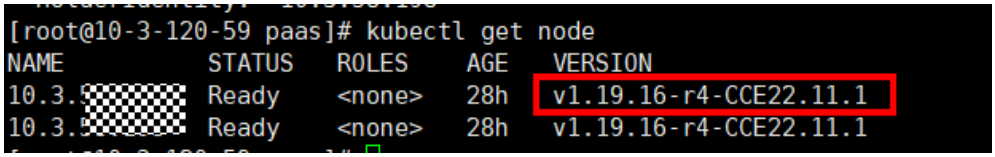
Solution

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 3-25 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

3.2.4.5.31 Everest Restrictions

Check Items

Check whether there are any compatibility restrictions on the current Everest add-on.

Table 3-33 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.

3.2.4.5.32 cce-hpa-controller Restrictions

Check Items

Check whether the current cce-controller-hpa add-on has compatibility restrictions.

Solution

The current cce-controller-hpa add-on has compatibility restrictions. An add-on that can provide metric APIs, for example, metric-server, must be installed in the cluster.

3.2.4.5.33 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 3-34 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

3.2.4.5.34 Health of Worker Node Components

Check Items

Check whether the container runtime and network components on the worker nodes are healthy.

Solution

If a worker node component malfunctions, log in to the node to check the status of the component and rectify the fault.

3.2.4.5.35 Health of Master Node Components

Check Items

Check whether the Kubernetes, container runtime, and network components of the master nodes are healthy.

Solution

If a master node component malfunctions, contact technical support.

3.2.4.5.36 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**.

3.2.4.5.37 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**.

3.2.4.5.38 Node 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 problem persists after NetworkManager is restarted, contact technical support.

3.2.4.5.39 Node ID File

Check Items

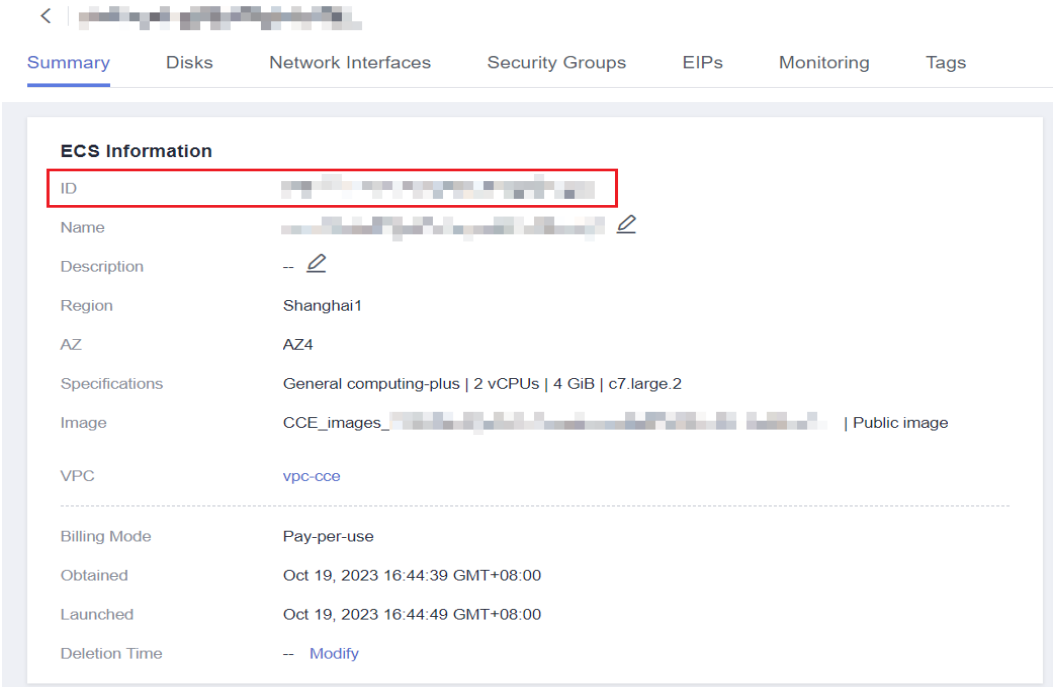
Check the ID file format.

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 3-26 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

3.2.4.5.40 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 3-35 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
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.

3.2.4.5.41 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

Contact technical support to restore the configuration file and then perform the upgrade.

3.2.4.5.42 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
```

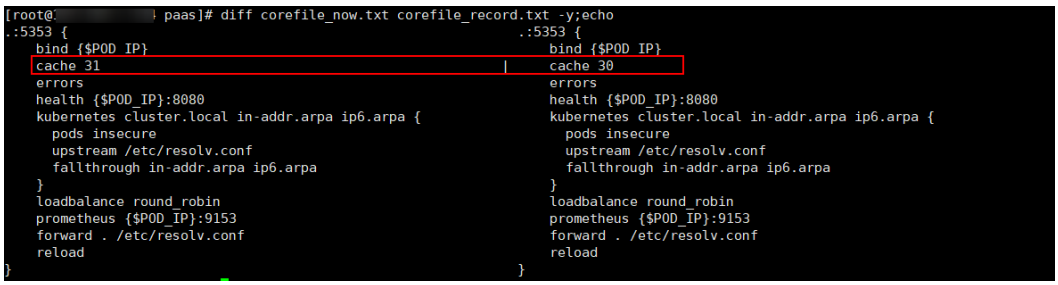
Step 3 Obtain the Corefile in the Helm Release records.

```
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 3-27 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

3.2.4.5.43 sudo Commands of a Node

Check Items

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
  ```

- 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
  ```

3.2.4.5.44 Key Commands of Nodes

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:
    ```
    rpm -qa
    ```
  - dpkg:
    ```
    dpkg -l
    ```

- 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
  ```


- Scenario 3: Executing the Python command failed.

  Check whether the command can be executed on the node.

  ```
  /usr/bin/python --version
  ```

3.2.4.5.45 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 the Dolphin add-on of versions earlier than 1.2.2, upgrade the add-on to 1.2.2 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.

3.2.4.5.46 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 3-28 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 **secertName** parameter in **spec.tls** of the ingress YAML files.

Run the following kubectl command to obtain the ingresses using a certificate (replace *<secret_id>* 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>
```

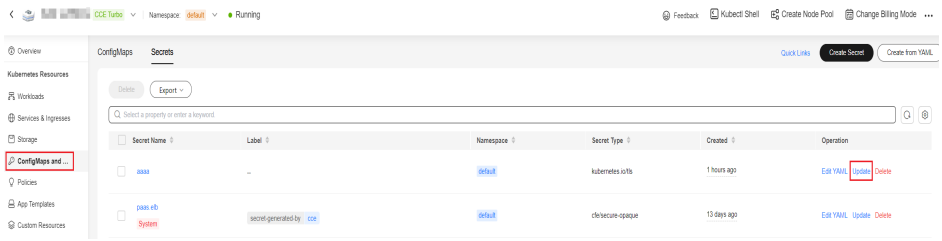
b. When configuring a listener, select **ELB server certificate** for **Certificate Source** and click **OK**. In this way, the certificate can be created or edited on the ELB console.

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 3-29 Modifying a Secret

----End

3.2.4.5.47 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.
  - The soft link of **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet** is created for CCE by default. Check whether it has been manually modified.
- Shared disk
  - By default, **/mnt/paas/** is mounted to CCE nodes. Check whether **/mnt** has been manually mounted.
  - The soft link of **/var/lib/kubelet** to **/mnt/paas/kubernetes/kubelet**, or **/var/lib/docker** or **containerd** to **/mnt/paas/runtime** is created for

CCE by default. Check whether the soft 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 3-30 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 link.

3.2.4.5.48 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
```

3.2.4.5.49 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

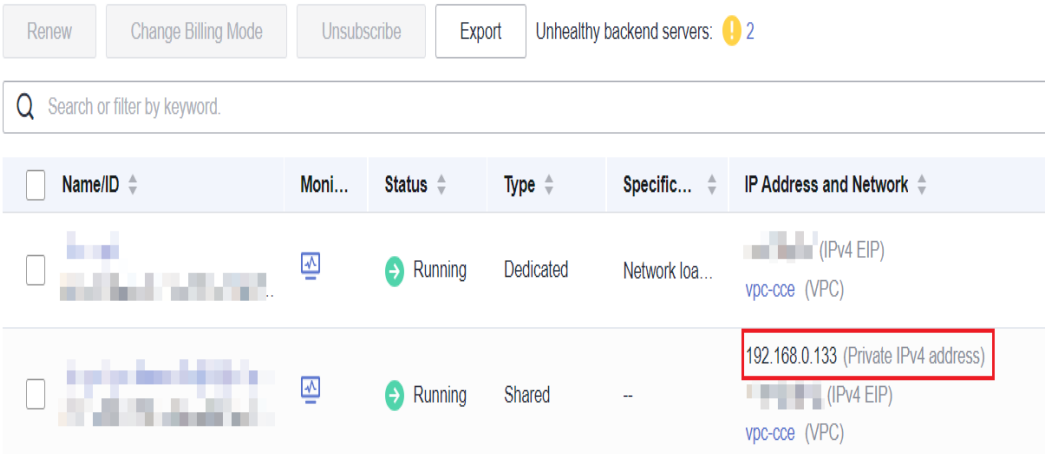
Solution 1: Delete the Service that is associated with a load balancer without a private IPv4 address.

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.

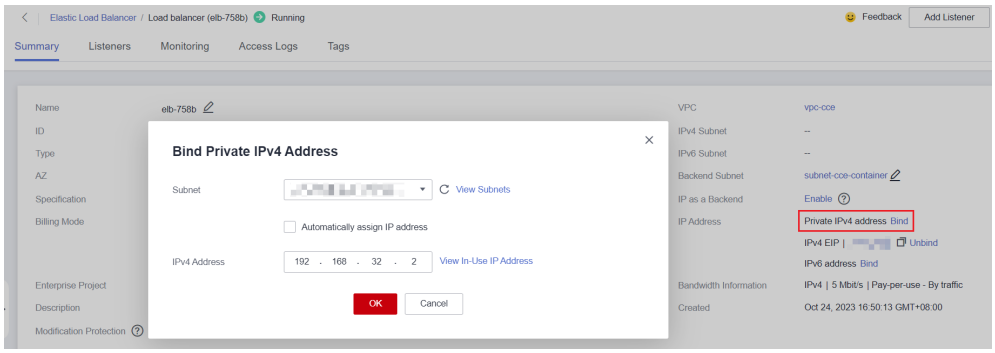
- Method 1: Obtain the load balancer ID based on the pre-upgrade check log. Go to the ELB console and filter load balancers by load balancer ID.
  elbs (**ids: [*****]**) without ipv4 private ip, please bind private ip tothese elbs and try again
- Method 2: Log in to the CCE console and click the cluster 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

3.2.4.5.50 Historical Upgrade Records

Check Items

Check whether the source version of the cluster is earlier than v1.11 and the target version is later than v1.23.

Solution

If the source version of the cluster is earlier than v1.11, it is risky to upgrade the cluster to a version later than v1.23. In this case, contact technical support.

3.2.4.5.51 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

If the CIDR block of the cluster management plane is different from that configured on the backbone network, contact technical support.

3.2.4.5.52 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.

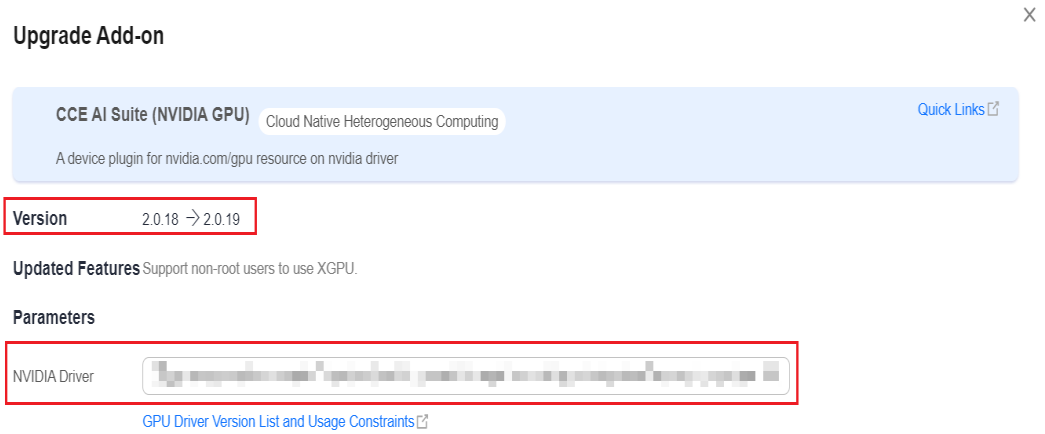
Perform the following operations to check the upgrade of the GPU driver to the target version and current driver configuration of GPU add-on:

Step 1 Log in to the CCE console and click **Add-ons** to view the GPU add-on.

NOTE

gpu-beta is the same as **gpu-device-plugin**. **gpu-beta** is renamed **gpu-device-plugin** in versions later than 2.0.0.

Step 2 Click **Upgrade** of the add-on to view the target version and driver configuration 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

3.2.4.5.53 Nodes' System Parameter Settings

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.

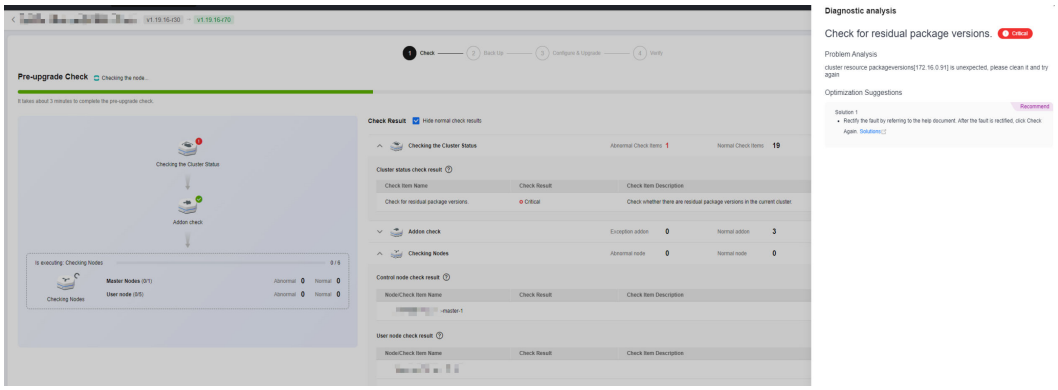
3.2.4.5.54 Residual Package Versions

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

3.2.4.5.55 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.

3.2.4.5.56 Node Swap

Check Items

Check whether swap has been enabled on cluster 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. Run the **swapoff -a** command to disable swap.

3.2.4.5.57 nginx-ingress Upgrade

Check Items

- 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.

Fault Locating

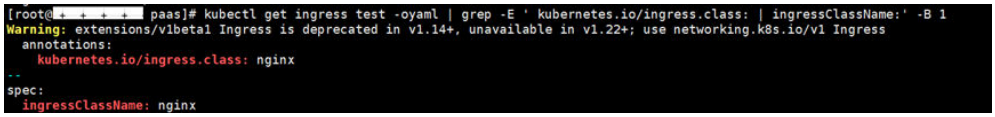
For an Nginx ingress, check the YAML. If the ingress type is not specified in the YAML file and the ingress is managed by the Nginx Ingress Controller, the ingress is at risk. For details, see **Possible Causes**.

Step 1 Check the Ingress type.

Run the following command:

```
kubectl get ingress <ingress-name> -oyaml | grep -E ' kubernetes.io/ingress.class: | ingressClassName:'
```

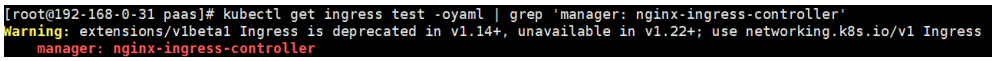
- Fault scenario: If the command output is empty, the Ingress type is not specified.
- Normal scenario: The command output is not empty, indicating that the Ingress type has been specified by **annotations** or **ingressClassName**.



Step 2 Ensure that the Ingress is managed by the Nginx Ingress Controller. The LoadBalancer Ingresses are not affected by this issue.

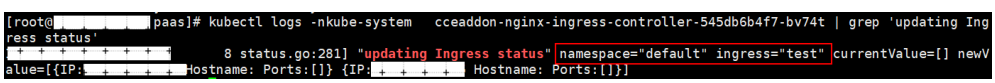
- For clusters of v1.19, confirm this issue using **managedFields**.

```
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 personnel.

----End

Solution

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.

3.2.4.5.58 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.

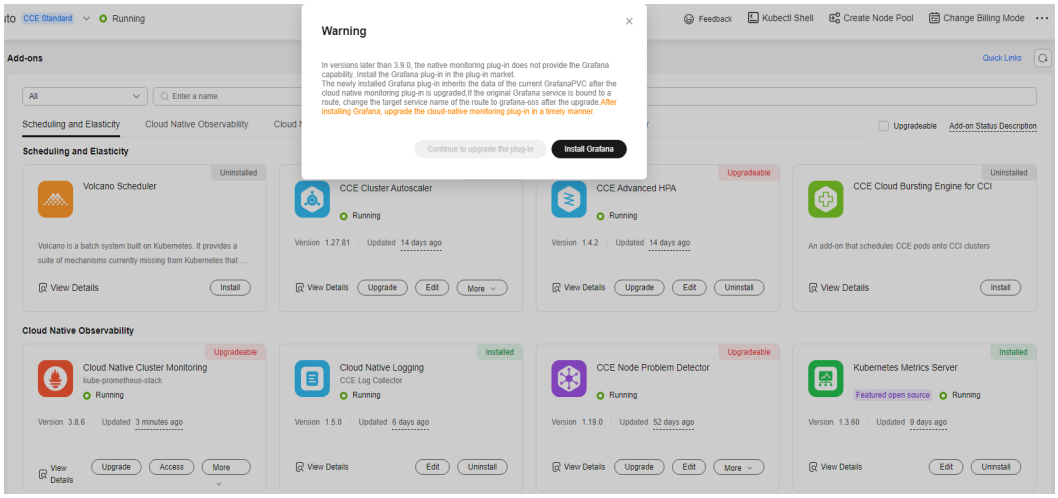
Solution

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.

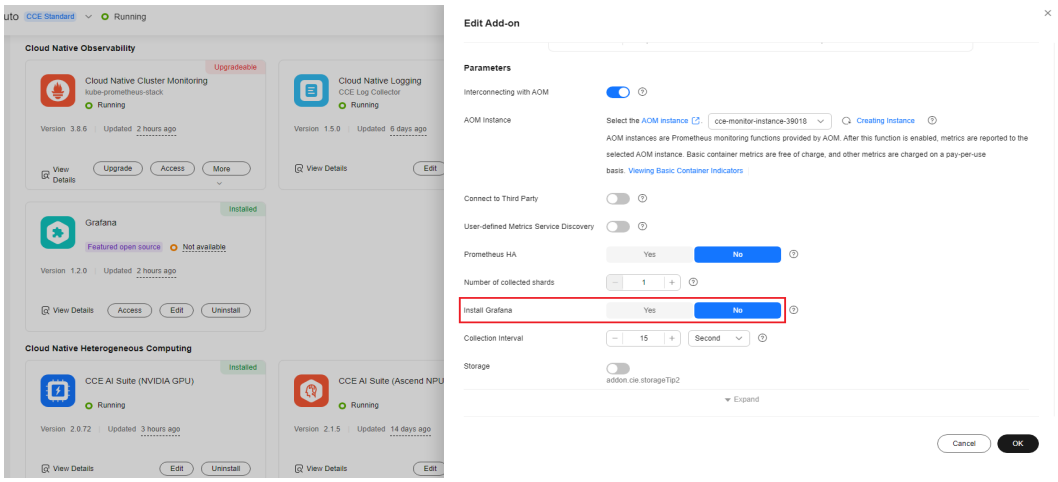
Solution 1: If Cloud Native Cluster Monitoring can be upgraded to 3.9.0 or later, upgrade it on the **Add-ons** page. In the displayed dialog box, click **Install Grafana**. After the request is submitted, continue to upgrade the add-on to the latest version.



Solution 2: If Cloud Native Cluster Monitoring cannot be upgraded to 3.9.0 or later, perform the following operations to manually migrate Grafana:

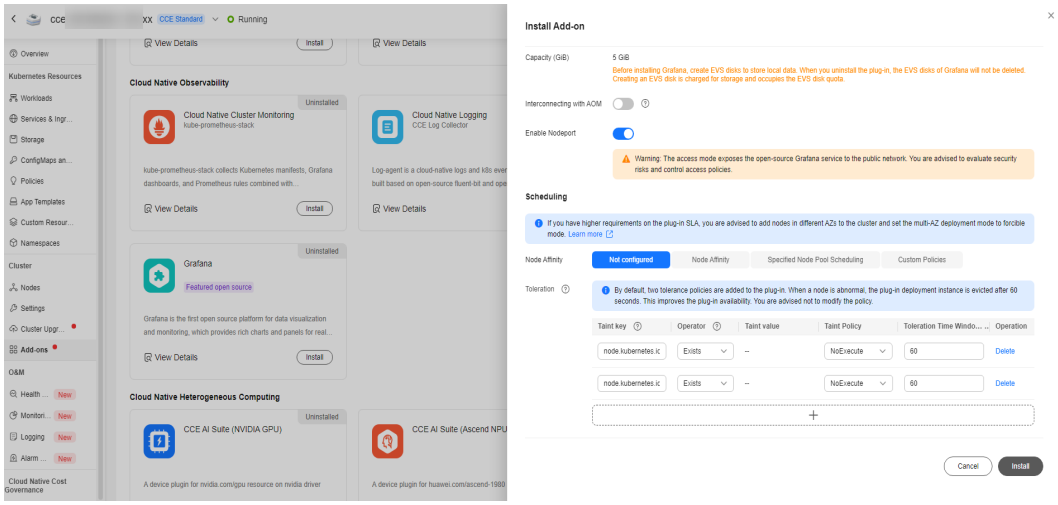
Step 1 Disable Grafana on Cloud Native Cluster Monitoring.

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

3.2.4.5.59 containerd Pod Restart Risks

Check Items

Check whether the service pods running on a containerd node are restarted when containerd is upgraded.

Solution

Upgrade the cluster when the impact on services is controllable (for example, during off-peak hours) to minimize the impact. If you need help, contact O&M personnel.

3.2.4.5.60 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 add-on IP address on the add-on details page. If no, change the value on the web page.

----End

3.2.4.5.61 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, contact O&M personnel.

3.2.4.5.62 ELB Listener Access Control

Check Items

Check whether the access control of the ELB listener has been configured for the Service in the current cluster using annotations and whether the configurations are correct.

Solution

In case of an incorrect configuration, correct it by following the instructions provided in **3.7.3.4.10 Configuring a Blocklist/Trustlist Access Policy for a Service**.

3.2.4.5.63 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

Flavor inconsistency is typically due to a modification made on the master nodes. After the cluster is upgraded, the modification of the master nodes may be restored. If the impact of the restoration cannot be evaluated, contact O&M personnel.

3.2.4.5.64 Subnet Quota of Master Nodes

Check Items

Check whether the number of available IP addresses in the cluster subnet supports rolling upgrade.

Solution

If the number of IP addresses in the selected cluster subnet is insufficient, rolling upgrade is not supported. Contact O&M personnel for support.

3.2.4.5.65 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 still want to create and use Docker nodes in a cluster later than v1.27, skip this alarm. However, you are advised to switch to containerd as soon as possible for better user experience and more powerful functions.

3.2.4.5.66 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 still want to create and use Docker node pools in a cluster later than v1.27, skip this alarm. However, you are advised to switch to containerd as soon as possible for better user experience and more powerful functions.

3.2.4.5.67 Number of Node Images

Check Items

Check the number of images on your node. If the number is greater than 1000, Docker startup may be slow.

Solution

Contact O&M personnel to check whether this issue affects the upgrade.

3.2.5 Managing a Cluster

3.2.5.1 Cluster Configuration Management

Scenario

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

Constraints

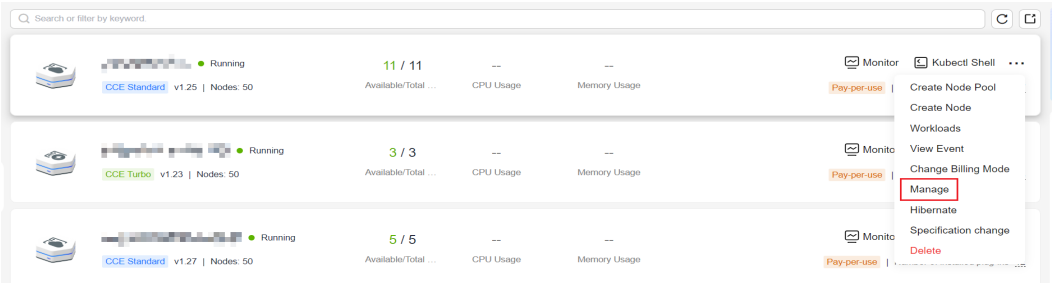
This function is supported only in clusters of **v1.15 and later**. It is not displayed for versions earlier than v1.15.

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 3-31 Configuration management



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

Table 3-36 kube-apiserver configuration

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 **3.5.3.12 Taints and Tolerations**. 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 **3.5.3.12 Taints and Tolerations**. 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. If enabled, concurrent requests are dynamically controlled based on the resource pressure of master nodes to keep them and the cluster available. 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.	Default: true
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.	Default: true

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.	Default: false

Table 3-37 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 **3.6.5 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 a cluster contains 1000 or more nodes, the default value is **200**.

Item	Parameter	Description	Value
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 a cluster contains 1000 or more nodes, the default value is **200**.
Whether to enable GPU sharing	enable-gpu-share	Whether to enable GPU sharing. This parameter is supported only by clusters of v1.23.7-r10, v1.25.3-r0, and later. • When disabled, ensure that pods in the cluster do not use the shared GPU (that is, the annotation of **cce.io/gpu-decision** does not exist in pods). • When enabled, ensure that the annotation of **cce.io/gpu-decision** exists in pods that use GPU resources in the cluster.	Default: true

Table 3-38 kube-controller-manager configurations

Item	Parameter	Description	Value
Number of concurrent processing of deployment	concurrent-deployment-syncs	Number of deployment objects that are allowed to sync concurrently	Default: 5
Concurrent processing number of endpoint	concurrent-endpoint-syncs	Number of endpoint syncing operations that will be done concurrently	Default: 5

Item	Parameter	Description	Value
Concurrent number of garbage collector	concurrent-gc-syncs	Number of garbage collector workers that are allowed to sync concurrently	Default: 20
Number of job objects allowed to sync simultaneously	concurrent-job-syncs	Number of job objects that are allowed to sync concurrently	Default: 5
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 are allowed to sync concurrently	Default: 10
Concurrent processing number of replicaset	concurrent-replicaset-syncs	Number of replica sets that are allowed to sync concurrently	Default: 5
ResourceQuota	concurrent-resource-quota-syncs	Number of resource quotas that are allowed to sync concurrently	Default: 5
Concurrent processing number of service	concurrent-service-syncs	Number of services that are allowed to sync concurrently	Default: 10
Concurrent processing number of serviceaccount-token	concurrent-serviceaccount-token-syncs	Number of service account token objects that are allowed to sync concurrently	Default: 5
Concurrent processing of ttl-after-finished	concurrent-ttl-after-finished-syncs	Number of **ttl-after-finished-controller** workers that are allowed to sync concurrently	Default: 5

Item	Parameter	Description	Value
RC	concurrent_rc_syncs	Number of replication controllers that are allowed to sync concurrently **NOTE** This parameter is used only in clusters of v1.19 or earlier.	Default: 5
RC	concurrent-rc-syncs	Number of replication controllers that are allowed to sync concurrently **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
Cluster elastic computing period	horizontal-pod-autoscaler-sync-period	How often HPA audits metrics in a cluster	Default: 15 seconds
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 a cluster contains 1000 or more nodes, the default value is **200**.

Item	Parameter	Description	Value
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 a cluster contains 1000 or more nodes, the default value is **200**.
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, v1.27.3-r0, v1.25.6-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. **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. **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. **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. **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 3-39 Networking component configurations (supported only by 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 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.	Default: 2
Cluster-level node warm-up container NIC recycling threshold	nic-max-above-warm-target	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. Only numbers are allowed. • A large value will accelerate pod startup but slow down the unbinding of idle container ENIs and decrease the IP address usage. • 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 3-40 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 **3.10.3 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**

3.2.5.2 Cluster Overload Control

Scenario

If enabled, concurrent requests are dynamically controlled based on the resource pressure of master nodes to keep them and the cluster available.

Constraints

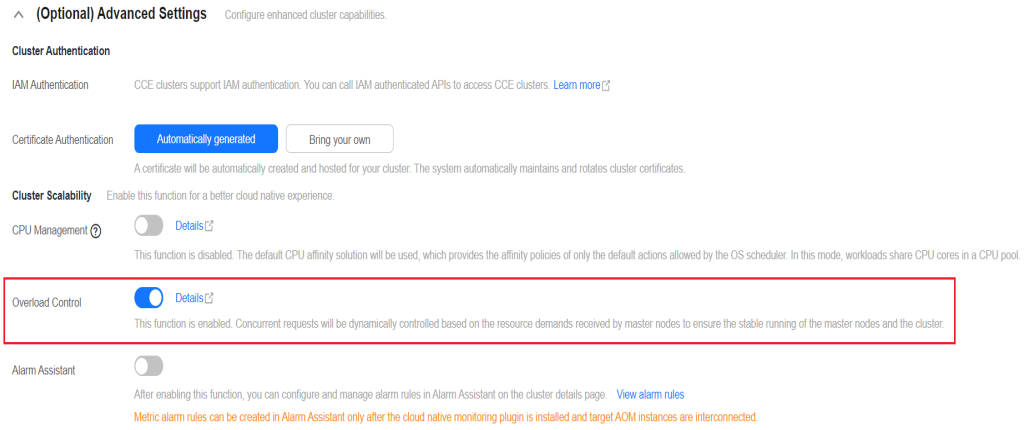
The cluster version must be 1.23 or later.

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 3-32 Enabling overload control during cluster creation

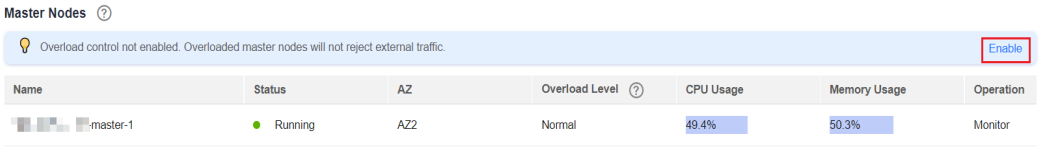


Method 2: Enabling it in an existing cluster

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 3-33 Enabling overload control for an existing cluster



----End

Disabling Cluster Overload Control

Step 1 Log in to the CCE console and go to 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

3.2.5.3 Changing Cluster Scale

Scenario

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

Constraints

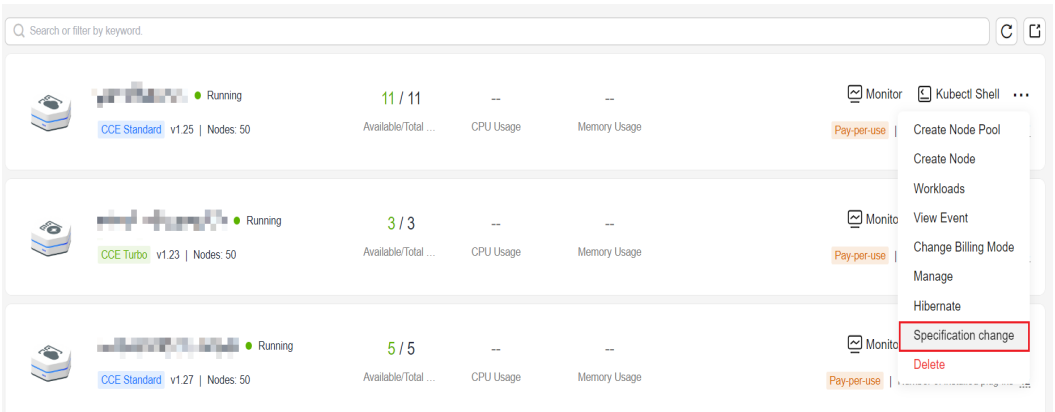
- This function is supported for clusters of v1.15 and later versions.
- Starting from v1.15.11, the number of nodes in a cluster can be changed to 2000. The number of nodes in a single master node cannot be changed to 1000 or more.
- The number of master nodes cannot be changed when you modify cluster specifications.
- Currently, a cluster can only be scaled out to a larger specification, but cannot be scaled in.
- During the specifications change, master nodes will be powered off and on, and the cluster cannot run properly. Perform the change during off-peak hours.
- Changing the cluster scale does not affect the services running in the cluster. However, the control plane (master nodes) will be interrupted for a short period of time. You are advised not to perform any other operations (such as creating workloads) during the change.
- Change failures will trigger a cluster rollback to the normal state. If the rollback fails, submit a service ticket.

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 **Specification change**.

Figure 3-34 Modifying specifications



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

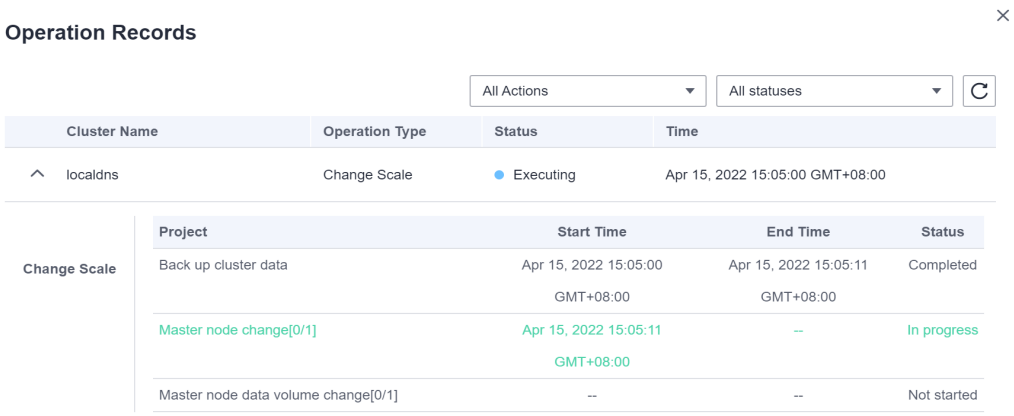
Step 4 Click **Next** to confirm the specifications and click **OK**.

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 **3.2.5.1 Cluster Configuration Management**.

Figure 3-35 Operation records



----End

3.2.5.4 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.

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**.
- The security group of the master node cannot be specified. Exercise caution when modifying the security group rules of the master node. For details, see **Configuring Cluster Security Group Rules**.

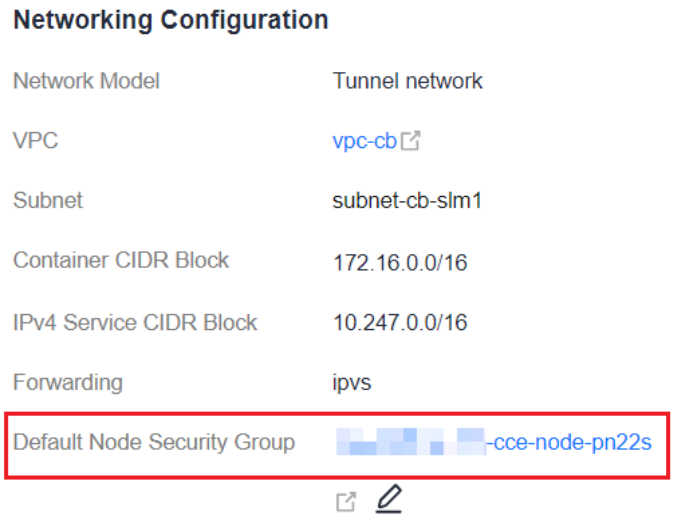
Procedure

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 **Network Configuration** area, click next to the **Default Node Security Group**.

Figure 3-36 Default node security group

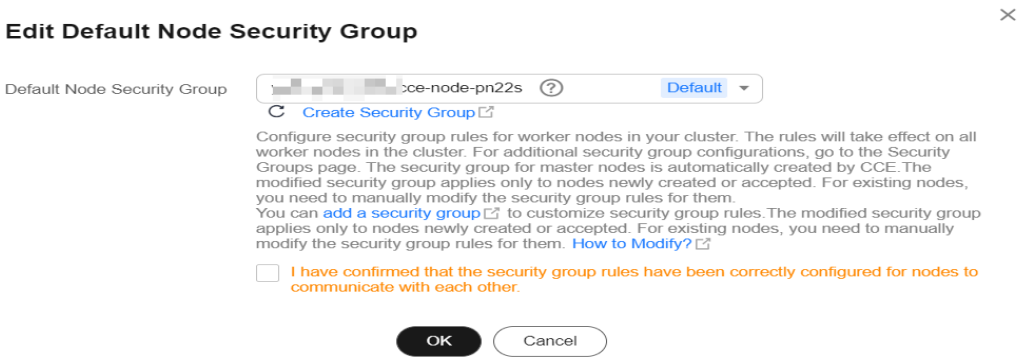


Step 4 Select an existing security group, confirm that the security group rules meet the cluster requirements, and click **OK**.

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 3-37 Editing default node security group



----End

3.2.5.5 Deleting a Cluster

Scenario

- You can directly delete pay-per-use clusters. For details, see **Deleting a 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 Cluster**.

Precautions

- When a cluster is deleted, the managed and yearly/monthly nodes will be removed from the cluster and the system will be reinstalled. The original login passwords of the nodes will become invalid. **Reset the login passwords of the nodes.**
- Deleting a cluster will not delete the yearly/monthly-billed resources in the cluster, and their billing continues.
- Deleting a cluster will delete the nodes in the cluster (excluding accepted nodes), data disks attached to the nodes, workloads, and Services. Related services cannot be restored. Before performing this operation, ensure that data has been backed up or migrated. Deleted data cannot be restored.

  Resources that are not created in CCE will not be deleted:

  - Accepted nodes (only the nodes created in CCE are deleted)
  - ELB load balancers associated with Services and ingresses (only the automatically created load balancers are deleted)
  - Manually created cloud storage resources associated with PVs or imported cloud storage resources (only the cloud storage resources automatically created by PVCs are deleted)
- If you delete a cluster that is not running (for example, frozen or unavailable), associated resources, such as storage and networking resources, will remain.
- If the cluster version is v1.13.10 or earlier, do not manually change the listener name and backend server name on the ELB console. Otherwise, residual resources will exist when you delete the cluster.

Deleting a Cluster

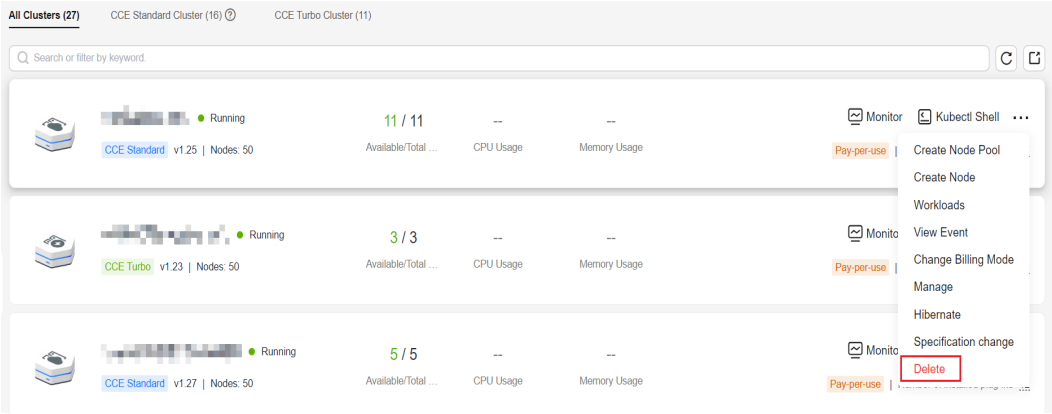
NOTICE

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 3-38 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.
- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).
- Delete the master log stream of LTS (Only automatically created log streams can be deleted).

> **NOTE**
>
> If you do not delete the log stream, existing logs will not be deleted and you may be charged for LTS logs. 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 Cluster

NOTICE

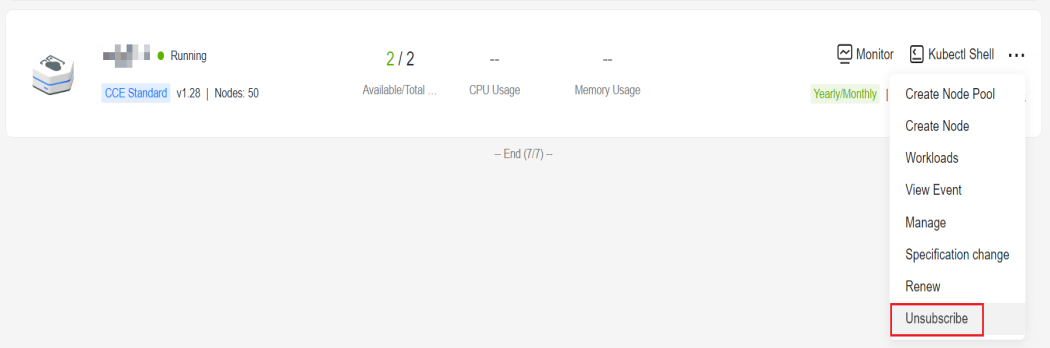
- When you unsubscribe from or release a cluster, only the resources associated with the order are unsubscribed from. Non-associated resources are retained and their billing keeps going.
- For a yearly/monthly cluster, if the retention expires, the cluster will be automatically released. For the nodes 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. If you want to reserve your resources and unsubscribe from only the unused renewal periods, log in to the Enterprise Projects console and **unsubscribe from renewal period as prompted**.
  - 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 3-39 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.
- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).
- Delete the master log stream of LTS (Only automatically created log streams can be deleted).

  NOTE

  If you do not delete the log stream, existing logs will not be deleted and you may be charged for LTS logs. For details, see **Price Calculator**.

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

----End

3.2.5.6 Hibernating and Waking Up a (Pay-per-Use) Cluster

Scenario

If you do not need to use a cluster temporarily, hibernate the cluster.

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

A hibernated cluster can be quickly woken up and used properly.

Constraints

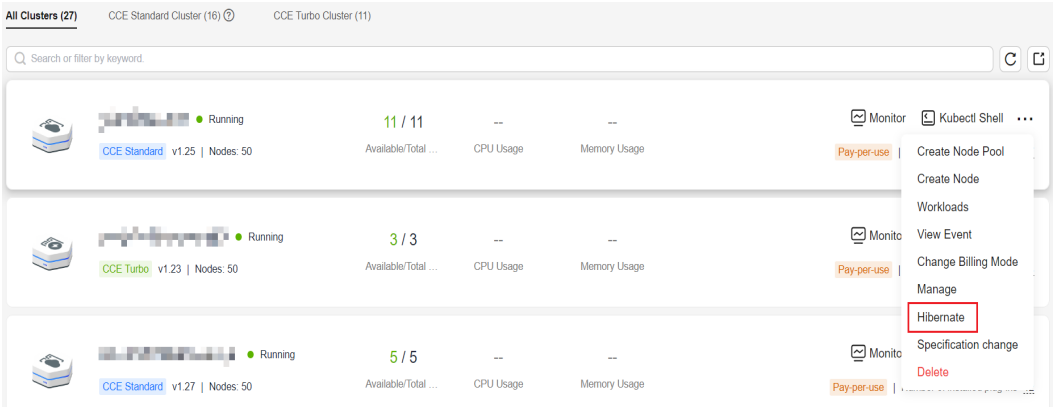
- Clusters billed on a yearly/monthly basis cannot be hibernated.
- During cluster wakeup, the master node may fail to start due to insufficient resources, which leads to a cluster wakeup failure. In this case, wait for a while and try again.
- After a cluster is woken up, it takes 3 to 5 minutes to 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 3-40 Hibernating a cluster

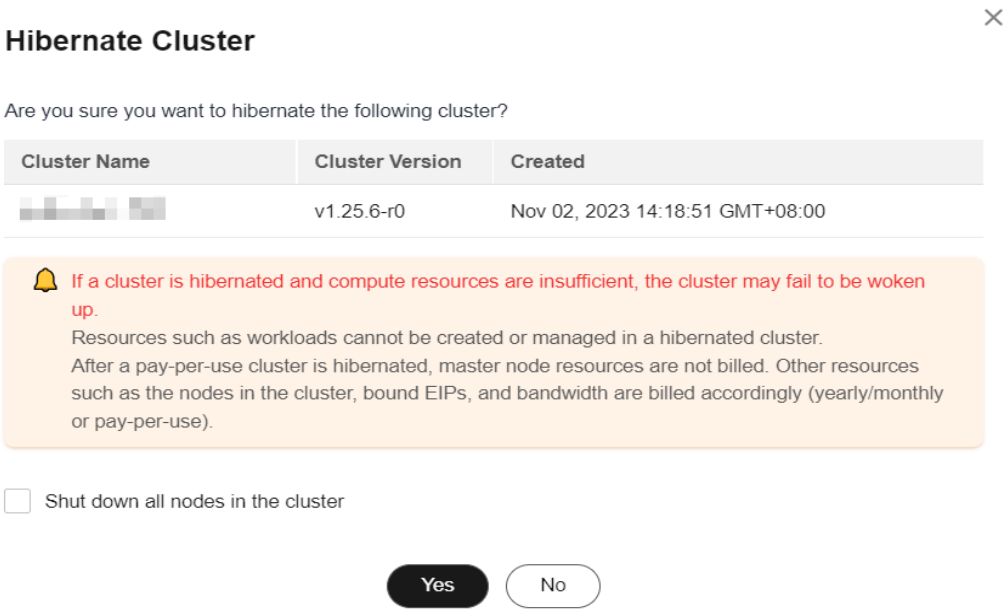


Step 3 In the dialog box displayed, check the precautions and click **Yes**. Wait until the cluster is hibernated.

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 **3.3.7.10 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 3-41 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

3.2.5.7 Renewing a Yearly/Monthly-Billed Cluster

You can renew a yearly/monthly-billed cluster.

Procedure

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

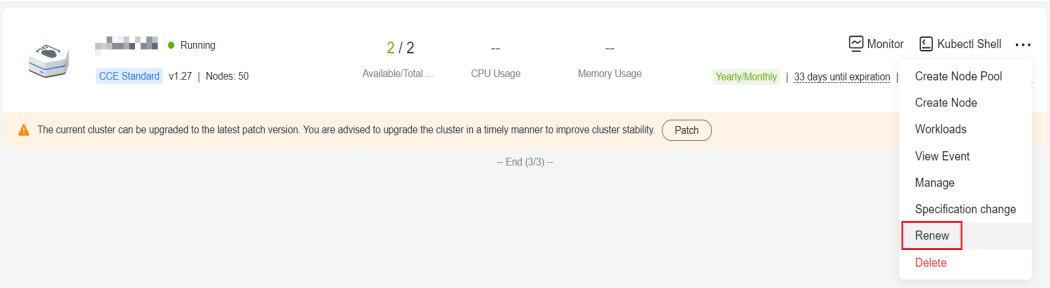
NOTICE

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 3-42 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

3.2.5.8 Changing the Billing Mode from Pay-per-Use to Yearly/Monthly

Currently, clusters support **pay-per-use** and **yearly/monthly** billing modes. A pay-per-use cluster can be converted to a yearly/monthly-billed cluster.

Constraints

- A pay-per-use node cannot be changed to a yearly/monthly one on the ECS console.
- Only nodes in the default node pool can be changed to the yearly/monthly billing mode.
- Nodes whose billing mode is changed to yearly/monthly do not support auto scaling.

Changing the Billing Mode of a Cluster

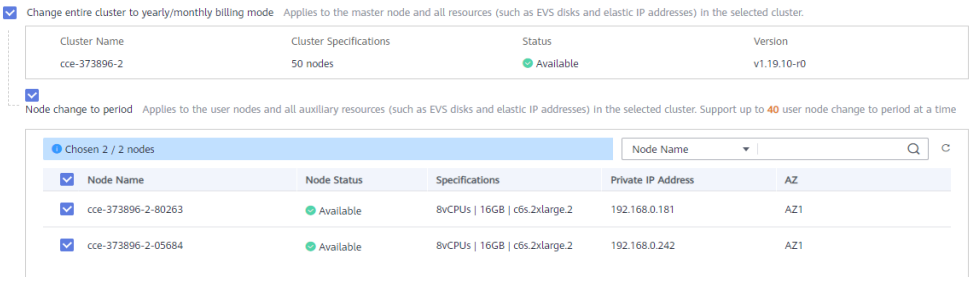
To change the billing mode of a cluster from pay-per-use to yearly/monthly, perform the following steps:

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

Step 2 Locate the target cluster, and click **Change Billing Mode** in the upper right corner.

Step 3 On the **Change Billing Mode** page, select the target cluster. You can also select the nodes whose billing mode you want to change.

Figure 3-43 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

3.3 Nodes

3.3.1 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 ext4 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 3-41 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.

Status	Status Attribute	Description
Unavailable	Stable state	The node is not running properly. Instances in this state no longer provide services. In this case, perform the operations in **3.3.7.3 Resetting a Node**.
Creating	Intermediate state	The node has been created but is not running.
Installing	Intermediate state	The Kubernetes software is being installed on the node.
Deleting	Intermediate state	The node is being deleted. If this state stays for a long time, an exception occurred.
Stopped	Stable state	The node is stopped properly. A node in this state cannot provide services. You can start the node on the ECS console.
Error	Stable state	The node is abnormal. Instances in this state no longer provide services. In this case, perform the operations in **3.3.7.3 Resetting a Node**.

3.3.2 Container Engine

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**. To ensure smooth service experience, migrate your Docker nodes to containerd nodes. For details, see **3.3.8.4 Migrating Nodes from Docker to containerd**.

Table 3-42 Comparison between container engines

Item	containerd	Docker
Tracing	kubelet --> CRI plugin (in the containerd process) --> containerd	• Docker (Kubernetes v1.23 and earlier): 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
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 3-43 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

OS	Kernel Version	Container Engine	Container Storage Rootfs	Container Runtime
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
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 3-44 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 3-45 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 3-46 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 3-47 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

Operation	Docker Command	containerd Command	
	docker	crictl	ctr
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
Check the resource usage of the container.	docker stats	crictl stats	None
Update container resource limits.	docker update	crictl update	None

Table 3-48 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 1.23 and earlier versions):

  kubelet --> docker shim (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

- Docker
  - EulerOS/CentOS: docker-engine 18.9.0, a Docker version customized for CCE. Security vulnerabilities will be fixed in a timely manner.
  - Ubuntu: docker-ce 18.9.9 (community version). You are advised to use the containerd engine for Ubuntu nodes.

    NOTE

    The open source docker-ce of Ubuntu may trigger bugs when concurrent exec operations are performed (for example, multiple exec probes are configured). You are advised to use HTTP/TCP probes.

- containerd: 1.6.14

3.3.3 Node OS

This section describes the mappings between released cluster versions and OS versions.

ECS (VM)

Table 3-49 ECS (VM) OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
Huawei Cloud EulerOS 2.0	v1.29	√	√	√	5.10.0-60.18.0.50.r1083_58.hce2.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.28	√	√	√	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.27	√	Supported in v1.27.3-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.25	√	Supported in v1.25.6-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
	v1.23	√	Supported in v1.23.11-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.x86_64
Huawei Cloud EulerOS 2.0 (Arm)	v1.29	√	√	√	5.10.0-60.18.0.50.r1083_58.hce2.aarch64
	v1.28	√	√	√	5.10.0-60.18.0.50.r1083_58.hce2.aarch64
	v1.27	√	Supported in v1.27.3-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.aarch64
	v1.25	√	Supported in v1.25.6-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.aarch64
	v1.23	√	Supported in v1.23.11-r0 or later.	√	5.10.0-60.18.0.50.r1083_58.hce2.aarch64
Ubuntu 22.04	v1.29	√	x	√	5.15.0-92-generic
	v1.28	√	x	√	5.15.0-92-generic

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.27	√	x	√	5.15.0-92-generic
	v1.25	√	x	√	5.15.0-92-generic
	v1.23	√	x	√	5.15.0-92-generic
Huawei Cloud EulerOS 1.1	v1.29	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.28	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.27	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.25	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.23	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
	v1.21	√	√	√	3.10.0-1160.76.2.hce1c.x86_64
CentOS Linux release 7.6	v1.29	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.28	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.27	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.25	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.23	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.21	√	√	√	3.10.0-1160.108.1.el7.x86_64
	v1.19 (End of maintenance)	√	√	√	3.10.0-1160.108.1.el7.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	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
	v1.13.7-r0 (end of maintenance)	√	√	√	3.10.0-957.21.3.el7.x86_64
EulerOS release 2.9	v1.29	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.28	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.27	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.25	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
	v1.23	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.21	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
	v1.19 (End of maintenance)	√	√	√	4.18.0-147.5.1.6.h1152.eulerosv2r9.x86_64
EulerOS release 2.9 (Arm)	v1.29	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.28	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.27	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.25	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.23	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
	v1.21	√	√	√	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 2.0 Network	
	v1.19 (End of maintenance)	√	√	√	4.19.90-vhulk2103.1.0.h1144.eulerosv2r9.aarch64
EulerOS release 2.8 (Arm, end of maintenance)	v1.27 or later	x	x	x	None
	v1.25	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.23	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.21	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.16	√	√	√	4.19.36-vhulk1907.1.0.h1350.eulerosv2r8.aarch64
	v1.19.10	√	√	√	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

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.5 (end of maintenance)	v1.27 or later	x	x	x	None
	v1.25	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.23	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.21	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.19.16	√	√	√	3.10.0-862.14.1.5.h687.eulerosv2r7.x86_64
	v1.19.10	√	√	√	3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64
	v1.19.8	√	√	√	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 2.0 Network	
	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	√	x	√	4.15.0-171-generic
	v1.23	√	x	√	4.15.0-171-generic
	v1.21	√	x	√	4.15.0-171-generic
	v1.19.16	√	x	√	4.15.0-171-generic
	v1.19.8	√	x	√	4.15.0-136-generic
	v1.17.17	√	x	√	4.15.0-136-generic

ECS (PM)

Table 3-50 ECS (PM) OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.10	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	√	√	√	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 3-51 BMS OS

OS	Cluster Version	CCE Standard Cluster		CCE Turbo Cluster	Latest Kernel
		VPC Network	Tunnel Network	Cloud Native 2.0 Network	
EulerOS release 2.9 (restricted use. Submit a service ticket to apply for it.)	v1.28	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	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	√	√	x	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64
	v1.19	√	√	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	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.23	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.21	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.19	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.17	√	√	x	3.10.0-514.41.4.28.h62.x86_64
	v1.15.11	√	√	x	3.10.0-514.41.4.28.h62.x86_64

3.3.4 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**.

Constraints

- The node has at least 2 vCPUs and 4 GiB of memory.
- 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. 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 **3.3.8.1 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 NIC named **gw_11cbf51a@eth0** on the node is the container network gateway and cannot be modified.
- If you want to modify the specifications of a purchased node, stop the node and perform the operations described in **General Operations for Modifying Specifications**. You can also purchase a new node and delete the old one.
- During the node creation, software packages are downloaded from OBS using the domain name. A private DNS server must be used to resolve the OBS domain name. Therefore, the DNS server address of the subnet where the node resides must be set to the **private DNS server address** so that the node can access the private DNS server. When you create a subnet, the private DNS server is used by default. If you change the subnet DNS, ensure that the DNS server in use can resolve the OBS domain name.
- Once a node is created, its AZ cannot be changed.
- Nodes purchased in the pay-per-use billing mode will be deleted after you delete them on the **Nodes** page on the CCE console. **Yearly/monthly-billed nodes** in a cluster cannot be deleted on the CCE console. You can choose **Billing Center** > **My Orders** in the upper right corner of the page to unsubscribe from the nodes.
- Services may be compromised by node process ID limits. Evaluate whether to adjust the maximum number of process IDs. For details, see **3.3.8.5.5 Changing Process ID Limits (kernel.pid_max)**.
- When IPv4/IPv6 dual stack is enabled, DHCP unlimited lease cannot be enabled for the selected node subnet.

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 3-52 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.
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 specifications that best fit your service needs. 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 (for example, **default**) is selected, node resources can be managed through the enterprise project, but enterprise project management permissions are not supported. The default value is the enterprise project selected for the cluster. Enterprise projects facilitate project-level management and grouping of cloud resources and users. 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**.

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 3-53 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. **Encryption**: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **This function is available only in certain regions.** • **Encryption** is not selected by default. • After setting **System Disk Encryption** to **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 **Encryption** text box.

Parameter	Description
Data Disk	**At least one data disk is required** for the container runtime and kubelet. **The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.** • First data disk: used for container runtime and kubelet components. The value ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. • Other 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. **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 **3.3.8.2 Data Disk Space Allocation**. • **Encryption**: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **This function is available only in certain regions.** – **Encryption** is not selected by default. – After selecting **Encryption**, you can select an existing key in the displayed dialog box. 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 **Encryption** text box. **Adding data disks** A maximum of four data disks can be added. By default, raw disks are 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**.

Parameter	Description
	• **Use as ephemeral volume**: applicable when there is a high performance requirement on EmptyDir. **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 PVs** and **Local 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 multiple volumes exist.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 3-54 Configuration parameters

Parameter	Description
VPC/Node Subnet	The node subnet selected during cluster creation is used by default. You can choose another subnet instead.
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 3-55 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined 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**.
Taint	This parameter is left blank by default. You can add taints to configure node 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 **3.3.7.2 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 **3.3.8.3 Maximum Number of Pods That Can Be Created on a Node**.

Parameter	Description
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	Pre-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	Pre-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 account administrator on the IAM console. By creating 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.

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. If you want to change the Kubernetes node name to cloud server name, you can remove this node from the cluster and accept it again. Before doing so, learn about the possible impacts on services when **removing a node** and **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

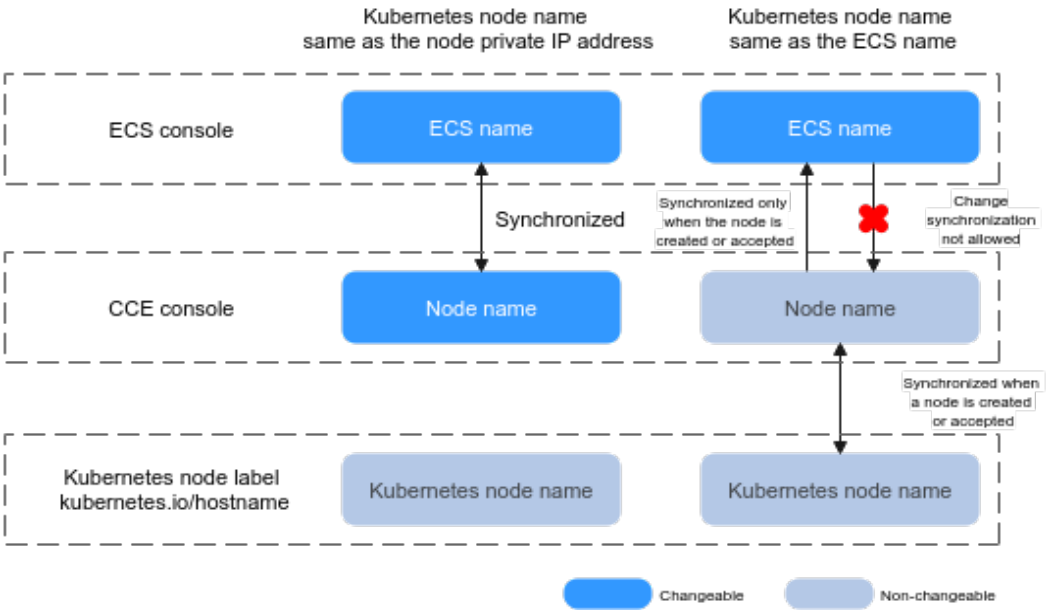
ECS Names, Node Names, and Kubernetes Node Names

- 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 3-44 shows the synchronization relationship between ECS names, node names, and Kubernetes node names.

Figure 3-44 Relationship between node names



Related Operations

Create a node injection script.

3.3.5 Accepting Nodes for Management

Scenario

In CCE, you can create a node (**3.3.4 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

- While an ECS is being accepted into a cluster, the operating system of the ECS will be reset to the standard OS image provided by CCE to ensure node stability. The CCE console prompts you to select the operating system and the login mode during the reset.
- 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 management. Ensure that the information has been backed up.
- While an ECS is being accepted into a cluster, do not perform any operation on the ECS through the ECS console.

Constraints

- The cluster version must be 1.15 or later.
- Kunpeng nodes can be managed by clusters of v1.19 or later.
- You can manage ECS, BMS, and DeH nodes, but not HECS nodes.
- 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.
- If a password or key has been set when the original VM node was created, reset the password or key during management. The original password or key will become invalid.
- 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 node flavors that can be selected on the console when you create a node.
- The Ubuntu OS is not supported when BMS nodes are managed.
- 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.

Prerequisites

A cloud server that meets the following conditions can be accepted:

- 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. 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 configurations can be added in batches.

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 3-56 Node configuration parameters

Parameter	Description
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**.
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**.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 3-57 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.
Data Disk	**At least one data disk is required** for the container runtime and kubelet. **The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.** 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 **3.3.8.2 Data Disk Space Allocation**. 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** and **local EVs**.

Advanced Settings

Table 3-58 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined 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 node 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 in the YAML files of pods. Otherwise, scale-up 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	Pre-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	Pre-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

3.3.6 Logging In to a Node

Constraints

- If you use SSH to log in to a node (an ECS), ensure that the ECS already has an EIP (a public IP address).
- 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 3-59 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**

3.3.7 Management Nodes

3.3.7.1 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 **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

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 3-60**.

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 3-60 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/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
node.kubernetes.io/container-engine	Container engine used by the node.
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.

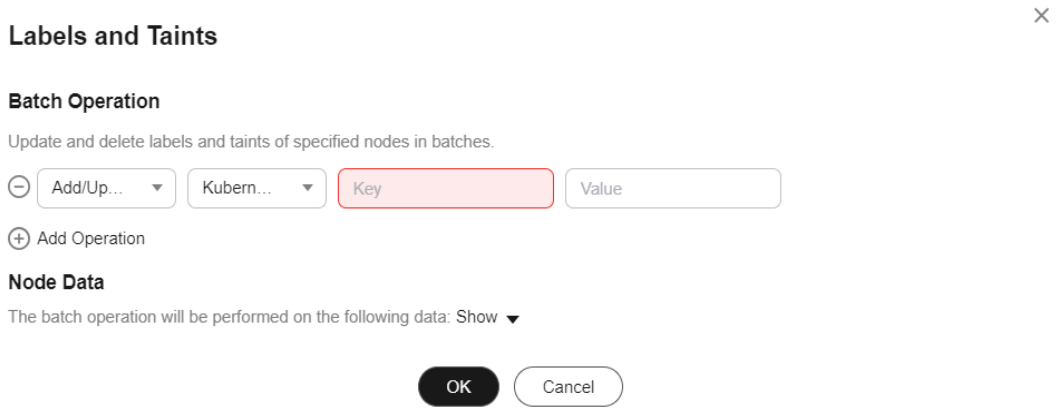
Step 2 In the navigation pane, choose **Nodes**. On the displayed page, click the **Nodes** tab, select the target node and click **Labels and Taints** in the upper left corner.

Step 3 In the displayed dialog box, click **Add batch operations** 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 3-45 Adding a node label



Step 4 After the label is added, check the added label in node data.

----End

3.3.7.2 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.

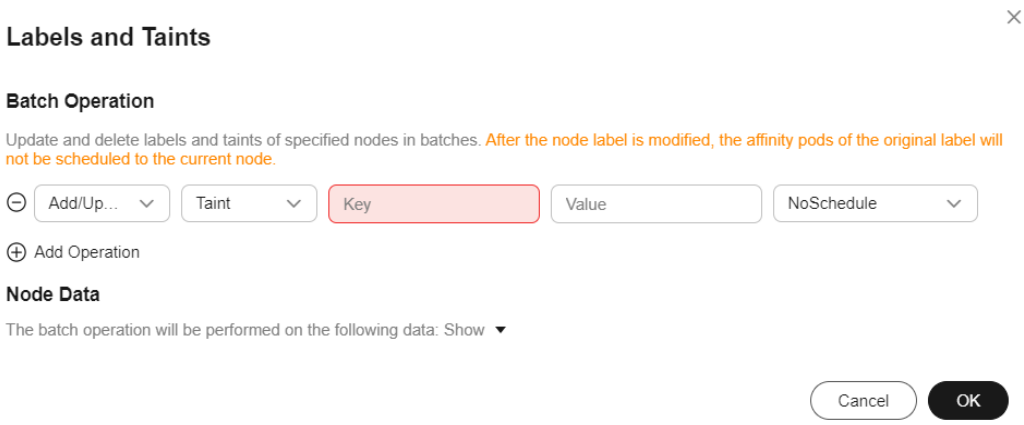
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

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 3-46 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

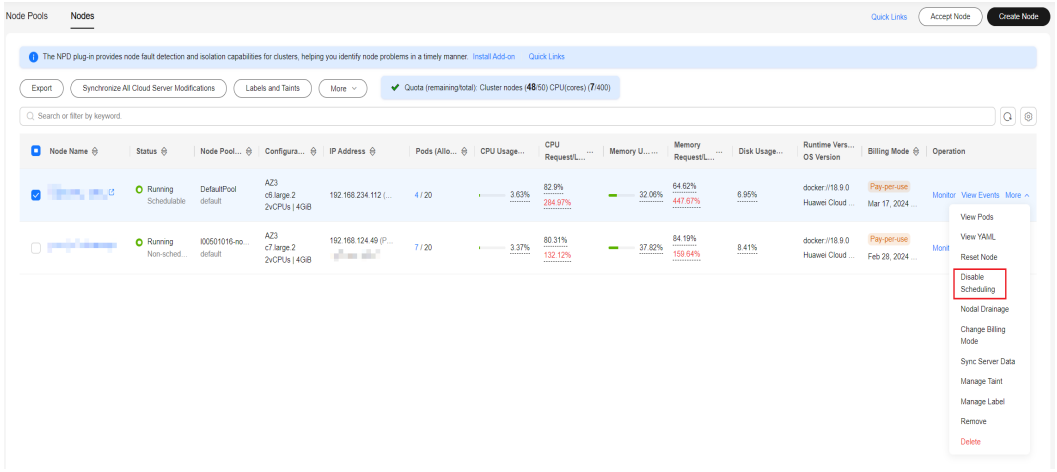
You can configure a node to be unschedulable on the console. Then, CCE will add a taint with key **node.kubernetes.io/unschedulable** and the **NoSchedule** setting to the node. After a node is set to be unschedulable, new pods cannot be scheduled to this node, but pods running on the node are not affected.

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.

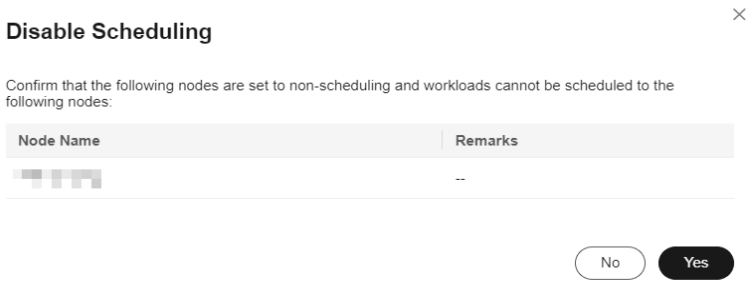
Step 3 In the node list, locate the target node and choose **More** > **Disable Scheduling** in the **Operation** column.

Figure 3-47 Disable Scheduling



Step 4 In the dialog box that is displayed, click **Yes** to configure the node to be unschedulable.

Figure 3-48 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"
```

3.3.7.3 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.

Constraints

- For CCE standard clusters and CCE Turbo clusters to support node resetting, 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.**
- **After a worker node with an extra data disk attached is reset, the attachment will be cleared. In this case, attach the disk again and data will be retained.**
- 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.
- While the node is being deleted, the backend will set the node to the unschedulable state.

- 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 reset 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.

Procedure

You can batch reset nodes using private images.

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 In the displayed dialog box, click **Next**.

- For nodes in the DefaultPool node pool, the parameter setting page is displayed. Set the parameters by referring to **Step 5**.
- For a node you create in a node pool, resetting the node does not support parameter configuration. You can directly use the configuration image of the node pool to reset the node.

Step 5 Specify node parameters.

Compute Settings

Table 3-61 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**.

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**.

Storage Settings

Configure storage resources on a node for the containers running on it.

Table 3-62 Configuration parameters

Parameter	Description
System Disk	Directly use the system disk of the cloud server.
Data Disk	**At least one data disk is required** for the container runtime and kubelet. **The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.** 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 **3.3.8.2 Data Disk Space Allocation**. 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** and **local EVs**.

Advanced Settings

Table 3-63 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined 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 node 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 in the YAML files of pods. Otherwise, scale-up 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	Pre-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	Pre-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

3.3.7.4 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.

Constraints

- Nodes can be removed only when the cluster is in the **Available** or **Unavailable** status.
- A CCE node can be removed only when it is in the **Active**, **Abnormal**, or **Error** status.
- A CCE node in the **Active** status can have its OS re-installed and CCE components cleared after it is removed.
- 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

- Removing 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.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- 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.

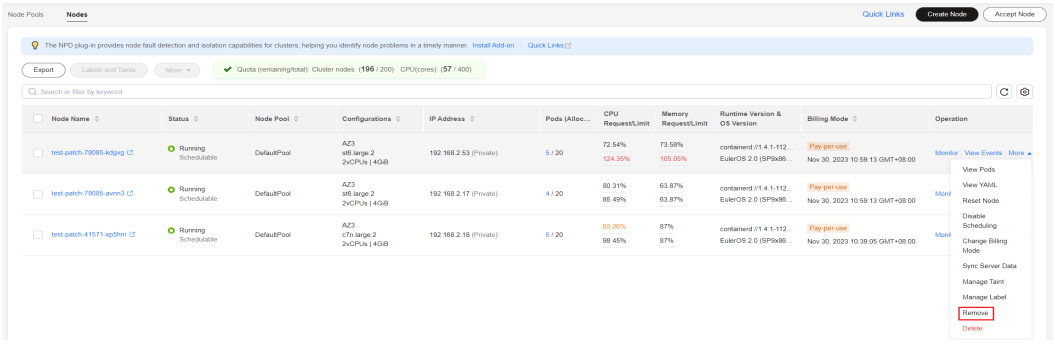
Procedure

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.

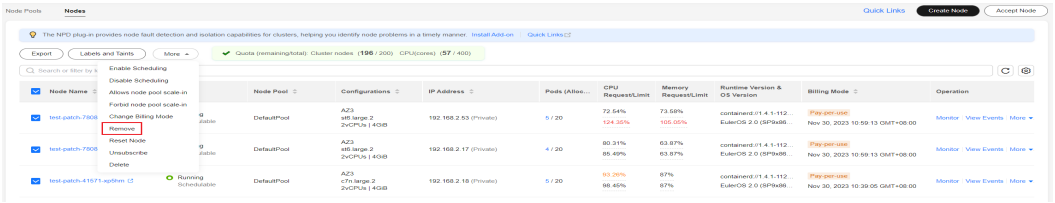
Step 3 Locate the target node and choose **More** > **Remove** in the **Operation** column.

Figure 3-49 Removing a node



You can also select multiple nodes and remove them at a time.

Figure 3-50 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

3.3.7.5 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**

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**.

- Data, such as the OS and image ID, cannot be synchronized. (Such parameters cannot be modified on the ECS console.)

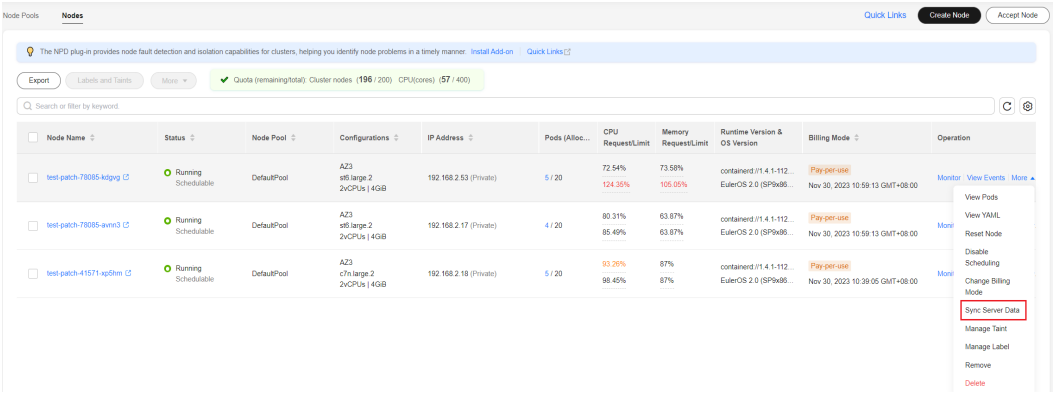
Synchronizing the Data of a Cloud Server

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 3-51 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**.

After the synchronization is complete, the **ECS data synchronization requested** message is displayed in the upper right corner.

----End

3.3.7.6 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 **Nodal Drainage Rules** 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.

Constraints

- Only clusters of the following versions support the nodal drainage function:
  - 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
- To use the nodal drainage function, an IAM user must have at least one of the following permissions. For details, see **3.16.3 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.
- If a **disruption budget** is not specify for the workload, the workload function may be unavailable during pod rescheduling.

Nodal Drainage Rules

The nodal drainage function securely evicts pods on a node. However, for pods that meet the following filtering criteria, the system performs the corresponding operations:

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
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 nodal 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

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)**: The drainage task automatically fails after the preset timeout period. The value 0 indicates that the timeout period is not set.
- **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 **Nodal Drainage Rules**.

Step 5 Click **OK** and wait until the nodal drainage is complete.

----End

3.3.7.7 Deleting a Node

Scenario

When a node in a CCE cluster is deleted, services running on the node will also be deleted. Exercise caution when performing this operation.

Constraints

- VM nodes that are being used by CCE do not support unsubscription or deletion on the ECS page.
- For a cluster of v1.17.11 or later, after a bare metal server is unsubscribed from or deleted on the BMS console, the corresponding node in the cluster is automatically deleted.
- 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 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.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- Only worker nodes can be deleted.
- Nodes in a yearly/monthly node pool will be automatically migrated to the default node pool before they are unsubscribed from. Exercise caution when performing this operation.

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** > **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

3.3.7.8 Changing the Billing Mode of a Node to Yearly/Monthly

CCE supports **pay-per-use** and **yearly/monthly** billing. A pay-per-use node can be changed to yearly/monthly-billed.

Constraints

- A pay-per-use node cannot be changed to a yearly/monthly one on the ECS console.
- Only nodes in the default node pool can be changed to the yearly/monthly billing mode.
- Nodes whose billing mode is changed to yearly/monthly do not support auto scaling.

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 **Changing the Billing Mode of an ECS from Pay-per-Use to Yearly/Monthly**.

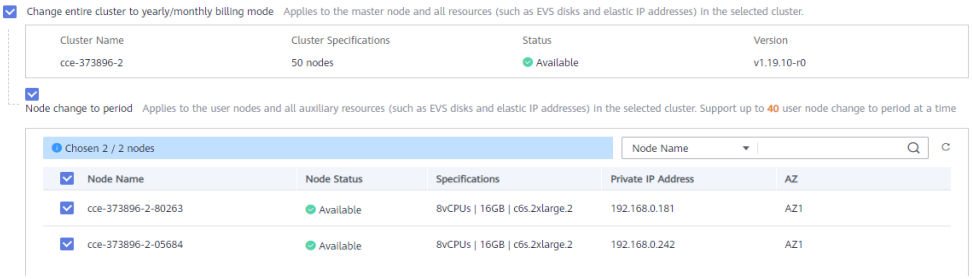
To change the billing mode of a pay-per-use node to yearly/monthly, perform the following steps:

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**. Locate the target node and choose **More** > **Change Billing Mode** in the **Operation** column.

Step 3 On the **Change Billing Mode** page, select the target node. The billing mode of a maximum of 40 nodes can be changed at a time.

Figure 3-52 Changing the billing mode of a node to yearly/monthly



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

----End

3.3.7.9 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.

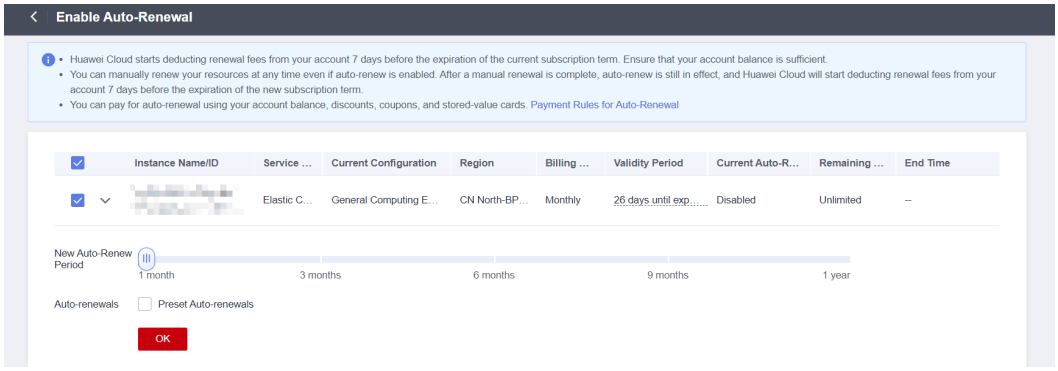
Step 2 In the navigation pane, choose **Nodes** and switch to the **Nodes** tab page.

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 3-53 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.

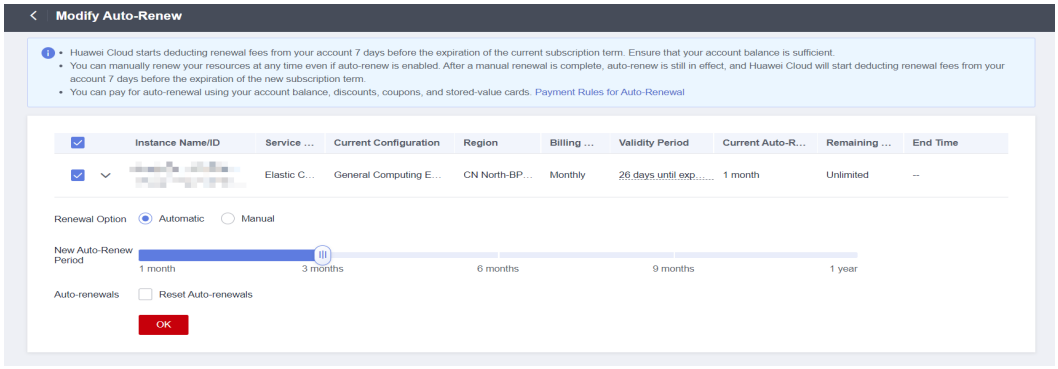
Step 2 In the navigation pane, choose **Nodes** and switch to the **Nodes** tab page.

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 3-54 Modifying auto-renewal



Step 5 Click **OK**.

----End

3.3.7.10 Stopping a Node

Scenario

After a node in the cluster is stopped, services on the node are also stopped. Before stopping a node, ensure that discontinuity of the services on the node will not result in adverse impacts.

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**.

Constraints

- Deleting 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.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- Only worker nodes can be stopped.

Procedure

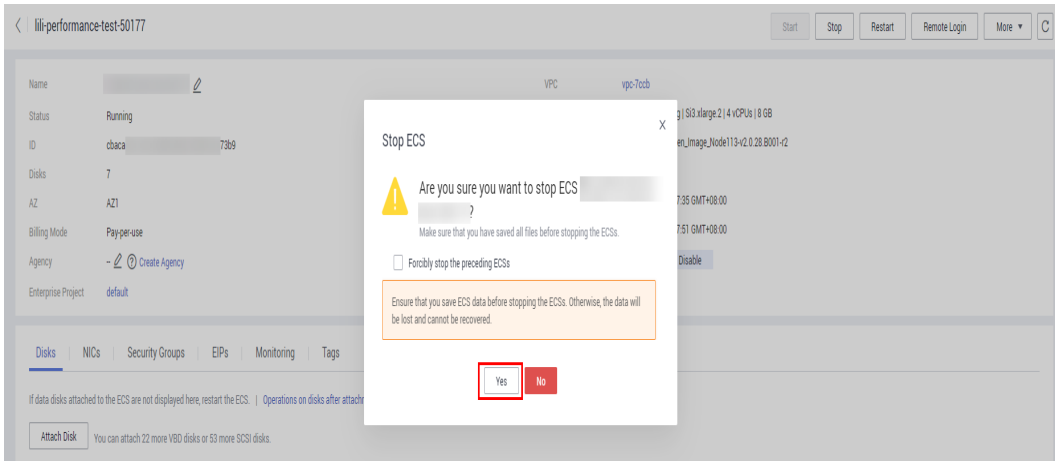
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

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 **Yes**.

Figure 3-55 ECS details page



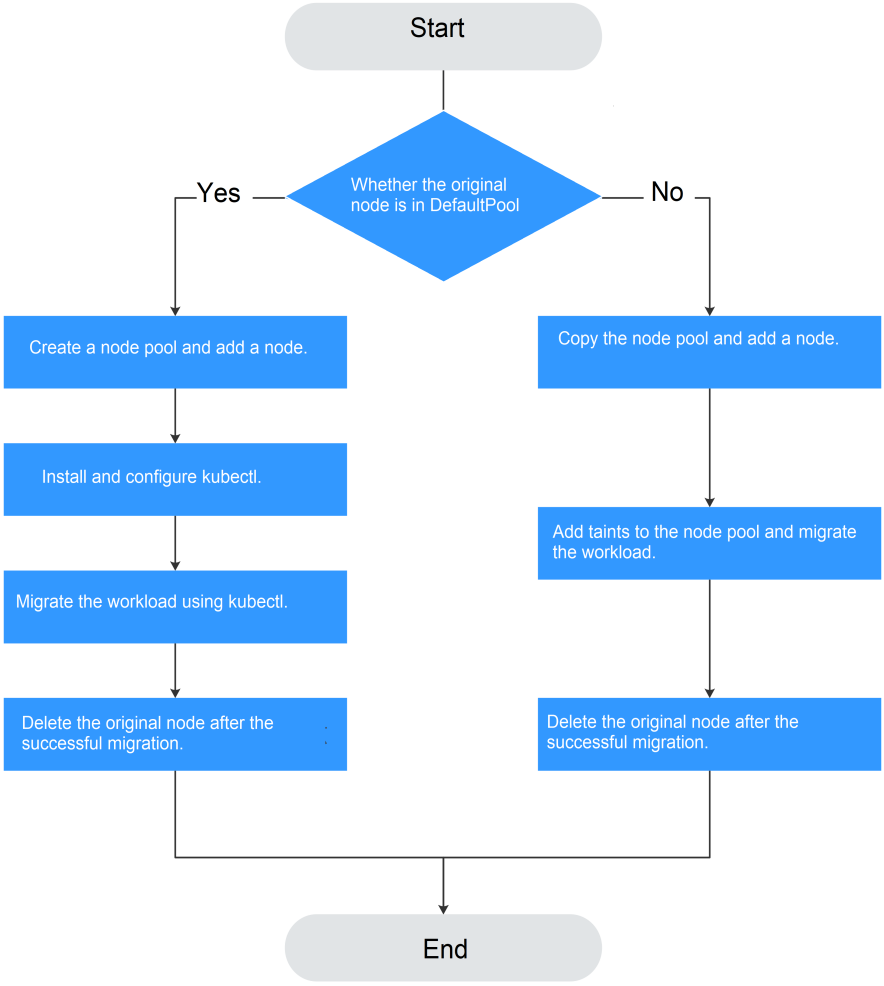
----End

3.3.7.11 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 3-56** shows the migration process.

Figure 3-56 Workload migration



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 DefaultPool does not support this configuration.

Scenario 1: The Original Node Is in DefaultPool

Step 1 Create a node pool. For details, see **3.4.2 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 **3.2.3.1 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 **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

After the workload is migrated, you can view that the workload is 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

Scenario 2: The Original Node Is Not in DefaultPool

Step 1 Copy the node pool and add nodes to it. For details, see **3.4.3.4 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 **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

After the workload is migrated, you can view that the workload is 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

3.3.8 Node O&M

3.3.8.1 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 MB.

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**.

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 3-64 Reservation rules for system components

Total Memory (TM)	Reserved Memory for System Components
TM ≤ 8 GB	0 MB
8 GB < TM ≤ 16 GB	[(TM – 8 GB) x 1024 x 10%] MB
16 GB < TM ≤ 128 GB	[8 GB x 1024 x 10% + (TM – 16 GB) x 1024 x 6%] MB
TM > 128 GB	(8 GB x 1024 x 10% + 112 GB x 1024 x 6% + (TM – 128 GB) x 1024 x 2%) MB

Table 3-65 Reservation rules for kubelet

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
TM ≤ 2 GB	None	TM x 25%

Total Memory (TM)	Number of Pods	Reserved Memory for kubelet
TM > 2 GB	0 < Max. pods on a node ≤ 16	700 MB
	16 < Max. pods on a node ≤ 32	[700 + (Max. pods on a node – 16) x 18.75] MB
	32 < Max. pods on a node ≤ 64	[1024 + (Max. pods on a node – 32) x 6.25] MB
	64 < Max. pods on a node ≤ 128	[1230 + (Max. pods on a node – 64) x 7.80] MB
	Max. pods on a node > 128	[1740 + (Max. pods on a node – 128) x 11.20] MB

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 **3.4.3.3 Configuring a Node Pool**.

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 3-66 Rules for reserving node memory v2

Reserved for	Basic/Floating	Reservation	Used by
OS	Basic	400 MB (fixed)	OS service components such as sshd and systemd-journald.
	Floating (depending on the node memory)	25 MB/GB	Kernel

Reserved for	Basic/Floating	Reservation	Used by
CCE	Basic	500 MB (fixed)	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 MB/pod containerd: 5 MB/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 3-70**.

Rules for Reserving Node CPU

Table 3-67 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.

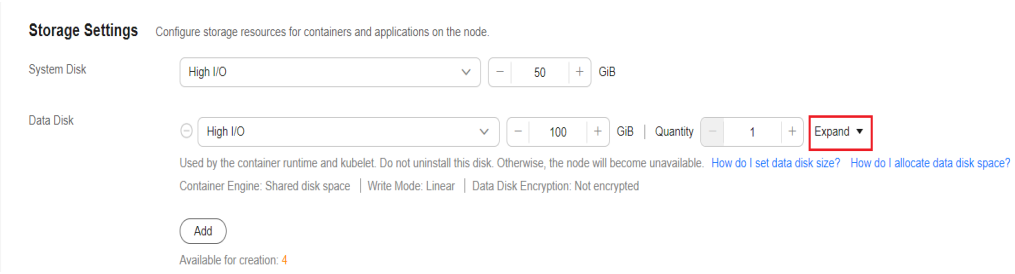
3.3.8.2 Data Disk Space Allocation

This section describes how to allocate data disk space to nodes so that you can configure the data disk space accordingly.

Allocating Data Disk Space

When creating a node, configure data disks for the node. You can also click **Expand** and customize the data disk space allocation for the node.

Figure 3-57 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 images, 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.
  - 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**.

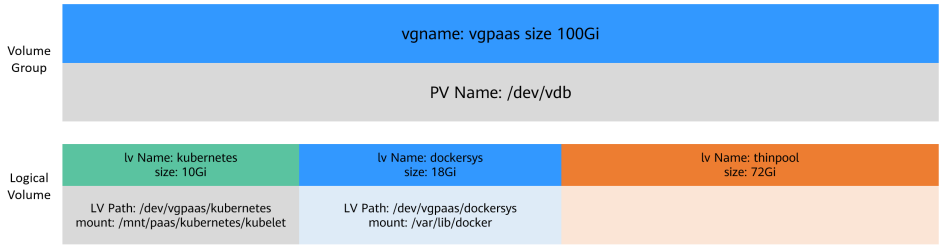
- **Rootfs (Device Mapper)**

  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.

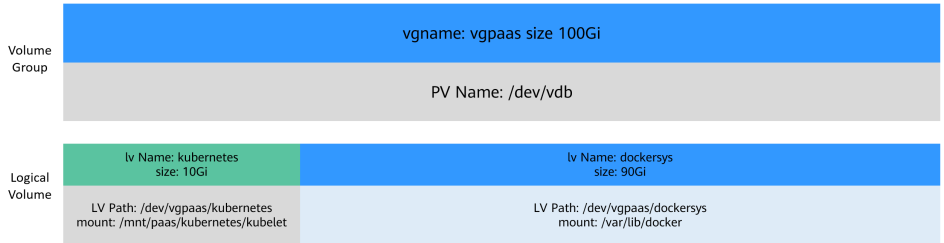
Figure 3-58 Space allocation for container engines of Device Mapper



- **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 3-59 Space allocation for container engines of OverlayFS



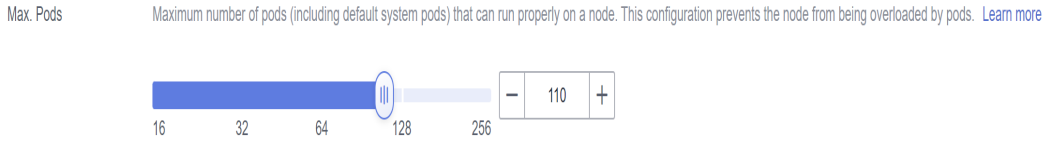
Space Allocation for Pods

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 on a node. The container engine space should be greater than the total disk space used by containers. Formula: **the 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 3-60 Maximum number of pods



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 GB 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 GB, the thin pool space occupied by files keeps increasing until 10 GB 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 3-68 Node OSs and container engines in CCE clusters

OS	Container Storage Rootfs	Customized 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 container engine is Docker. The default value is 10 GiB. Not supported when Rootfs is set to OverlayFS.

OS	Container Storage Rootfs	Customized Basesize
EulerOS 2.3	Device Mapper	Supported only when the container engine is Docker. The default value is 10 GiB.
EulerOS 2.5	Device Mapper	Supported only when the container engine 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 container engine is Docker. The default value is 10 GiB. Supported only when Rootfs is set to OverlayFS and the container engine is Docker. There are no limits by default.
EulerOS 2.9	OverlayFS	Supported only by clusters of v1.19.16, v1.21.3, v1.23.3, or later. There are no limits by default. Not supported if the cluster versions are earlier than v1.19.16, v1.21.3, or v1.23.3.
EulerOS 2.10	OverlayFS	Supported only by Docker clusters of versions 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 versions. There are no limits by default.
Ubuntu 18.04	OverlayFS	Not supported.
Huawei Cloud EulerOS 1.1	OverlayFS	Not supported.
Huawei Cloud EulerOS 2.0	OverlayFS	Supported only by Docker clusters of versions 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 versions. There are no limits by default.

Table 3-69 Node OSs and container engines in CCE Turbo clusters

OS	Container Storage Rootfs	Customized Basesize
CentOS 7.x	OverlayFS	Not supported.
Ubuntu 18.04	OverlayFS	Not supported.
EulerOS 2.9	ECS VMs use OverlayFS. ECS PMs use Device Mapper.	Supported only when Rootfs is set to OverlayFS and the container engine 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 versions. Supported when Rootfs is set to Device Mapper and the container engine is Docker. The default value is 10 GiB.
Huawei Cloud EulerOS 1.1	OverlayFS	Not supported.
Huawei Cloud EulerOS 2.0	OverlayFS	Supported only by Docker clusters of versions 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 versions. 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.

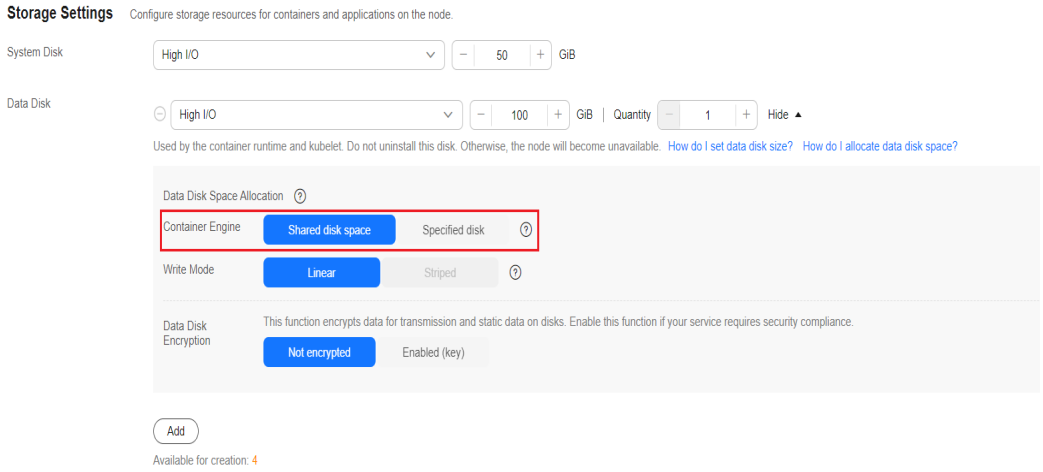
Data Disk Shared Between a Container Engine and kubelet Components

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**.

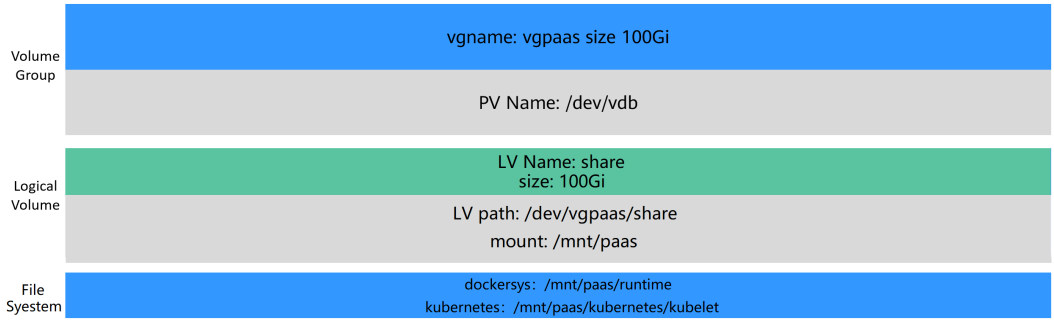
Figure 3-61 Configuration for sharing disk space



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 3-62 Allocating the storage space of a shared data disk



Common Issues

How Do I Expand the Storage Capacity of a Container?

Expanding the Disk Capacity of a Node in a CCE Cluster

3.3.8.3 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:

- For a cluster using the container tunnel network model, the value depends only on **the maximum number of pods on a node**.
- For clusters using the VPC network model, the value depends on **the maximum number of pods on a node** and **the minimum number of container IP addresses that can be allocated to a node**. It is recommended that the maximum number of pods on a node be less than or equal to the number of container IP addresses that can be allocated to the node. Otherwise, pods may fail to be scheduled.
- For CCE Turbo clusters using the Cloud Native Network 2.0 model, the value depends on **the maximum number of pods on a node** and **the minimum number of ENIs on a CCE Turbo cluster node**. It is recommended that the maximum number of pods on a node be less than or equal to the number of ENIs on the node. Otherwise, pods may fail to be scheduled.

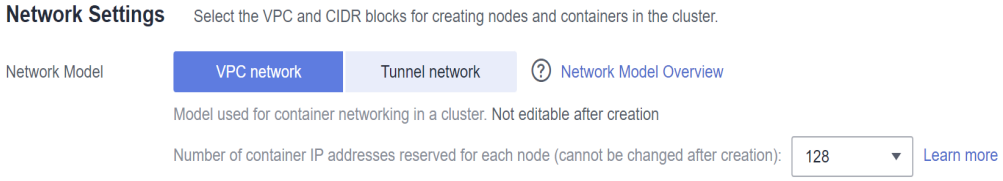
Number of Container IP Addresses That Can Be Allocated on a Node

If you select **VPC network** for **Network Model** when creating a CCE cluster, you also need to set the number of container IP addresses that can be allocated to

each node (alpha.cce/fixPoolMask). If the pod uses the host network (**hostNetwork: true**), the pod does not occupy the IP address of the allocatable container network. For details, see **Container Network vs. Host Network**.

This parameter affects the maximum number of pods that can be created on a node. Each pod occupies an IP address (when the **container network** is used). If the number of available IP addresses is insufficient, pods cannot be created. If the pod uses the host network (**hostNetwork: true**), the pod does not occupy the IP address of the allocatable container network.

Figure 3-63 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 3-64 Specifying the maximum number of pods on a node

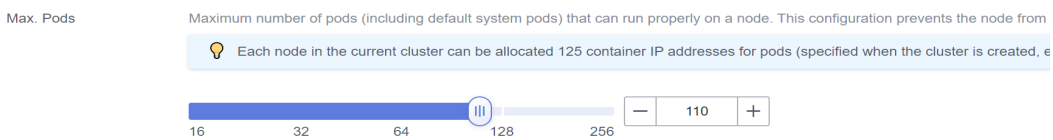


Table 3-70 lists the default maximum number of pods on a node based on node specifications.

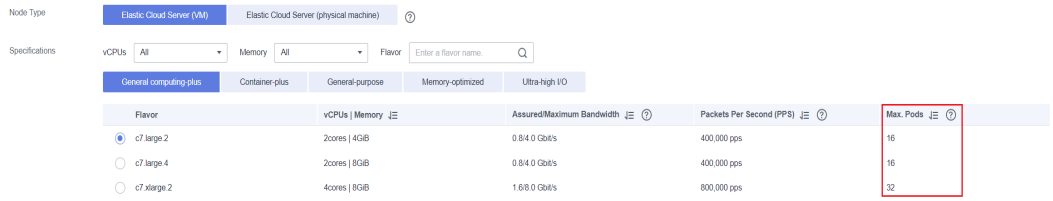
Table 3-70 Default maximum number of pods on a node

Memory	Max. Pods
4 GB	20
8 GB	40
16 GB	60
32 GB	80
64 GB or above	110

Number of Node ENIs (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 3-65 Node ENIs



Container Network vs. 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: The pod uses the host network (**hostNetwork: true** needs to be configured for the pod) and occupies the host port. The pod IP address is the host IP address. The pod does not occupy the IP addresses of the container network. To use the host network, you must confirm whether the container ports conflict with the host ports. Do not use the host network unless you know exactly which host port is used by which container.

3.3.8.4 Migrating Nodes from Docker to containerd

Kubernetes has removed dockershim from v1.24 and does not support Docker by default. CCE is going to stop the support for Docker. Change the node container engine from Docker to 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 **3.3.2 Container Engine**.

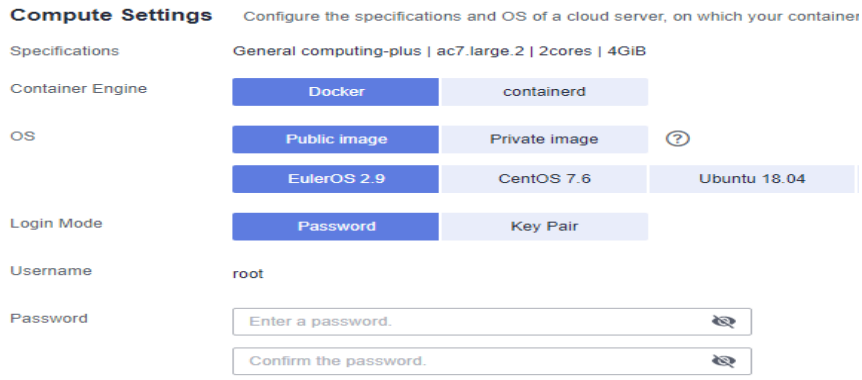
Migrating a Node

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 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.

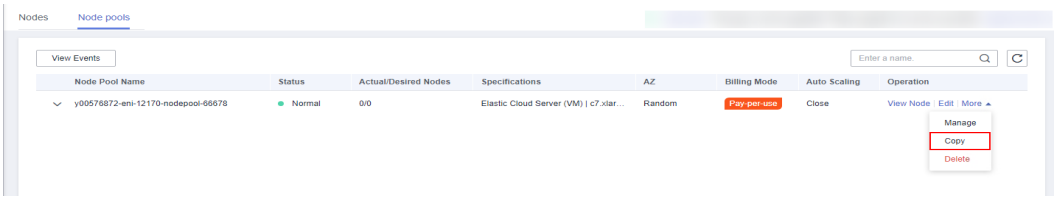
----End

Migrating a Node Pool

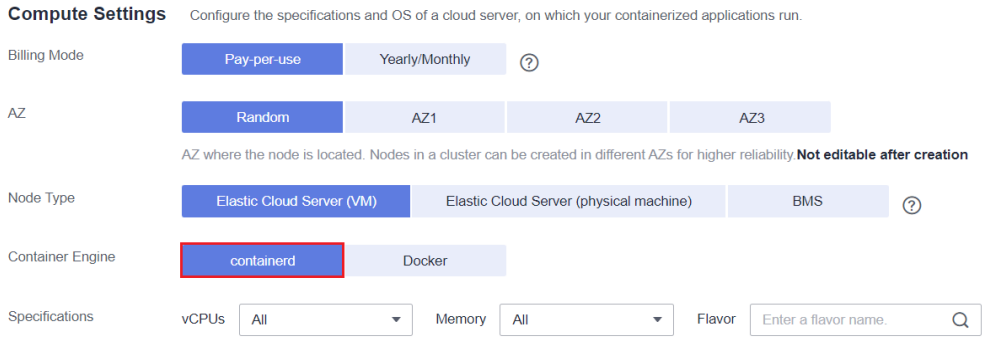
You can **copy a node pool**, set the container engine of the new node pool to containerd, and keep other configurations the same as those of the original Docker node pool.

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 On the **Compute Settings** area, set **Container Engine** to **containerd** and modify other parameters as required.



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 set node affinity for the workloads deployed on the original Docker nodes or node pool, set 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

3.3.8.5 Optimizing Node System Parameters

3.3.8.5.1 List of Node System Parameters That Can Be Optimized

CCE provides default node system parameters, which may cause performance bottlenecks in some scenarios. Therefore, you can customize and optimize some node system parameters. **3.3.8.5.1 List of Node System Parameters That Can Be Optimized** describes the node system parameters.

NOTICE

- The modification has certain risks. Be familiar with Linux commands and Linux OS.
- The parameters listed in **Table 3-71** 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 3-71 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`	**3.3.8.5.5 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`	**3.3.8.5.2 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`
			3.3.8.5.4 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 RST packets that are not in the TCP window are marked as invalid. • **1**: The function is enabled. Only RST 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 in the **Close_wait** phase for a long time, exhausting 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`	

3.3.8.5.2 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.

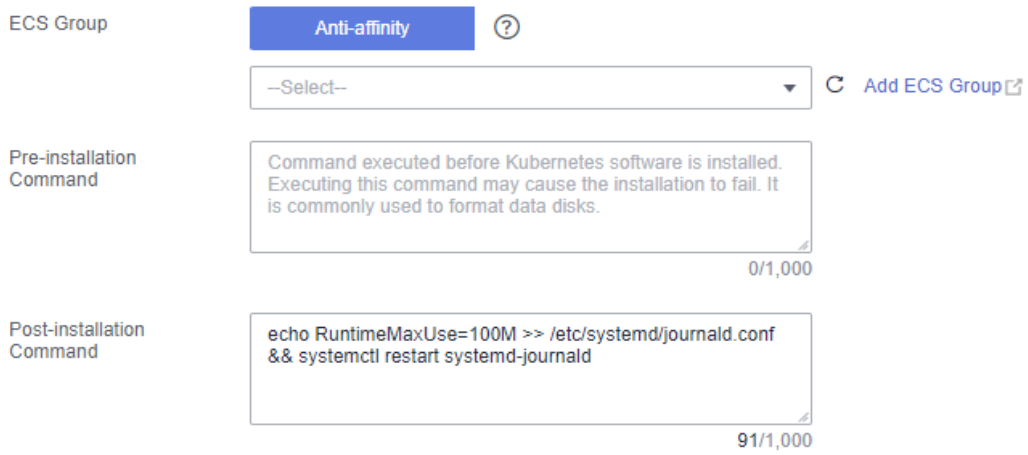
Step 1 Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

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

3.3.8.5.3 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

3.3.8.5.4 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 3-72 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 RST packets that are not in the TCP window are marked as invalid. • **1**: The function is enabled. Only RST 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, exhausting 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 3-72 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 3-72.

To modify other kernel parameters, replace the parameter names and values in the commands by referring to Table 3-72.

- 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 3-72 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 3-72.

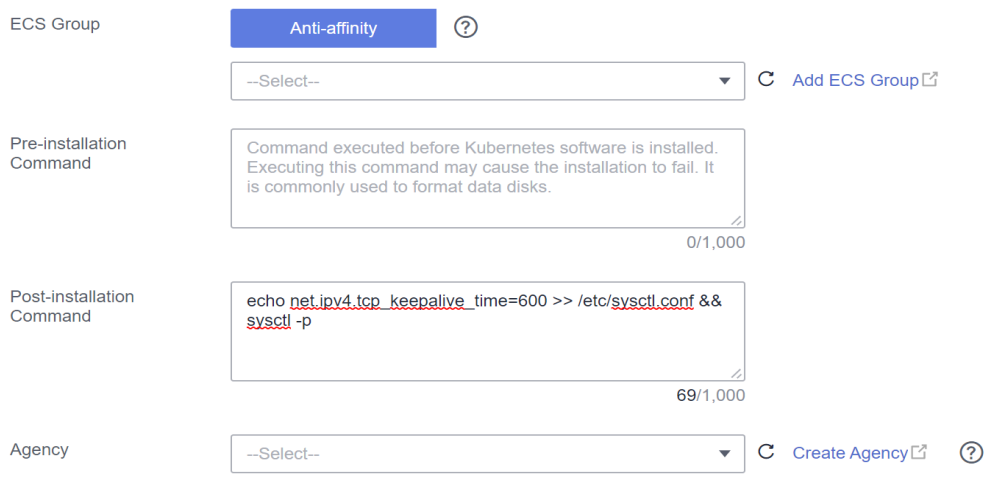
Step 1 Confirm the OS of the node or node pool to be created, for example, CentOS 7.6.

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 3-72**.

- 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 3-72** to check whether the modification is successful.

----End

3.3.8.5.5 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 3-73 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

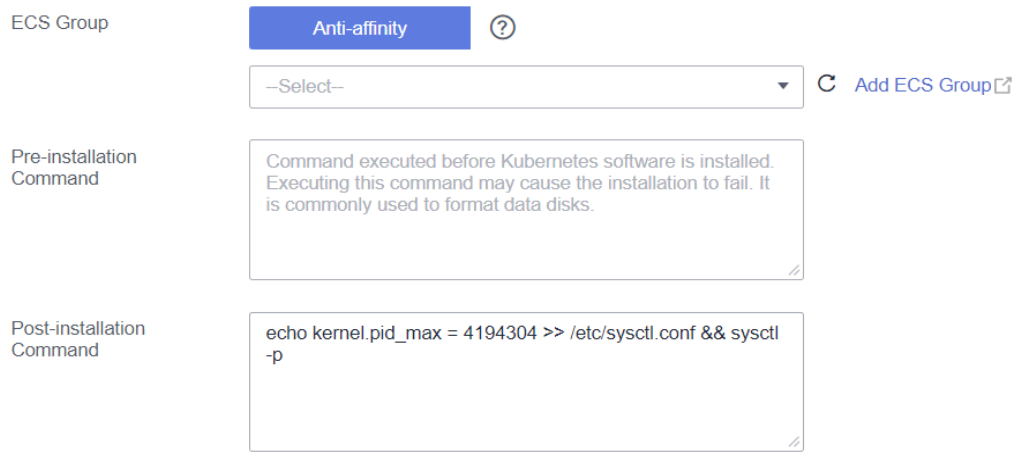
EulerOS 2.3: Configuration required.

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

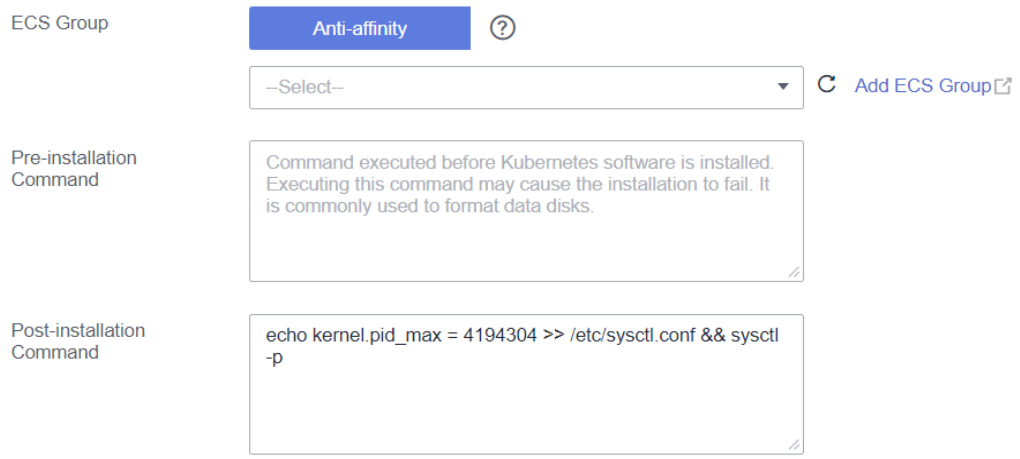
EulerOS 2.3: Configuration required.

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



3.3.8.6 Node Fault Detection Policy

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 **3.14.4 CCE Node Problem Detector** add-on has been installed in the cluster.

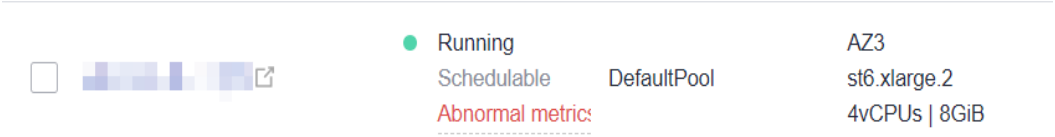
Enabling Node Fault Detection

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

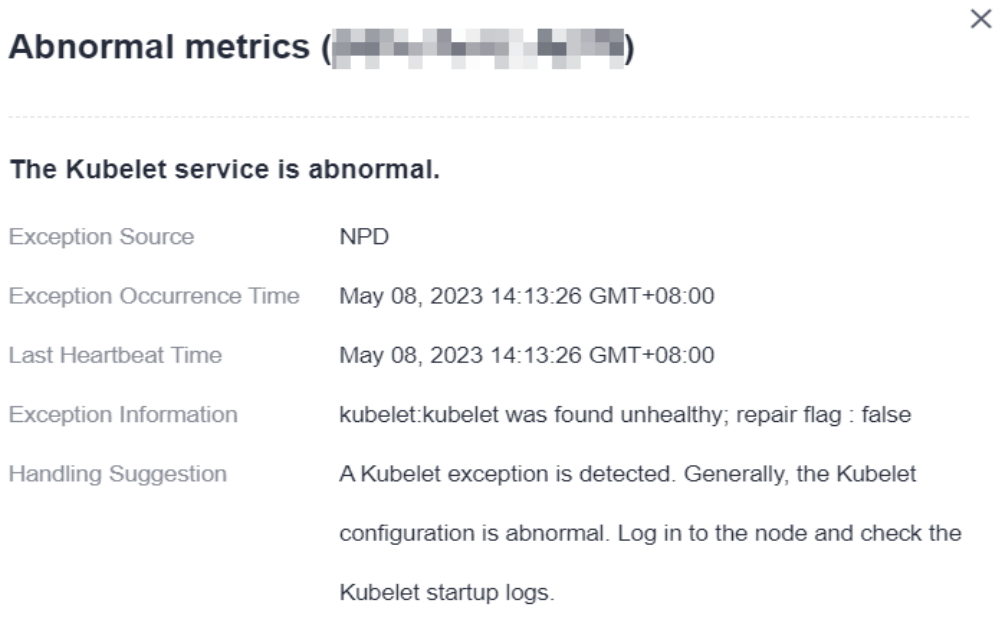
Step 2 In the navigation pane, choose **Nodes** and then click the **Nodes** tab. Check whether the NPD add-on has been installed in the cluster or whether the add-on has been upgraded to the latest version. After the NPD add-on has been installed, you can use the fault detection function.

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

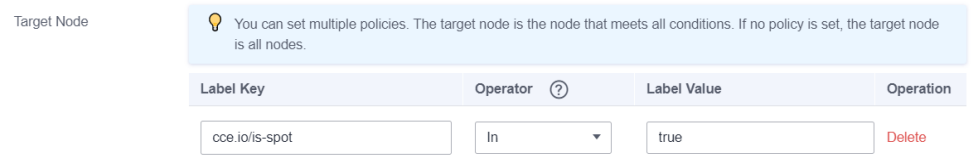
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

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 3-74 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 3-75 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 3-76 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 3-77 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 3-78 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 a temporary 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"`
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` • Threshold: – Average usage: ioutil >= 0.99 – Average I/O queue length: avgqu-sz >= 1 – Average I/O transfer volume: iops (w/s) + ioth (wMB/s) <= 1 **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 3-79 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 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 3-80 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.

Check Item	Function	Description
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%

3.4 Node Pools

3.4.1 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.

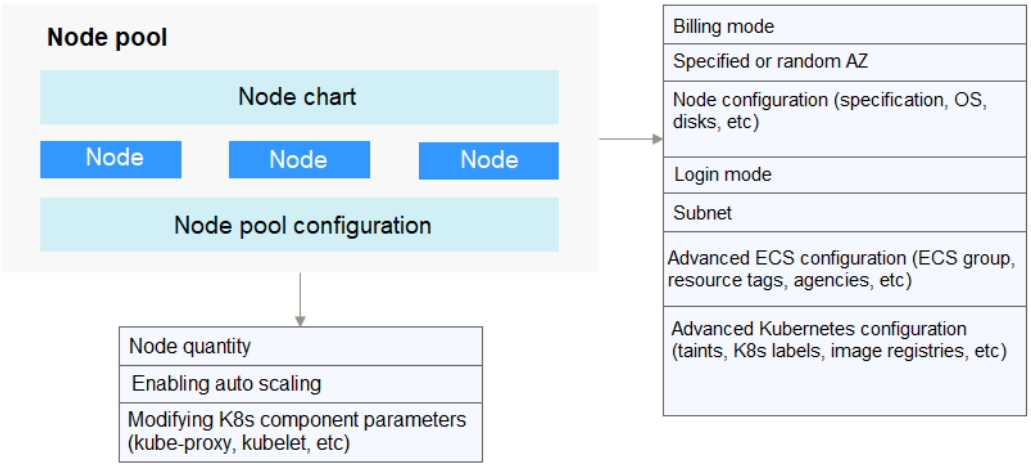
You can also use node pools for auto scaling (supported only by pay-per-use node pools).

- 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 3-66 Overall architecture of a node pool



Generally, all nodes in a node pool have the following same attributes:

- Node OS
- Node flavor
- Node login mode
- Node container runtime
- Startup parameters of Kubernetes components on a node
- User-defined 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

Description of DefaultPool

DefaultPool is not a real node pool. It only **classifies** nodes that are not in the user-created 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 3-81 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.
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	When a node pool is deleted, the nodes in the node pool are deleted first. When a yearly/monthly node pool is deleted, the nodes are migrated to the default node pool first. Workloads on the original nodes are 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.
Enabling 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.

Function	Description	Precaution
Changing node pool configurations	You can modify the node pool name, node quantity, Kubernetes labels (and their quantity), resource tags, and taints.	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.
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 and later. It is not displayed for versions earlier than v1.15. • The default node pool DefaultPool does not support this type of configuration.

Deploying a Workload in a Specified Node Pool

When creating a workload, you can constrain pods to run in a specified node pool.

For example, on the CCE console, you can set the affinity between the workload and the node on the **Scheduling Policies** tab page on the workload details page to forcibly deploy the workload to a specific node pool. In this way, the workload runs only on nodes in the 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 **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

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:

- **3.4.2 Creating a Node Pool**
- **3.4.3 Managing a Node Pool**
- **3.5.2.1 Creating a Deployment**
- **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**

3.4.2 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 **3.4.1 Node Pool Overview**.

Constraints

- The Autoscaler add-on needs to be installed for node auto scaling. For details about the add-on installation and parameter configuration, see **3.14.6 CCE Cluster Autoscaler**.
- Auto scaling is available only for pay-per-use node pools, not for those billed by year or month.
- Only clusters of v1.19 or later support custom security groups.

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 3-82 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.

Parameter	Description
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 (for example, **default**) is selected, node resources can be managed through the enterprise project, but enterprise project management permissions are not supported. The default value is the enterprise project selected for the cluster. Enterprise projects facilitate project-level management and grouping of cloud resources and users. For more details, see **Enterprise Management**.
Expected Initial Nodes	Number of nodes to be created in this node pool. A maximum of 50 nodes that can be created at a time.

Configurations

You can configure the flavor and OS of a cloud server, on which your containerized applications run.

Table 3-83 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.

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 or AZs. 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 multiple node flavors, you can select different AZs. • 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**.

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 3-84 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. **Encryption**: System disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **This function is available only in certain regions.** • **Encryption** is not selected by default. • After setting **System Disk Encryption** to **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 **Encryption** text box.

Parameter	Description
Data Disk	**At least one data disk is required** for the container runtime and kubelet. **The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.** • First data disk: used for container runtime and kubelet components. The value ranges from 20 GiB to 32768 GiB. The default value is 100 GiB. • Other 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. **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 **3.3.8.2 Data Disk Space Allocation**. • **Encryption**: Data disk encryption safeguards your data. Snapshots generated from encrypted disks and disks created using these snapshots automatically inherit the encryption setting. **This function is available only in certain regions.** – **Encryption** is not selected by default. – After selecting **Encryption**, you can select an existing key in the displayed dialog box. 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 **Encryption** text box. **Adding data disks** A maximum of four data disks can be added. By default, raw disks are 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**.

Parameter	Description
	• **Use as ephemeral volume**: applicable when there is a high performance requirement on EmptyDir. **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 PVs** and **Local 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 multiple volumes exist.

Network Settings

Configure networking resources to allow node and containerized application access.

Table 3-85 Configuration parameters

Parameter	Description
Virtual Private Cloud	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.
Node IP Address	Random allocation is supported.

Parameter	Description
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 **Configuring Cluster Security Group Rules**. **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 3-86 Advanced configuration parameters

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined 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**. 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 **3.3.7.2 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.
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.

Parameter	Description
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 **3.3.8.3 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	Pre-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	Pre-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 account administrator on the IAM console. By creating 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.

Parameter	Description
User-defined node name 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.
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. If you want to change the Kubernetes node name to cloud server name, you can remove this node from the cluster and accept it again. Before doing so, learn about the possible impacts on services when **removing a node** and **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

3.4.3 Managing a Node Pool

3.4.3.1 Updating a Node Pool

Constraints

- 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.
- The modification of resource tags, container engine, pre-installation and post-installation scripts, or OS of a node pool takes effect only on new nodes. To synchronize the modification onto existing nodes, manually reset the existing 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 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 3-87 Basic settings

Parameter	Description
Node Pool Name	Name of the node pool.
Expected Nodes	Change the number of nodes based on service requirements.

Configurations

Table 3-88 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 multiple node flavors, you can select different AZs. • 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.

Storage Settings

Table 3-89 Configuration parameters

Parameter	Description
System Disk	System disk used by the node OS. The disk size 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. **This function is available only in certain regions.** • **System Disk Encryption** is not selected by default. • After setting **System Disk Encryption** to **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 key 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 data disk is required** for the container runtime and kubelet. **The data disk cannot be deleted or uninstalled. Otherwise, the node will be unavailable.** • First 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 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. **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 **3.3.8.2 Data Disk Space Allocation**. **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. **This function is available only in certain regions.** – **Not encrypted** is not 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 key text box. **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 four data disks can be added. By default, raw disks are 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-**

Parameter	Description
	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. **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 PVs** and **local EVs** can be written in the following 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 multiple volumes exist. **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 3-90 Configuration parameters

Parameter	Description
Virtual Private Cloud	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 3-91 Advanced settings

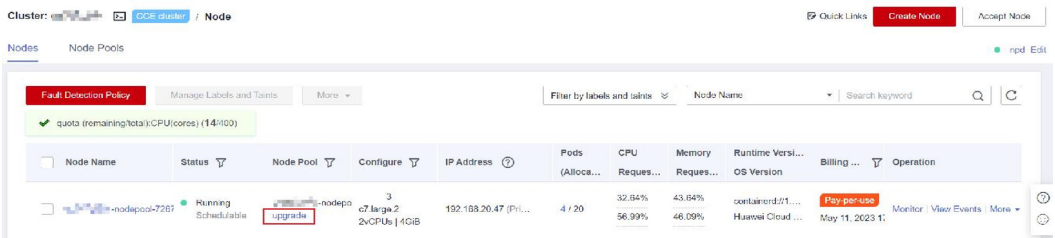
Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the TMS console. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency. For 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** 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**.
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 **3.3.7.2 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.
Edit key pair	Only node pools that use key pairs for login support key pair editing. You can select another key pair. **NOTE** The edited key pair automatically takes effect on newly added nodes. For existing nodes, manually reset the nodes for the modification to take effect.

Step 4 After the configuration, click **OK**.

After the node pool parameters are updated, go to the **Nodes** page to check whether the node to which the node pool belongs is updated. You can reset the node to synchronize the configuration updates for the node pool.



----End

3.4.3.2 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.

Constraints

To enable AS, the **3.14.6 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 the auto scaling add-on 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 **3.14.6 CCE Cluster Autoscaler**.
- If the auto scaling add-on has been installed, directly configure auto 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 3-92 Custom rules

Rule Type	Configuration
Metric-based	– **Trigger**: Select **CPU allocation rate** or **Memory allocation rate** and enter a value. The value must be greater than the scale-in percentage configured in the auto scaling add-on. **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 based on the **CPU allocation rate** and **a periodic rule** are configured and they both meet the scale-out conditions, one of them will be executed randomly. The rule executed first (rule A) changes the node pool to the scaling state. As a result, the other rule (rule B) cannot be executed. After rule A is executed and the node pool status becomes normal, rule B will not be executed. ▪ If rules based on the **CPU allocation rate** and **memory allocation rate** are configured, 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. ▪ When the number of nodes in the cluster reaches the upper limit, or the CPU or memory usage reaches the upper limit of the autoscaler add-on, node 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
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

Specification selection: Configure whether to enable auto scaling for node flavors in a node pool.

Step 4 Click **OK**.

----End

3.4.3.3 Configuring a Node Pool

Constraints

The default node pool DefaultPool does not support the following management operations.

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 and 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 Components** page on the right, change the values of Kubernetes parameters.

Table 3-93 kubelet

Item	Parameter	Description	Value	Modification
CPU management policy	cpu-manager-policy	CPU management policy configuration. For details, see **3.6.2 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	None
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

Item	Parameter	Description	Value	Modification
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
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

Item	Parameter	Description	Value	Modification
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 Pod Security Policy**.	Default: []	None

Item	Parameter	Description	Value	Modification
Node oversubscription	over-subscription-resource	Whether to enable node oversubscription. If this parameter is set to **true**, node oversubscription is enabled on nodes. For details, see **3.6.6.1 Dynamic Resource Oversubscription**.	• For clusters of versions earlier than v1.23.9-r0 or v1.25.4-r0: enabled (**true**) by default • Disabled by default if the cluster version is v1.23.9-r0, v1.25.4-r0, v1.27-r0, v1.28.1-r0, or later	None
Hybrid deployment	colocation	Whether to enable hybrid deployment on nodes. If this parameter is set to **true**, hybrid deployment is enabled on nodes. For details, see **3.6.6.1 Dynamic Resource Oversubscription**.	• For clusters of versions earlier than v1.23.9-r0 or v1.25.4-r0: enabled (**true**) by default • Disabled by default if the cluster version is v1.23.9-r0, v1.25.4-r0, v1.27-r0, v1.28.1-r0, or later	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 The value ranges from 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	-
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	-
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**.
Lower Limit for Image Garbage Collection	image-gc-low-threshold	When the disk usage reduces to this value, image garbage collection stops.	Default: 70 Value range: 1 to 100	The value of this parameter cannot be greater than the upper limit for image garbage collection.

Item	Parameter	Description	Value	Modification
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 **3.3.8.1 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.		

Table 3-94 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 -w net.nf_conntrack_max`	Default: 131072	None
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 -w net.netfilter.nf_conntrack_tcp_timeout_close_wait`	Default: 1h0m0s	None

Table 3-95 Network 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.
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 current number of ENIs bound to the node is greater than the **nic-maximum-target** value, CCE will pre-bind on 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.

Table 3-96 Pod security group in a node pool (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

Table 3-97 Docker (available only for node pools that use Docker)

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.

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: {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.
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.

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.

Table 3-98 containerd (available only for node pools that use containerd)

Item	Parameter	Description	Value	Modification
Available data space for a single container	devmapper-base-size	Maximum data space that can be used by each container.	Default: 0	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
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`

Item	Parameter	Description	Value	Modification
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.
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.

Step 5 Click **OK**.

----End

3.4.3.4 Copying a Node Pool

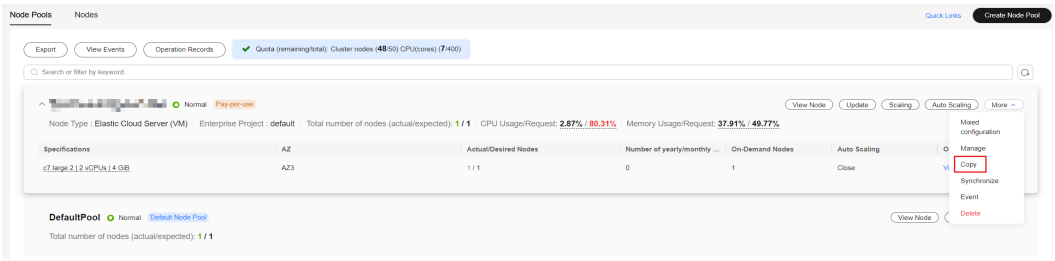
You can copy the configuration of an existing node pool on the CCE console to create new node pools.

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 3-67 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 **3.4.2 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

3.4.3.5 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

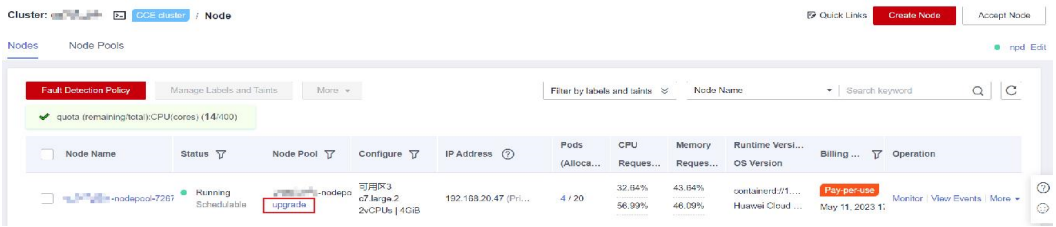
- Do not delete or reset nodes during batch synchronization. Otherwise, the synchronization of node pool configuration may fail.
- This operation involves resetting nodes. **Workloads running on a node may be interrupted due to standalone deployment or insufficient schedulable resources.** Evaluate the 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.
  - The modification of resource tags, container engine, pre-installation and post-installation scripts, or OS of a node pool takes effect only on new nodes. To synchronize the modification onto existing nodes, manually reset the existing 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 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

3.4.3.6 Upgrading an OS

When CCE releases a new OS image, existing nodes cannot be automatically upgraded. You can manually upgrade them in batches.

NOTICE

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.

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**.
- Pre-installation script:

  Pre-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:

  Pre-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

3.4.3.7 Modifying the Auto-Renewal Configuration of a Yearly/Monthly Node Pool

Node pools billed on a yearly/monthly basis support auto-renewal. To change the auto-renewal period, follow the instructions provided in this section.

Constraints

After the auto-renewal configuration of a yearly/monthly node pool is modified, the modification takes effect only on new nodes. To apply the modification onto existing nodes, manually modify the auto-renewal configuration. For details, see **3.3.7.9 Modifying the Auto-Renewal Configuration of a Yearly/Monthly Node**.

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**. Click **Update** in the **Operation** column of the target node pool.

Step 3 On the **Update Node Pool** page, enable or disable auto-renewal as needed.

NOTE

- If the original node pool is purchased by month, the subsequent renewal period is one month after auto-renewal is enabled.
- If the original node pool is purchased by year, the subsequent renewal period is one year after auto-renewal is enabled.

Step 4 Click **Next: Confirm**, confirm the modification, and click **Submit**.

----End

3.4.3.8 Migrating a Node

Nodes in a node pool can be migrated to the default node pool. Nodes in the default node pool or a custom node pool cannot be migrated to other custom node pools.

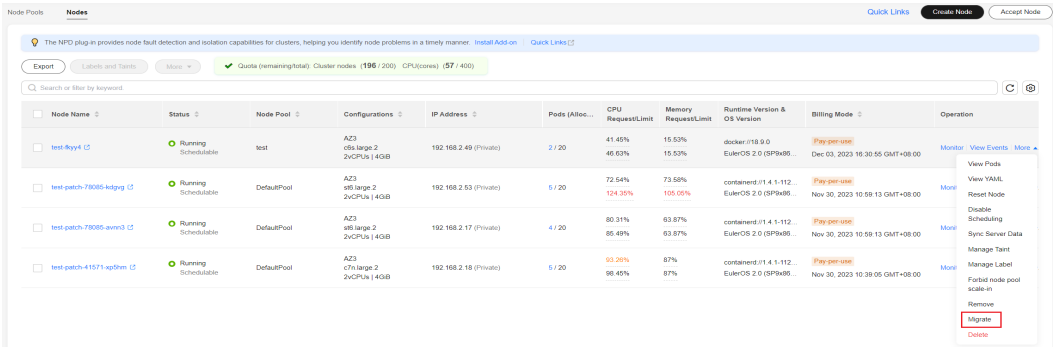
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Nodes** and click the **Node Pools** tab.

Step 3 Click **View Node** in the **Operation** column of the node pool to be migrated.

Step 4 Click **More** > **Migrate** in the **Operation** column of the target node to migrate the node.

Figure 3-68 Migrating a node



Step 5 In the displayed **Migrate Node** dialog box, confirm the information.

NOTE

- The migration does not affect custom resource tags, 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

3.4.3.9 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.

Constraints

- A yearly/monthly-billed node pool cannot be deleted before all nodes in it is removed first.
- 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

3.5 Workloads

3.5.1 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.

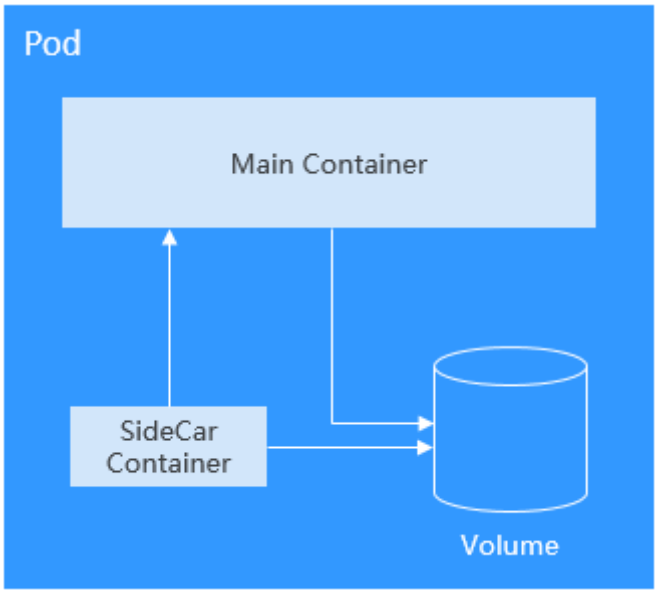
Overview of Pod

A pod is the smallest and 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 3-69**. 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 3-69 Pod running multiple containers

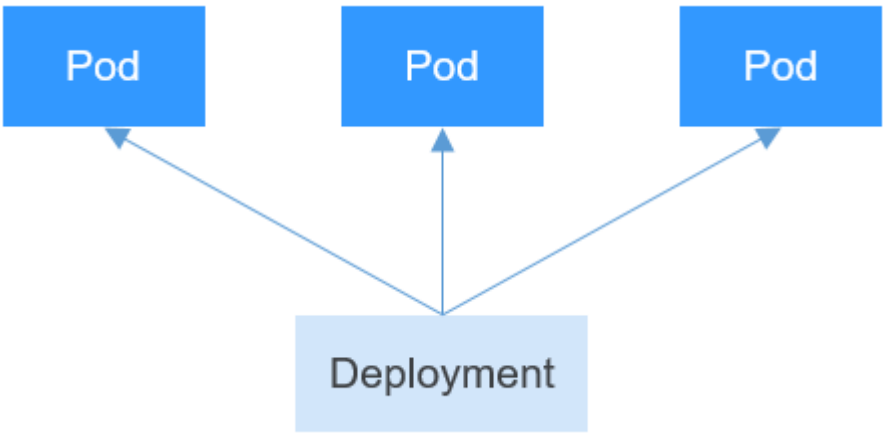


In Kubernetes, pods are rarely created directly. Instead, Kubernetes controller manages pods through pod instances such as Deployments and jobs. A controller typically uses a pod template to create pods. The controller can also manage multiple pods and provide functions such as replica management, rolling 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 3-70 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.

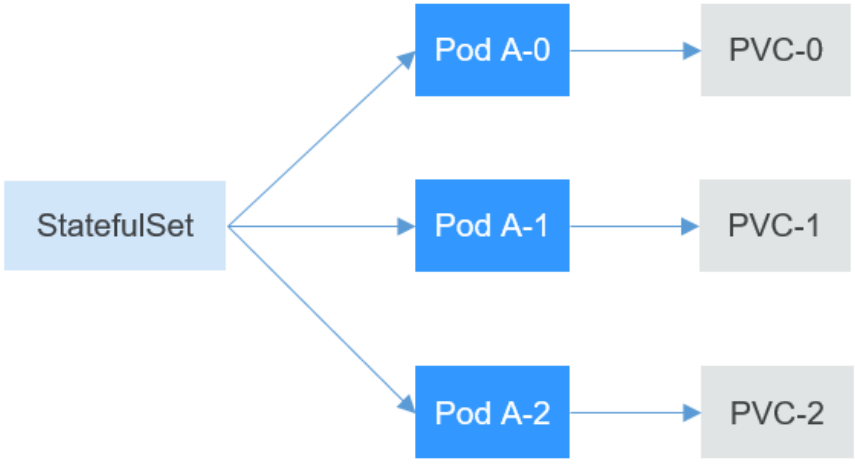
However, Deployments cannot meet the requirements in some distributed scenarios when each pod requires its own status or in a distributed database where each pod requires independent storage.

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 PersistentVolumeClaims (PVCs) with fixed identifiers to ensure that pods can access the same persistent data after being rescheduled.

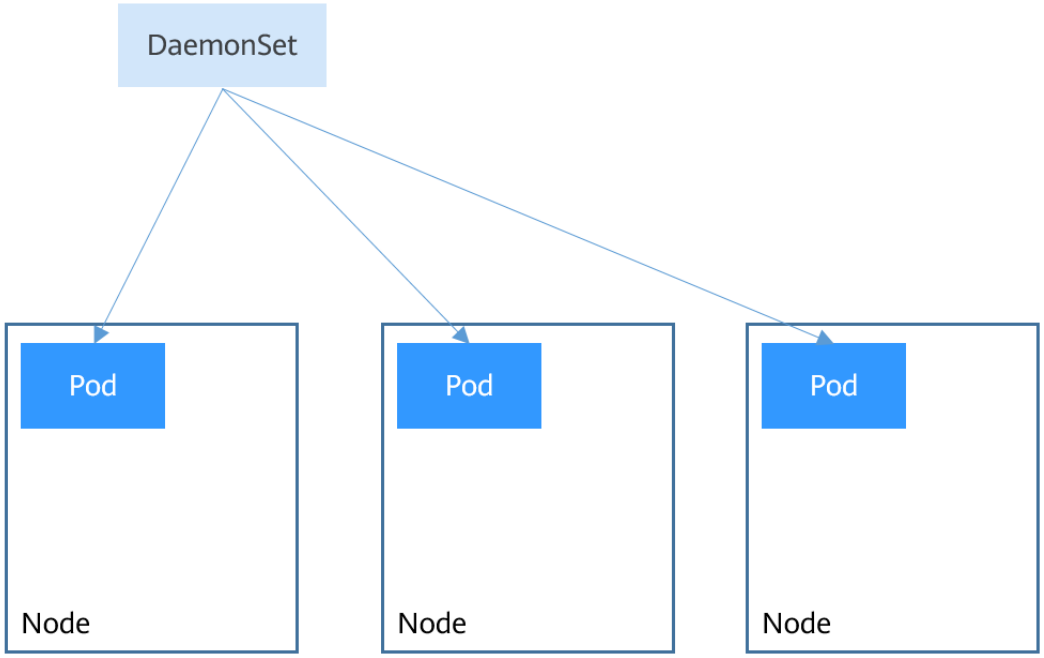


Overview of DaemonSet

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 3-71 DaemonSet



Overview of Job and CronJob

Jobs and cron jobs 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 cron job runs a job periodically on a specified schedule. A cron job 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 3-99 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.

3.5.2 Creating a Workload

3.5.2.1 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 on how to create a cluster, see **3.2.2.1 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 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **Deployment**. For details about workload types, see **3.5.1 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 **3.10.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses runC by default, whereas a CCE Turbo cluster supports both runC and Kata. For details about the differences, see **3.5.3.1 Kata 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 **3.5.3.2 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. To use a third-party image, see **3.5.3.4 Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.

Parameter	Description
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 **3.5.3.5 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 **3.5.3.5 Configuring Container Specifications**.
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on is installed. ▪ **All**: No GPU will be used. ▪ **Dedicated**: GPU resources are dedicated for the container. ▪ **Shared**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container uses 10% of GPU resources. For details about how to use GPUs in the cluster, see **3.6.3.1 Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required Ascend chips. The value must be an integer and the **3.14.12 CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **3.6.4 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.

Parameter	Description
(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 **3.5.3.6 Configuring Container Lifecycle Parameters**.
  - (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **3.5.3.7 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 **3.5.3.8 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 **3.8 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 **3.9.4.2.2 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 3-116**.

- **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. 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 **3.7.3.1 Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **3.5.3.10 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.
    - **Multi-AZ deployment is preferred**: Workload pods are preferentially scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to that AZ but onto different nodes for high availability. If there are fewer nodes than pods, the extra pods will fail to run.
    - **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.
    - **Custom policies**: Affinity and anti-affinity policies can be customized as needed. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.
  - **Node Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Node Affinity**: 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 as needed. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.
- **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 **3.5.3.12 Taints and Tolerations**.
- **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 **3.5.3.13 Labels and Annotations**.

- **DNS**: Configure a separate DNS policy for the workload. For details, see **3.7.5.2 DNS Configuration**.
- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **3.5.3.9 Configuring APM Settings for Performance Bottleneck Analysis**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **3.7.6.2 Configuring QoS for a Pod**.

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 connect to the cluster. For details, see **3.2.3.1 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 3-100**.

Table 3-100 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
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

Parameter	Description	Mandatory/ Optional
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 "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 **3.7 Network**.

----End

Documentation

- **Upgrading Pods Without Interrupting Services**
- **Configuring Domain Name Resolution for CCE Containers**

3.5.2.2 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.

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 **3.7.3.6 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 **3.2.2.1 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 **3.5.1 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 **3.10.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses runC by default, whereas a CCE Turbo cluster supports both runC and Kata. For details about the differences, see **3.5.3.1 Kata 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 **3.5.3.2 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. To use a third-party image, see **3.5.3.4 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 **3.5.3.5 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 **3.5.3.5 Configuring Container Specifications**.

Parameter	Description
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on is installed. ■ **All**: No GPU will be used. ■ **Dedicated**: GPU resources are dedicated for the container. ■ **Shared**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container uses 10% of GPU resources. For details about how to use GPUs in the cluster, see **3.6.3.1 Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required Ascend chips. The value must be an integer and the **3.14.12 CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **3.6.4 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 **3.5.3.6 Configuring Container Lifecycle Parameters**.
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **3.5.3.7 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 **3.5.3.8 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 **3.8 Storage**.

    NOTE

    - StatefulSets support dynamic attachment of EVS disks. For details, see **3.8.3.4 Dynamically Mounting an EVS Disk to a StatefulSet** or **3.8.7.4 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 **3.9.4.2.2 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 3-116**.

- **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. 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 **3.7.3.6 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 **3.7.3.1 Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **3.5.3.10 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.
    - **Multi-AZ deployment is preferred**: Workload pods are preferentially scheduled to nodes in different AZs through pod anti-affinity (**podAntiAffinity**). If all the nodes in the cluster are deployed in the same AZ, the pods will be scheduled to that AZ but onto different nodes for high availability. If there are fewer nodes than pods, the extra pods will fail to run.
    - **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.
    - **Custom policies**: Affinity and anti-affinity policies can be customized as needed. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.
  - **Node Affinity**: Common load affinity policies are offered for quick load affinity deployment.
    - **Node Affinity**: 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 as needed. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.
- **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 **3.5.3.12 Taints and Tolerations**.
- **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 **3.5.3.13 Labels and Annotations**.
- **DNS**: Configure a separate DNS policy for the workload. For details, see **3.7.5.2 DNS Configuration**.

- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **3.5.3.9 Configuring APM Settings for Performance Bottleneck Analysis**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **3.7.6.2 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 **3.7.6.4.3 Configuring a Static IP Address for a Pod**.

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 connect to the cluster. For details, see **3.2.3.1 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 **3.7 Network**.

----End

3.5.2.3 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 **3.2.2.1 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 **3.5.1 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 **3.10.1 Creating a Namespace**.

- **Container Runtime**: A CCE standard cluster uses runC by default, whereas a CCE Turbo cluster supports both runC and Kata. For details about the differences, see **3.5.3.1 Kata 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 **3.5.3.2 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. To use a third-party image, see **3.5.3.4 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 **3.5.3.5 Configuring Container Specifications**.

Parameter	Description
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 **3.5.3.5 Configuring Container Specifications**.
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on is installed. ▪ **All**: No GPU will be used. ▪ **Dedicated**: GPU resources are dedicated for the container. ▪ **Shared**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container uses 10% of GPU resources. For details about how to use GPUs in the cluster, see **3.6.3.1 Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required Ascend chips. The value must be an integer and the **3.14.12 CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **3.6.4 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 **3.5.3.6 Configuring Container Lifecycle Parameters**.
  - (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **3.5.3.7 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 **3.5.3.8 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 **3.8 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 **3.9.4.2.2 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 3-116**.

- **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. 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 **3.7.3.1 Overview**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and parameters of the workload. **Rolling upgrade** and **Replace upgrade** are available. For details, see **3.5.3.10 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.
    - **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 as needed. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

- **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 **3.5.3.12 Taints and Tolerations**.
- **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 **3.5.3.13 Labels and Annotations**.
- **DNS**: Configure a separate DNS policy for the workload. For details, see **3.7.5.2 DNS Configuration**.
- **APM Settings**: Use Application Performance Management (APM) to provide more accurate problem analysis and location for Java programs. For details, see **3.5.3.9 Configuring APM Settings for Performance Bottleneck Analysis**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **3.7.6.2 Configuring QoS for a Pod**.

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 connect to the cluster. For details, see **3.2.3.1 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 documents**.

```
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 **3.7 Network**.

----End

3.5.2.4 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 **3.3.4 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 **3.5.1 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 **3.10.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.
- **Container Runtime**: A CCE standard cluster uses runC by default, whereas a CCE Turbo cluster supports both runC and Kata. For details about the differences, see **3.5.3.1 Kata 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.

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. To use a third-party image, see **3.5.3.4 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 **3.5.3.5 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 **3.5.3.5 Configuring Container Specifications**.
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on is installed. ▪ **All**: No GPU will be used. ▪ **Dedicated**: GPU resources are dedicated for the container. ▪ **Shared**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container uses 10% of GPU resources. For details about how to use GPUs in the cluster, see **3.6.3.1 Default GPU Scheduling in Kubernetes**.

Parameter	Description
(Optional) NPU Quota	Number of required Ascend chips. The value must be an integer and the **3.14.12 CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **3.6.4 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 **3.5.3.6 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 **3.5.3.8 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 **3.8 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 **3.9.4.2.2 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 3-116**.

- **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. 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 **3.5.3.13 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 **3.7.6.2 Configuring QoS for a Pod**.

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, all pods of the job are terminated. The priority of .spec.activeDeadlineSeconds is higher than that of .spec.backoffLimit. That is, if a job reaches the .spec.activeDeadlineSeconds, the spec.backoffLimit is ignored.

Based on the **.spec.completions** and **.spec.parallelism** settings, jobs are classified into the following types.

Table 3-101 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
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
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019893809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151557485724245415069595082953311686172785588907509838175463746493931925506040092770167113900984882401285836160356370766010471018194295559619894676783744944825537977472684710404753464620804668425906949129331367702898915210475216205696602405803815019351125338243003558764024749647326391419927260426992279678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295532116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321721477235014144197356854816136115735255213347574184946843852332390739414333454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438382796797668145410095388378636095068006422512520511739298489608412848862694560424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759 01
```

----End

Related Operations

After a one-off job is created, you can perform operations listed in **Table 3-102**.

Table 3-102 Other operations

Operation	Description
Editing a YAML file	Click **More** > **Edit YAML** next to the job name to edit the YAML file corresponding to the current 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.

3.5.2.5 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 **3.3.4 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 **3.5.1 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 **3.10.1 Creating a Namespace**.
- **Container Runtime**: A CCE standard cluster uses runC by default, whereas a CCE Turbo cluster supports both runC and Kata. For details about the differences, see **3.5.3.1 Kata 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.
Image Name	Click **Select Image** and select the image used by the container. To use a third-party image, see **3.5.3.4 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 **3.5.3.5 Configuring Container Specifications**.

Parameter	Description
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 **3.5.3.5 Configuring Container Specifications**.
(Optional) GPU Quota	Configurable only when the cluster contains GPU nodes and the **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on is installed. ▪ **All**: No GPU will be used. ▪ **Dedicated**: GPU resources are dedicated for the container. ▪ **Shared**: percentage of GPU resources used by the container. For example, if this parameter is set to **10%**, the container uses 10% of GPU resources. For details about how to use GPUs in the cluster, see **3.6.3.1 Default GPU Scheduling in Kubernetes**.
(Optional) NPU Quota	Number of required Ascend chips. The value must be an integer and the **3.14.12 CCE AI Suite (Ascend NPU)** add-on must be installed. For details about how to use NPUs in the cluster, see **3.6.4 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 **3.5.3.6 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 **3.5.3.8 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. 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 is executed by hour as an example. As **/2, /3, /4, /6, /8, and /12** can exactly divide 24 hours, an accurate period can be represented. If another period is used, the last period will be reset at the beginning of a new day. For example, if the cron expression is *** */12 * * ***, the execution time is **00:00:00** and **12:00:00** every day. If the cron expression is *** */13 * * ***, the execution time is **00:00:00** and **13:00:00** every day. At 00:00 on the next day, the execution time is updated even if the period does not reach 13 hours.

- **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 **3.5.3.13 Labels and Annotations**.
- **Network Configuration**
  - Pod ingress/egress bandwidth limitation: You can set ingress/egress bandwidth limitation for pods. For details, see **3.7.6.2 Configuring QoS for a Pod**.

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 Cron job. 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 cron job.

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 3-103**.

Table 3-103 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. Therefore, exercise caution when deleting a job.

3.5.3 Configuring a Container

3.5.3.1 Kata Runtime and Common Runtime

The most significant difference is that each Kata container (pod) runs on an independent micro-VM, has an independent OS kernel, and is securely isolated at the virtualization layer. With Kata 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 common runtime or Kata runtime as required. The differences between them are as follows.

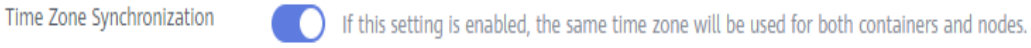
Category	Kata Runtime	Common Runtime
Node type used to run containers	ECS (PM)	ECS (VM) ECS (PM)
Container engine	containerd	Docker and containerd

Category	Kata Runtime	Common Runtime
Container runtime	Kata	runC
Container kernel	Exclusive kernel	Sharing the kernel with the host
Container isolation	Lightweight VMs	cgroups and namespaces
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-core CPUs and 356 MiB of memory.	None
Minimal specifications	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.
3.7.6.1 Host Network	Not supported	Supported

For details about how to use containerd and Docker commands, see **3.3.2 Container Engine**.

3.5.3.2 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
```

3.5.3.3 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.

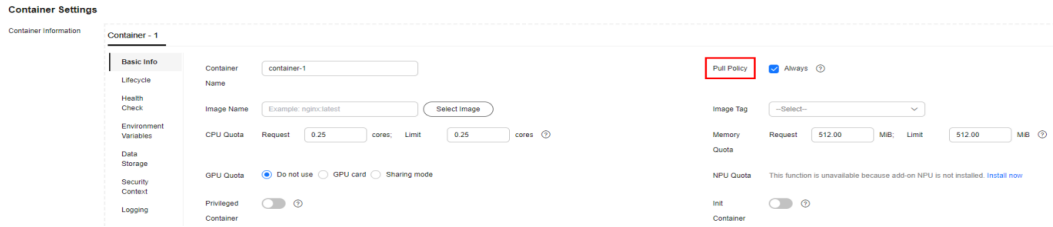
```
apiVersion: v1
kind: Pod
```



```
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 3-72 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.

3.5.3.4 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.

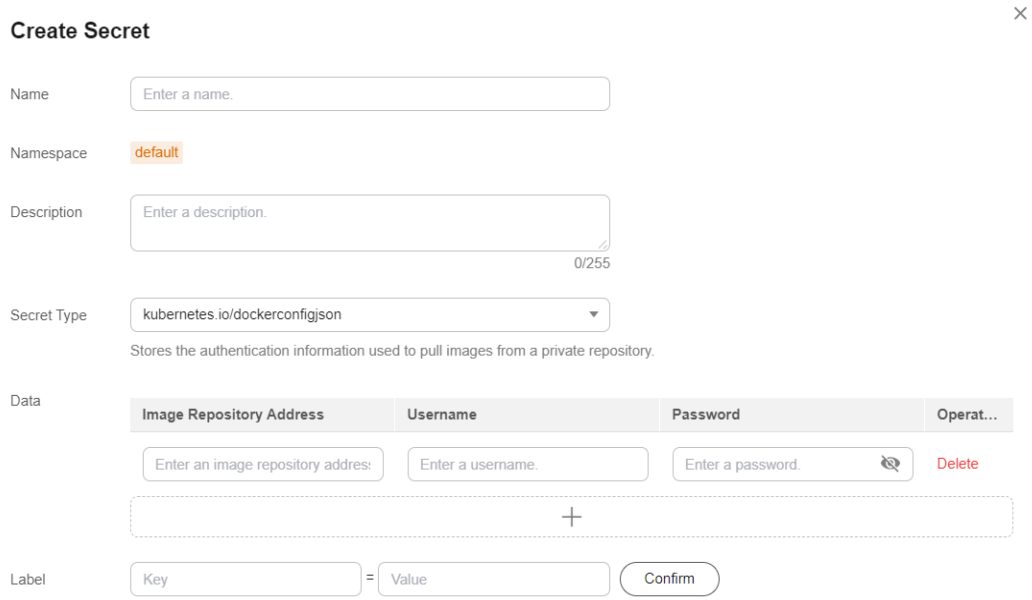
Using the Console

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 **3.11.3 Creating a Secret**.

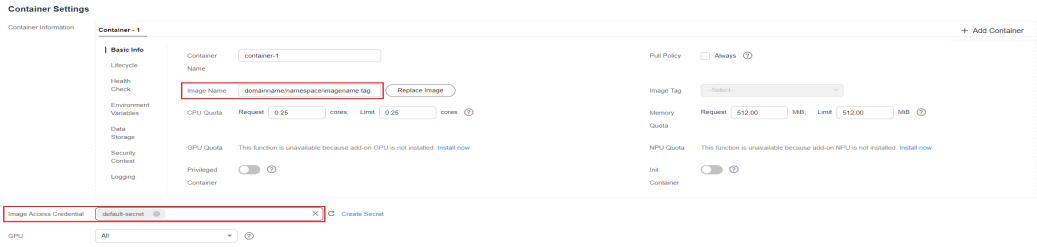
Enter the username and password used to access the third-party image repository.

Figure 3-73 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 3-74 Private image path



Step 3 Set other parameters and click **Create Workload**.

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

3.5.3.5 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 3-104 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 3-105 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
```


3.5.3.6 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 during image start. To run a specific command or rewrite the default image value, you must perform specific settings:

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 3-106 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 3-107 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.
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 3-108 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.

Parameter	Description
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 3-109 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 the **uninstall.sh** script 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
```

3.5.3.7 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.

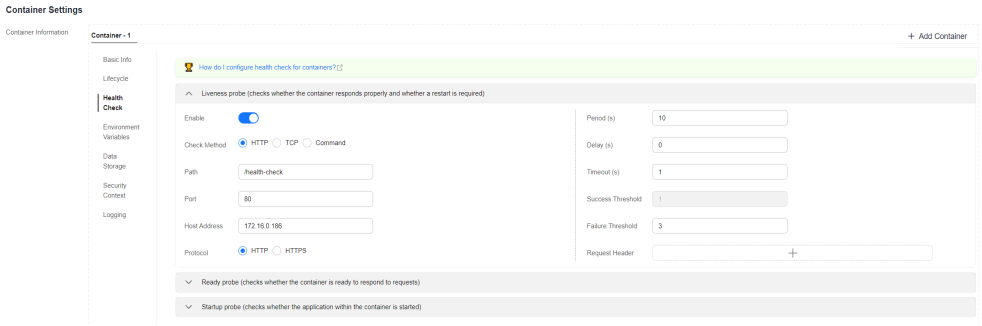
Check Method

- **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 3-75 HTTP request-based check

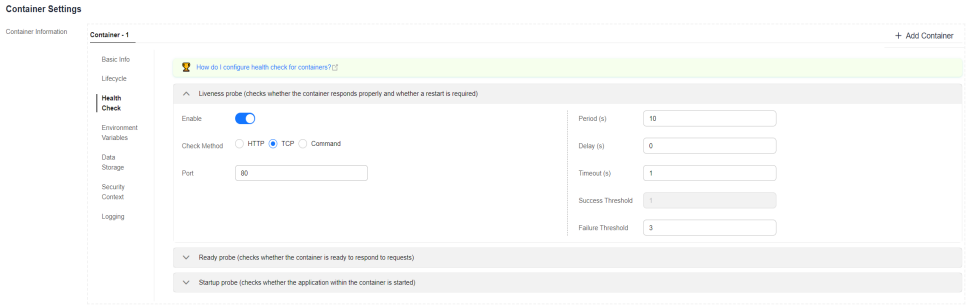
  

- **TCP port**

  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 3-76 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.

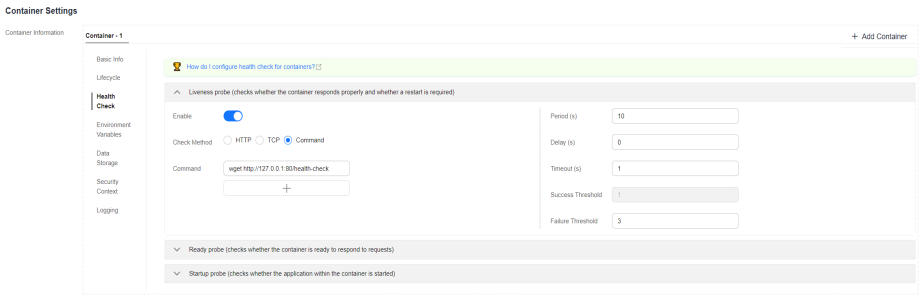
  The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

  - 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 3-77 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**

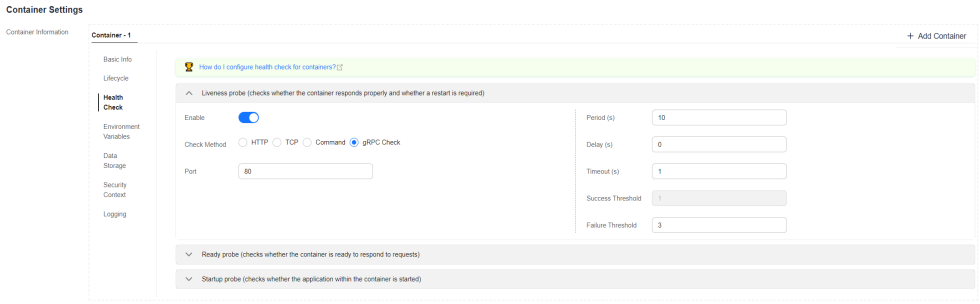
  gRPC checks can configure startup, liveness, and readiness probes for your gRPC application without exposing any HTTP endpoint, nor do you need an

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 3-78 gRPC check



Common Parameters

Table 3-110 Common parameter description

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.

Parameter	Description
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
```


3.5.3.8 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 key**: Import all keys in a ConfigMap as environment variables.
- **Added from ConfigMap**: Import a key in a ConfigMap as the value of an environment variable. As shown in **Figure 3-79**, 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 keys 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 3-79**, 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 3-79**, 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 3-79**, 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

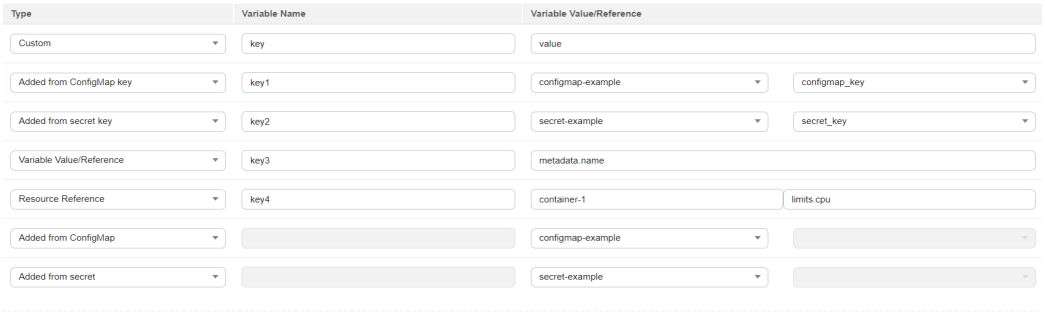
Step 1 Log in to the CCE console.

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.

Figure 3-79 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
ey1=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.
```


3.5.3.9 Configuring APM Settings for Performance Bottleneck Analysis

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 2.0** 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

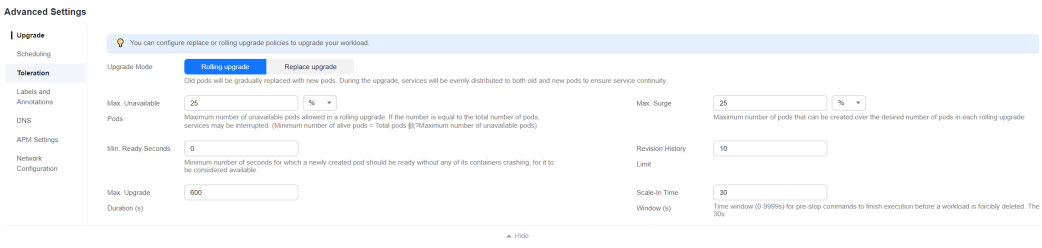
3.5.3.10 Workload Upgrade Policies

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 3-80 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
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

Parameter	Description	Constraint
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 the earlier version when a fault occurs during the 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**.

3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)

Kubernetes supports node affinity and pod affinity/anti-affinity. You can configure custom rules to achieve affinity and anti-affinity scheduling. For example, you can deploy frontend pods and backend pods together, deploy the same type of applications on a specific node, or deploy different applications on different nodes.

Kubernetes affinity applies to nodes and pods.

- **nodeAffinity**: similar to pod nodeSelector, and they both schedule pods only to the nodes with specified labels. The difference between nodeAffinity and nodeSelector lies in that nodeAffinity features stronger expression than nodeSelector and allows you to specify preferentially selected soft constraints. The two types of node affinity are as follows:
  - requiredDuringSchedulingIgnoredDuringExecution: hard constraint that **must be met**. The scheduler can perform scheduling only when the rule is met. This function is similar to nodeSelector, but it features stronger syntax expression. For details, see **Node Affinity (nodeAffinity)**.
  - preferredDuringSchedulingIgnoredDuringExecution: soft constraint that is **met as much as possible**. The scheduler attempts to find the node that meets the rule. If no matching node is found, the scheduler still schedules the pod. For details, see **Node Preference Rules**.
- **Workload Affinity (podAffinity)**/**Workload Anti-affinity (podAntiAffinity)**: The nodes to which a pod can be scheduled are determined based on the label of the pod running on a node, but not the label of the node. Similar to node affinity, workload affinity and anti-affinity are also of requiredDuringSchedulingIgnoredDuringExecution and preferredDuringSchedulingIgnoredDuringExecution types.

> **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.

You can create the preceding affinity policies on the console. For details, see **Configuring Load Affinity on the Console** or **Configuring Node Affinity on the Console**.

Configuring Load Affinity on the Console

Step 1 When creating a workload, click **Scheduling** in the **Advanced Settings** area. For details about how to create a workload, see **3.5.2 Creating a Workload**.

Step 2 Select a load affinity scheduling policy.

- **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 different AZs and different nodes through pod anti-affinity. 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 3-111**.

  Select a proper policy type and click + to add a policy. For details about the parameters, see **Table 3-112**.

Table 3-111 Load affinity policies

Policy	Type	Description
Workload Affinity	Required	Hard constraint, which is used to configure the conditions that must be met and corresponds to the **requiredDuringSchedulingIgnoredDuringExecution** field in YAML. Select pods that require affinity by label. If such pods have been running on a node in the topology domain, the scheduler will **forcibly** schedule the created pods to that topology domain. **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 domain for scheduling.

Policy	Type	Description
	Preferred	Soft constraint, which is used to configure the conditions that preferentially to be met and corresponds to the **preferredDuringSchedulingIgnoredDuringExecution** field in YAML. Select pods that require affinity by label. If such pods have been running on a node in the topology domain, the scheduler will **preferentially** schedule the created pods to that topology domain. **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 domain 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 have been running on a node in the topology domain, the scheduler will **not** schedule the created pods to that topology domain. **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 domains where the pods that meet the label filtering conditions locate will not be scheduled.
	Preferred	Soft constraint, which corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in YAML for specifying the conditions that are preferentially met. Select one or more pods that require anti-affinity by label. If such pods have been running on a node in the topology domain, the scheduler will **preferentially** schedule the created pods to other topology domains. **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 domains involve the pods that require anti-affinity, a topology domain will be selected for scheduling.

Table 3-112 Parameters for configuring load affinity/anti-affinity scheduling policies

Parameter	Description
Weight	This parameter is available only in a **Preferred** scheduling policy. The weight ranges from 1 to 100. During scheduling, the scheduler adds the weight to the scores of other priority functions and schedules pods to the node with the largest total score.
Namespace	Namespace for which the scheduling policy takes effect.
Topology Key	A topology domain (**topologyKey**) determines the range of nodes to be scheduled based on node labels. For example, if the node label is **kubernetes.io/hostname**, the range of nodes is determined by node name. Nodes with different names are in different topology domains. In this case, a topology domain contains only one node. If the specified label is **kubernetes.io/os**, the range of nodes is determined by node OS. Nodes running different OSs belong to different topology domains. In this case, a topology domain may contain multiple nodes. After the node range is determined using the topology domain, configure the policy for scheduling, including the label name, operator, and label value. The minimum unit for scheduling is a topology domain. For example, if a node in a topology domain meets the load affinity policy, all nodes in the topology domain can be scheduled.
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.
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 name. – **DoesNotExist**: The affinity or anti-affinity object does not have the specified label name.
Label Value	When configuring a workload affinity or anti-affinity policy, enter the value of the workload label.

Step 3 After the scheduling policy is added, click **Create Workload**.

----End

Configuring Node Affinity on the Console

Step 1 When creating a workload, click **Scheduling** in the **Advanced Settings** area. For details about how to create a workload, see **3.5.2 Creating a Workload**.

Step 2 Select a node affinity scheduling policy.

- **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 based on 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 based on the default cluster scheduling policy.
- **Custom policies**: allow flexible scheduling of workload pods based on node labels. For details about the supported scheduling policies, see **Table 3-113**.

  Select a proper policy type and click + to add a policy. For details about the parameters, see **Table 3-114**. 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 3-113 Node affinity settings

Parameter	Description
Required	Hard constraint, which corresponds to **requiredDuringSchedulingIgnoredDuringExecution** for specifying the conditions that must be met. If multiple rules **that must be met** are added, scheduling will be performed when only one rule is met.
Preferred	Soft constraint, which corresponds to **preferredDuringSchedulingIgnoredDuringExecution** for specifying the conditions that are preferentially met. If multiple rules **that are preferentially met** are added, scheduling will be performed even if one or none of the rules is met.

Table 3-114 Parameters for configuring node affinity scheduling policies

Parameter	Description
Label	When configuring node affinity, enter the node label to be matched. Both default labels and custom labels are supported.

Parameter	Description
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 name. – **DoesNotExist**: The affinity or anti-affinity object does not have the specified label name. – **Gt**: (available only for node affinity) The label value of the scheduled node is greater than the list value (string comparison). – **Lt**: (available only for node affinity) The label value of the scheduled node is less than the list value (string comparison).
Label Value	When configuring node affinity, enter the value of the node label.

Step 3 After the scheduling policy is added, click **Create Workload**.

----End

Node Affinity (nodeAffinity)

Workload node affinity rules are implemented using node labels. When a node is created in a CCE cluster, certain labels are automatically added. You can run the **kubectl describe node** command to view the labels. The following is an example:

```
$ kubectl describe node 192.168.0.212
Name:               192.168.0.212
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    failure-domain.beta.kubernetes.io/is-baremetal=false
                    failure-domain.beta.kubernetes.io/region=******
                    failure-domain.beta.kubernetes.io/zone=******
                    kubernetes.io/arch=amd64
                    kubernetes.io/availablezone=******
                    kubernetes.io/eniquota=12
                    kubernetes.io/hostname=192.168.0.212
                    kubernetes.io/os=linux
                    node.kubernetes.io/subnetid=fd43acad-33e7-48b2-a85a-24833f362e0e
                    os.architecture=amd64
                    os.name=EulerOS_2.0_SP5
                    os.version=3.10.0-862.14.1.5.h328.eulerosv2r7.x86_64
```

In workload scheduling, common node labels are as follows:

- **failure-domain.beta.kubernetes.io/region**: region where the node is located.
- **failure-domain.beta.kubernetes.io/zone**: availability zone to which the node belongs.
- **kubernetes.io/hostname**: host name of the node.

Kubernetes provides the **nodeSelector** field. When creating a workload, you can set this field to specify that the pod can be deployed only on a node with the specific label. The following example shows how to use a nodeSelector to deploy the pod only on the node with the **gpu=true** label.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:              # Node selection. A pod is created on a node only when the node meets gpu=true.
    gpu: true
...
```

Node affinity rules can achieve the same results. Compared with nodeSelector, node affinity rules seem more complex, but with a more expressive syntax. You can use the **spec.affinity.nodeAffinity** field to set node affinity. There are two types of node affinity:

- **requiredDuringSchedulingIgnoredDuringExecution**: Kubernetes cannot schedule the pod unless the rule is met.
- **PreferredDuringSchedulingIgnoredDuringExecution**: Kubernetes tries to find a node that meets the rule. If a matching node is not available, Kubernetes still schedules the pod.

NOTE

In these two types of node affinity, **requiredDuringScheduling** or **preferredDuringScheduling** indicates that the pod can be scheduled to a node only when all the defined rules are met (required). **IgnoredDuringExecution** indicates that if the node label changes after Kubernetes schedules the pod, the pod continues to run and will not 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.

The following is an example of setting 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:
```



```
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

In this example, the scheduled node must contain a label with the key named **gpu**. The value of **operator** is to **In**, indicating that the label value must be in the **values** list. That is, the key value of the **gpu** label of the node is **true**. For details about other values of **operator**, see **Operator Values**. Note that there is no such thing as nodeAntiAffinity because operators **NotIn** and **DoesNotExist** provide the same function.

The following describes how to check whether the rule takes effect. Assume that a cluster has three nodes.

```
$ kubectl get node
NAME           STATUS   ROLES    AGE   VERSION
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Add the **gpu=true** label to the **192.168.0.212** node.

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME           STATUS   ROLES    AGE   VERSION                         GPU
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

Create the Deployment. You can find that all pods are deployed on the **192.168.0.212** node.

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE
gpu-6df65c44cf-42xw4     1/1     Running   0          15s   172.16.0.37   192.168.0.212
gpu-6df65c44cf-jzjvs     1/1     Running   0          15s   172.16.0.36   192.168.0.212
gpu-6df65c44cf-zv5cl     1/1     Running   0          15s   172.16.0.38   192.168.0.212
```

Node Preference Rules

The preceding **requiredDuringSchedulingIgnoredDuringExecution** rule is a hard selection rule. There is another type of selection rule, that is, **preferredDuringSchedulingIgnoredDuringExecution**. It is used to specify which nodes are preferred during scheduling.

To achieve this effect, add a node attached with SAS disks to the cluster, add the **DISK=SAS** label to the node, and add the **DISK=SSD** label to the other three nodes.

```
$ kubectl get node -L DISK,gpu
NAME           STATUS   ROLES    AGE     VERSION                         DISK     GPU
192.168.0.100   Ready    <none>   7h23m   v1.15.6-r1-20.3.0.2.B001-15.30.2   SAS
192.168.0.212   Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD      true
192.168.0.94    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
192.168.0.97    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
```


Define a Deployment. Use the **preferredDuringSchedulingIgnoredDuringExecution** rule to set the weight of nodes with the SSD disk installed as **80** and nodes with the **gpu=true** label as **20**. In this way, pods are preferentially deployed on the nodes with the SSD disk installed.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  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:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: DISK
                operator: In
                values:
                - SSD
          - weight: 20
            preference:
              matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

After the deployment, there are five pods deployed on the node **192.168.0.212** (label: **DISK=SSD** and **GPU=true**), three pods deployed on the node **192.168.0.97** (label: **DISK=SSD**), and two pods deployed on the node **192.168.0.100** (label: **DISK=SAS**).

From the preceding output, you can find that no pods of the Deployment are scheduled to node **192.168.0.94** (label: **DISK=SSD**). This is because the node already has many pods on it and its resource usage is high. This also indicates that the **preferredDuringSchedulingIgnoredDuringExecution** rule defines a preference rather than a hard requirement.

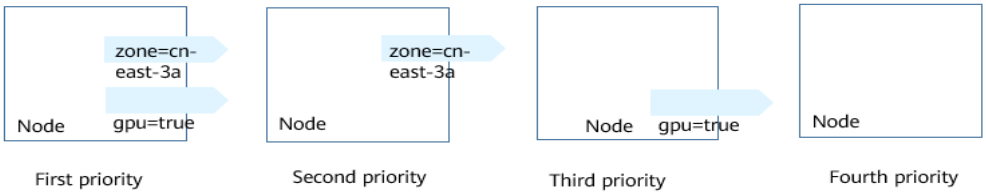
```
$ kubectl create -f affinity2.yaml
deployment.apps/gpu created
```



```
$ kubectl get po -o wide
NAME                   READY   STATUS    RESTARTS   AGE     IP           NODE
gpu-585455d466-5bmcz   1/1     Running   0          2m29s   172.16.0.44   192.168.0.212
gpu-585455d466-cg2l6   1/1     Running   0          2m29s   172.16.0.63   192.168.0.97
gpu-585455d466-f2bt2   1/1     Running   0          2m29s   172.16.0.79   192.168.0.100
gpu-585455d466-hdb5n   1/1     Running   0          2m29s   172.16.0.42   192.168.0.212
gpu-585455d466-hkgvz   1/1     Running   0          2m29s   172.16.0.43   192.168.0.212
gpu-585455d466-mngvn   1/1     Running   0          2m29s   172.16.0.48   192.168.0.97
gpu-585455d466-s26qs   1/1     Running   0          2m29s   172.16.0.62   192.168.0.97
gpu-585455d466-sxtzm   1/1     Running   0          2m29s   172.16.0.45   192.168.0.212
gpu-585455d466-t56cm   1/1     Running   0          2m29s   172.16.0.64   192.168.0.100
gpu-585455d466-t5w5x   1/1     Running   0          2m29s   172.16.0.41   192.168.0.212
```

In the preceding example, the node scheduling priority is as follows. Nodes with both **SSD** and **gpu=true** labels have the highest priority. Nodes with the **SSD** label but no **gpu=true** label have the second priority (weight: 80). Nodes with the **gpu=true** label but no **SSD** label have the third priority. Nodes without any of these two labels have the lowest priority.

Figure 3-81 Scheduling priority



Workload Affinity (podAffinity)

Node affinity rules affect only the affinity between pods and nodes. Kubernetes also supports configuring inter-pod affinity rules. For example, the frontend and backend of an application can be deployed together on one node to reduce access latency. There are also two types of inter-pod affinity rules: **requiredDuringSchedulingIgnoredDuringExecution** and **preferredDuringSchedulingIgnoredDuringExecution**.

NOTE

For workload affinity, topologyKey cannot be left blank when requiredDuringSchedulingIgnoredDuringExecution and preferredDuringSchedulingIgnoredDuringExecution are used.

Assume that the backend of an application has been created and has the **app=backend** label.

```
$ kubectl get po -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE
backend-658f6cb858-dlrz8   1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

You can configure the following pod affinity rule to deploy the frontend pods of the application to the same node as the backend pods.

```
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:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - backend
```

Deploy the frontend and you can find that the frontend is deployed on the same node as the backend.

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67   192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70   192.168.0.100
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71   192.168.0.100
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72   192.168.0.100
```

The **topologyKey** field is used to divide topology domains to specify the selection range. If the label keys and values of nodes are the same, the nodes are considered to be in the same topology domain. Then, the contents defined in the following rules are selected. The effect of **topologyKey** is not fully demonstrated in the preceding example because all the nodes have the **kubernetes.io/hostname** label, that is, all the nodes are within the range.

To see how **topologyKey** works, assume that the backend of the application has two pods, which are running on different nodes.

```
$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-5bpd6    1/1     Running   0          23m     172.16.0.40   192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

Add the **prefer=true** label to nodes **192.168.0.97** and **192.168.0.94**.

```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled

$ kubectl get node -L prefer
```

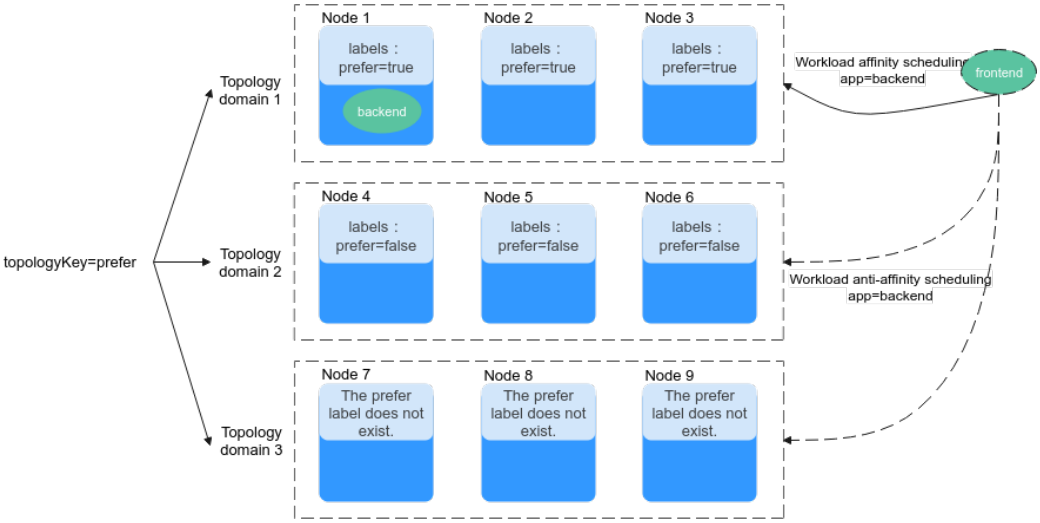


```
NAME            STATUS   ROLES    AGE   VERSION                          PREFER
192.168.0.100   Ready    <none>   44m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212   Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.97    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
```

If the **topologyKey** of **podAffinity** is set to **prefer**, the node topology domains are divided as shown in **Figure 3-82**.

```
    affinity:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: prefer
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - backend
```

Figure 3-82 Topology domains



During scheduling, node topology domains are divided based on the **prefer** label. In this example, **192.168.0.97** and **192.168.0.94** are divided into the same topology domain. If a pod with the **app=backend** label runs in the topology domain, even if not all nodes in the topology domain run the pod with the **app=backend** label (in this example, only the **192.168.0.97** node has such a pod), **frontend** is also deployed in this topology domain (**192.168.0.97** or **192.168.0.94**).

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP           NODE
backend-658f6cb858-5bpd6    1/1     Running   0          26m     172.16.0.40  192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67  192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70  192.168.0.97
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71  192.168.0.97
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72  192.168.0.97
```


Workload Anti-Affinity (podAntiAffinity)

Unlike the scenarios in which pods are preferred to be scheduled onto the same node, sometimes, it could be the exact opposite. For example, if certain pods are deployed together, they will affect the performance.

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.

The following is an example of defining an anti-affinity rule. This rule divides node topology domains by the **kubernetes.io/hostname** label. If a pod with the **app=frontend** label already exists on a node in the topology domain, pods with the same label cannot be scheduled to other nodes in the topology domain.

```
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:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname   # Topology domain of the node
            labelSelector:    # Pod label matching rule
              matchExpressions:
              - key: app
                operator: In
                values:
                - frontend
```

Create an anti-affinity rule and view the deployment result. In the example, node topology domains are divided by the **kubernetes.io/hostname** label. The label values of nodes with the **kubernetes.io/hostname** label are different, so there is only one node in a topology domain. If a **frontend** pod already exists in a topology domain, pods with the same label will not be scheduled to the topology domain. In this example, there are only four nodes. Therefore, there is one pod which is in the **Pending** state and cannot be scheduled.


```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE
frontend-6f686d8d87-8dlsc   1/1     Running   0          18s   172.16.0.76   192.168.0.100
frontend-6f686d8d87-d6l8p   0/1     Pending   0          18s   <none>       <none>
frontend-6f686d8d87-hgcq2   1/1     Running   0          18s   172.16.0.54   192.168.0.97
frontend-6f686d8d87-q7cfq   1/1     Running   0          18s   172.16.0.47   192.168.0.212
frontend-6f686d8d87-xl8hx   1/1     Running   0          18s   172.16.0.23   192.168.0.94
```

Operator Values

You can use the **operator** field to set the logical relationship of the usage rule. The value of **operator** can be:

- **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 name.
- **DoesNotExist**: The affinity or anti-affinity object does not have the specified label name.
- **Gt**: (available only for node affinity) The label value of the scheduled node is greater than the list value (string comparison).
- **Lt**: (available only for node affinity) The label value of the scheduled node is less than the list value (string comparison).

3.5.3.12 Taints and Tolerations

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.

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.

Taint Policy	No Taint Toleration Configured	Taint Toleration Configured
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 3-115 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**.
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
```

3.5.3.13 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 3-116 Pod annotations

Annotation	Description	Default Value

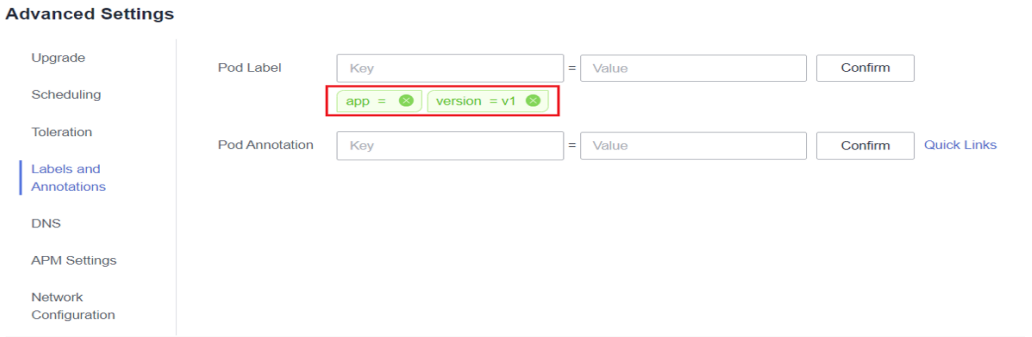
kubernetes.AOM.log.stdout	Standard output parameter. If not specified, the standard log output of all containers is reported to AOM. You can collect stdout logs from certain containers or ignore them at all.
	Example:
	• Collecting none of the stdout logs:
	`kubernetes.AOM.log.stdout: '[]'`
	• Collecting 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 **3.9.6.2 Monitoring Custom Metrics on AOM**.	None

Annotation	Description	Default Value
prometheus.io/scrape	Parameter for reporting Prometheus metrics. If the value is **true**, the current workload reports the monitoring metrics. For details, see **3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	None
prometheus.io/path	URL for Prometheus to collect data. For details, see **3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	/metrics
prometheus.io/port	Endpoint port number for Prometheus to collect data. For details, see **3.9.6.1 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 **3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.	None
kubernetes.io/ingress-bandwidth	Ingress bandwidth of a pod. For details, see **3.7.6.2 Configuring QoS for a Pod**.	None
kubernetes.io/egress-bandwidth	Egress bandwidth of a pod. For details, see **3.7.6.2 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.



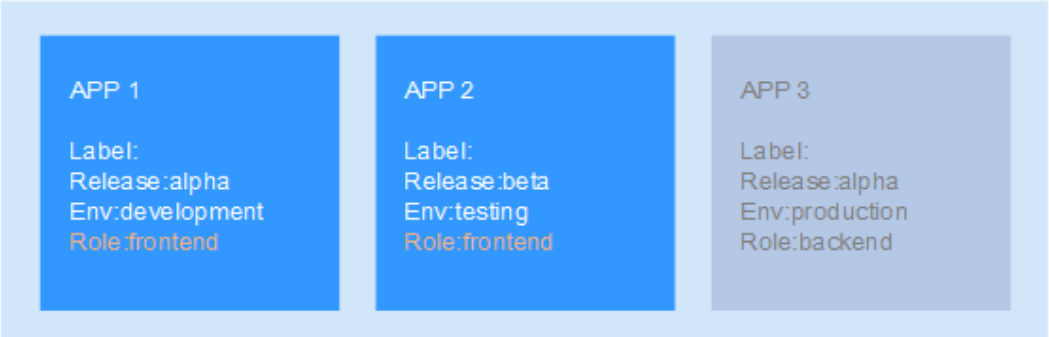
Example YAML:

```
...
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 3-83 Label example



For example, if **key/value** is set to **role/backend**, APP 3 will be selected for affinity scheduling. For details, see **Workload Affinity (podAffinity)**.

3.5.4 Accessing a Container

Scenario

If you encounter unexpected problems when using a container, you can log in to the container to debug it.

Logging In to a Container Using CloudShell

NOTICE

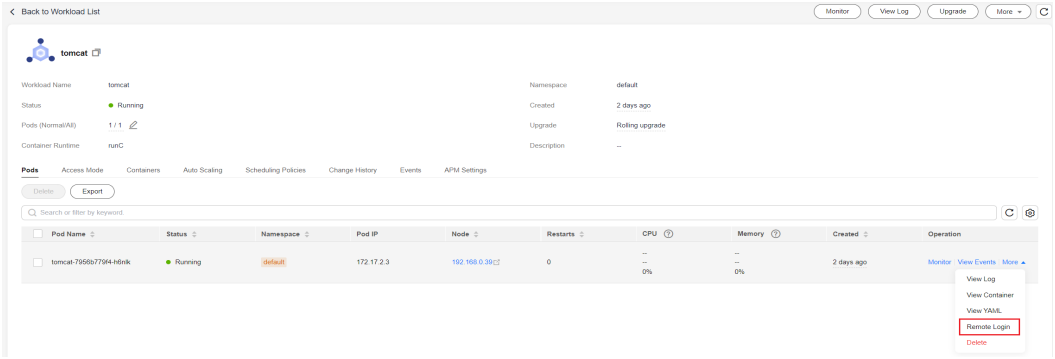
- CloudShell is implemented based on VPC Endpoint (VPCEP). To use kubectl to access a cluster, configure the security group (*Cluster name*-cce-control-*Random number*) on the master node of the cluster to allow access to port 5443. By default, port 5443 allows access from all CIDR blocks. If you have hardened security groups and any cluster cannot be accessed in CloudShell, check whether port 5443 allows access from 198.19.0.0/16.
- Currently, you can use CloudShell to log in to containers only in CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and CN North-Ulanqab1 regions.

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.

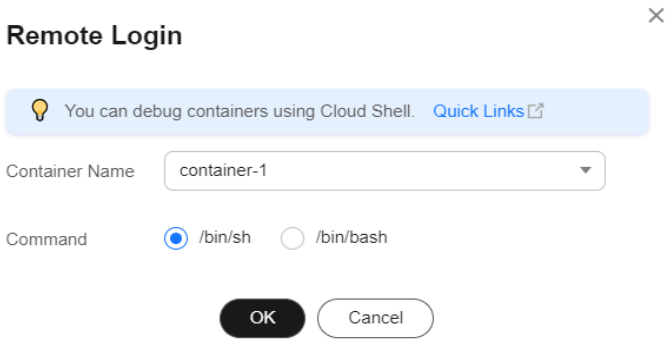
Step 3 Locate the target pod and choose **More** > **Remote Login** in the **Operation** column.

Figure 3-84 Accessing a container



Step 4 In the displayed dialog box, select the container you want to access and the command, and click **OK**.

Figure 3-85 Selecting a container and login command

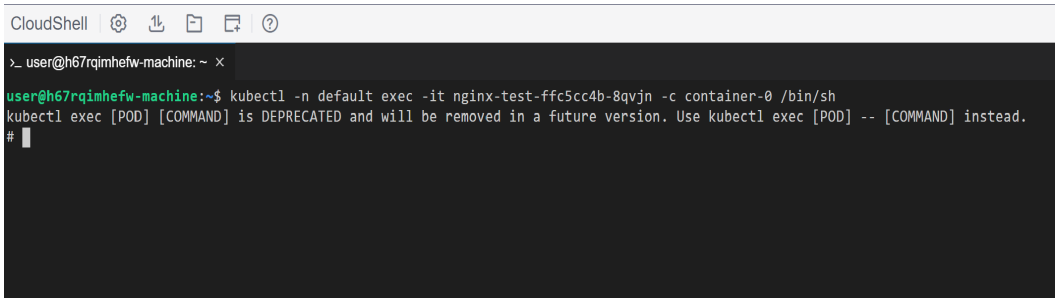


Step 5 You will be automatically redirected to CloudShell. Then, the system initializes kubectl and runs the **kubectl exec** command to log in to the container.

NOTE

Wait for 5 to 10 seconds until the **kubectl exec** command is automatically executed.

Figure 3-86 CloudShell page



----End

Logging In to a Container Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

3.5.5 Managing Workloads and Jobs

Scenario

After a workload is created, you can upgrade, monitor, roll back, or delete the workload, as well as edit its YAML file.

Table 3-117 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 containers 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 pod continues 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.

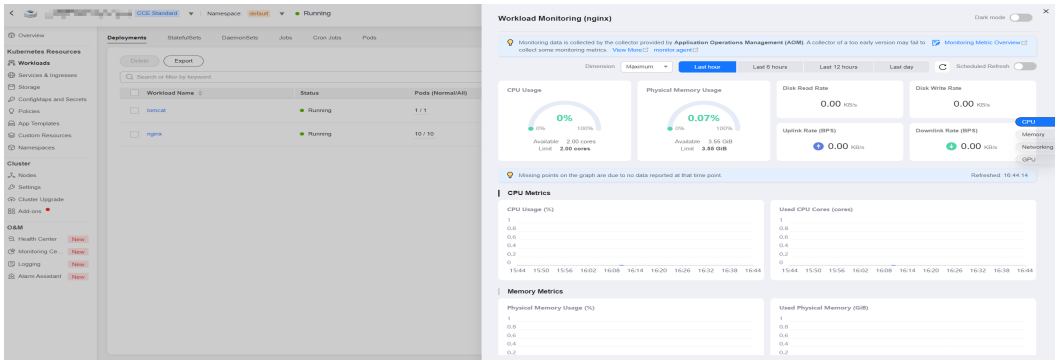
Monitoring a Workload

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 3-87 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

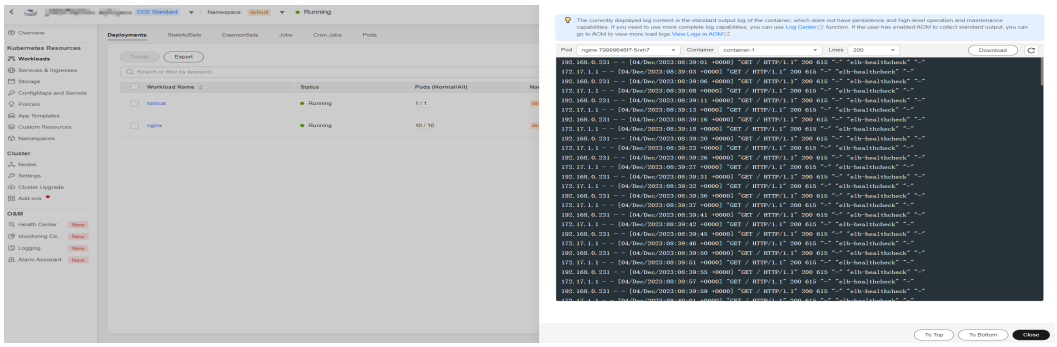
Before viewing logs, ensure that the time of the browser is the same as that on the backend server.

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 the **View Log** of the target workload.

In the displayed **View Log** window, you can view logs.

Figure 3-88 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 **3.9.4.2.2 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 containers 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)

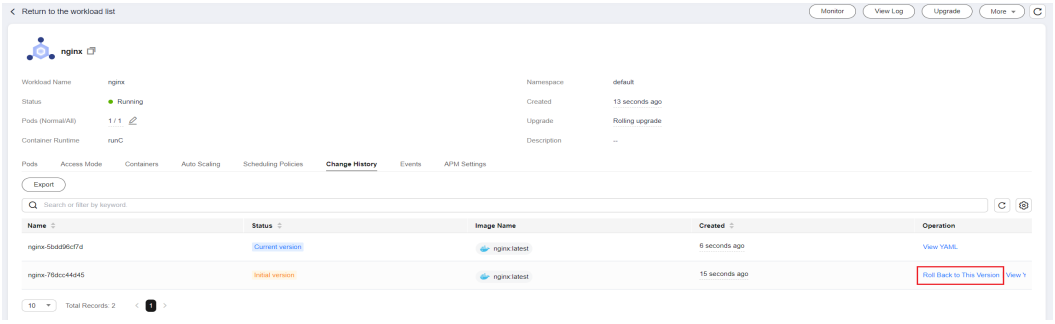
CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

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, 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 3-89 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.
- If a Deployment is being upgraded, it can be upgraded or rolled back. Its pods will inherit the latest updates of the Deployment. If they are inconsistent, the pods are 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

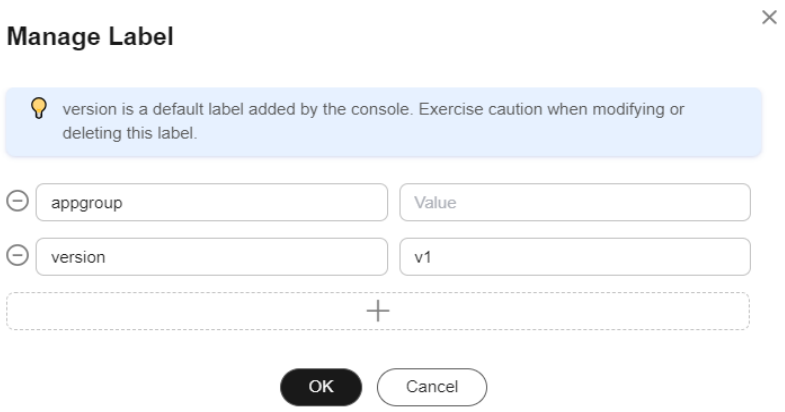
Labels are key-value pairs and can be attached to workloads. You can manage and select workloads by labels. You can add labels to multiple workloads or a specified workload.

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 **Add**, enter a key and a value, and click **OK**.

Figure 3-90 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. Exercise caution when you perform this operation. 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 Deployments as an example to illustrate 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

3.5.6 Pod Security

3.5.6.1 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.
- PodSecurityPolicy was deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25. You can use pod security admission as a substitute for PodSecurityPolicy. For details, see **3.5.6.2 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 3-118**.

Table 3-118 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.
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 defaultAllowPrivilegeEscalation	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.

Item	Description
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 Pod Security Policy

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 3-118**.

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

3.5.6.2 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 3-119 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.

Level	Description
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 3-120**). 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 3-120 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 3-120**.
  - <LEVEL>: must be **privileged**, **baseline**, or **restricted**. For details about the levels, see **Table 3-119**.
- 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 3-120**.
  - <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**

3.6 Scheduling

3.6.1 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 thus 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.	3.6.2.1 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.	3.6.2.2 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.	3.6.3.1 Default GPU Scheduling in Kubernetes
GPU virtualization	GPU virtualization dynamically divides the GPU memory and computing power. A single GPU can be virtualized into a maximum of 20 GPU virtual 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.	3.6.3.2 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.	3.6.4 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.	3.6.5.3 Resource Usage-based Scheduling
Priority-based scheduling	Scheduling policies are customized based on service importance and priorities to guarantee the resources of key services.	3.6.5.4 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.	3.6.5.5 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.	3.6.5.6 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.	**3.6.6.1 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.	**3.6.6.2 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.	**3.6.6.3 Egress Network Bandwidth Guarantee**

3.6.2 CPU Scheduling

3.6.2.1 CPU Policy

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 thus 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 and CPU cache affinity and scheduling latency significantly affect workload performance, kubelet allows alternative CPU management policies (CPU binding) to determine some placement preferences on the node. The CPU manager preferentially allocates resources on a socket and full physical cores to avoid interference.

Constraints

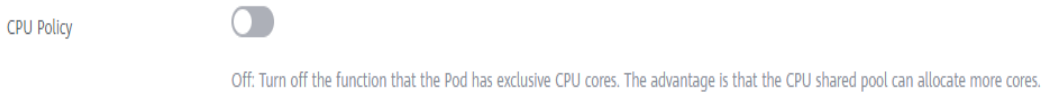
The CPU management policy cannot take effect on physical cloud server nodes.

Enabling the CPU Management Policy

A **CPU management policy** is specified by the kubelet flag **--cpu-manager-policy**. By default, Kubernetes supports the following policies:

- Disabled (**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.
- Enabled (**static**): The **static** policy allows containers in **guaranteed** pods with integer GPU requests to be granted increased CPU affinity and exclusivity on the node.

When creating a cluster, you can configure the CPU management policy in **Advanced Settings**.



You can also configure the policy in a node pool. The configuration will change the kubelet flag **--cpu-manager-policy** on the node. Log in to the CCE console, click the cluster name, access the cluster details page, and choose **Nodes** in the navigation pane. On the page displayed, click the **Node Pools** tab. Choose **More** > **Manage** in the **Operation** column of the target node pool, and change the value of **cpu-manager-policy** to **static**.

Allowing Pods to Exclusively Use the CPU Resources

Prerequisites:

- Enable the **static** policy on the node. For details, see **Enabling the CPU Management Policy**.
- 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 **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)** 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
```

3.6.2.2 Enhanced CPU Policy

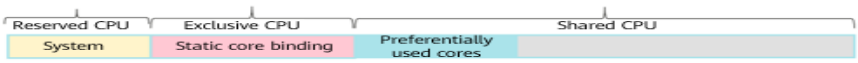
Kubernetes provides two **CPU policies**: none and static.

- **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.

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 cannot take effect on physical cloud server nodes.

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 In the **Manage Components** window that is displayed, change the **cpu-manager-policy** value of the kubelet component to **enhanced-static**.

Figure 3-91 CPU policy

Step 5 Click **OK**.

----End

Verification

Take a node with 8 vCPUs and 32 GB memory as an example. Deploy a workload whose CPU request is 1 and limit is 2 in the cluster in advance.

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-986d5d8919b9/5ba5603434b95fd22d36fba6a5f1c44eba83c18c2e1de9b52ac9b52e93547a1
3/cpuset.preferred_cpus
```

If the following command output is displayed, CPU 1 is preferentially used.

```
1
```

----End

3.6.3 GPU Scheduling

3.6.3.1 Default GPU Scheduling in Kubernetes

You can use GPUs in CCE containers.

Prerequisites

- A GPU node has been created. For details, see **3.3.4 Creating a Node**.
- The gpu-device-plugin (previously gpu-beta add-on) has been installed. During the installation, select the GPU driver on the node. For details, see **3.14.11 CCE AI Suite (NVIDIA GPU)**.
- gpu-device-plugin mounts the driver directory to **/usr/local/nvidia/lib64**. To use GPU resources in a container, add **/usr/local/nvidia/lib64** to the **LD_LIBRARY_PATH** environment variable.

Generally, you can use any of the following methods to add a file:

a. 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
```

b. 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 && ..."
```

c. 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

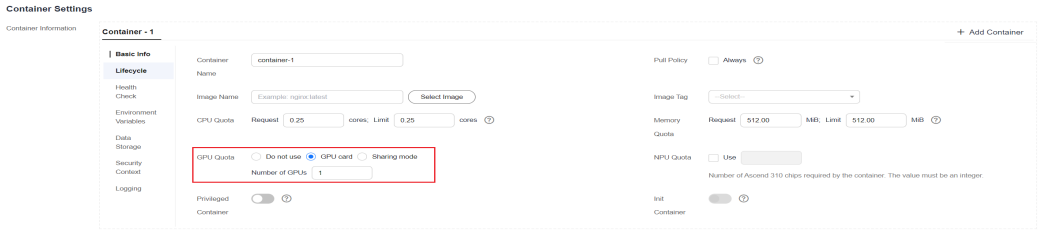
When you use **nvidia.com/gpu** to specify the number of GPUs, the values of requests and limits must be the same.

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 3-92 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
```

3.6.3.2 GPU Virtualization

3.6.3.2.1 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 power ratio and GPU memory size are configured in a refined manner. The computing power allocation granularity is 5% GPU, and the GPU memory allocation granularity is MB.
- **Isolated:** A single GPU memory can be isolated and both the computing power and GPU memory can also be isolated at the same time.
- **Compatible:** Services do not need to be recompiled or the CUDA library does not need to be replaced.

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: • **3.14.13 Volcano Scheduler**: 1.10.5 or later • **3.14.11 CCE AI Suite (NVIDIA GPU)**: 2.0.5 or later

Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- After GPU virtualization is used, init containers are not supported.
- 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 nodes.
- 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.

3.6.3.2.2 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: • **3.14.13 Volcano Scheduler**: 1.10.5 or later • **3.14.11 CCE AI Suite (NVIDIA GPU)**: 2.0.5 or later

Step 1: Install the Add-on

Both **3.14.11 CCE AI Suite (NVIDIA GPU)** and **3.14.13 Volcano Scheduler** must be installed in the cluster.

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 **CCE AI Suite (NVIDIA GPU)** on the right and click **Install**.

Step 3 On the displayed page, configure the add-on.

- **Add-on Specifications**: Select **Default** or **Custom** as required.
- **Containers:** Configurable only when **Add-on Specifications** is set to **Custom**.
- **NVIDIA Driver**: Enter the address of the NVIDIA driver. All GPU nodes in the cluster use the same driver.

NOTICE

- If the download link is a public network address, for example, **https://us.download.nvidia.com/tesla/470.57.02/NVIDIA-Linux-x86_64-470.57.02.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.
- After the driver version is changed, restart the node for the change to take effect.

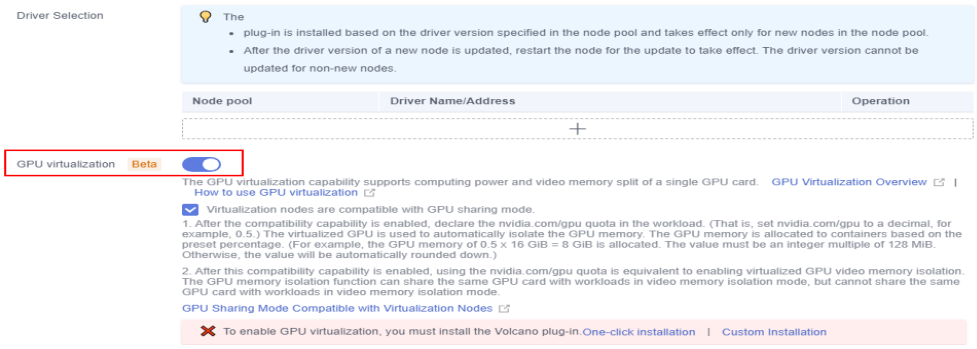
- **Driver Selection**: If you do not want all GPU nodes in a cluster to use the same driver, CCE allows you to install a different GPU driver for each node pool.

  NOTE

  - The add-on installs the driver with the version specified by the node pool. The driver takes effect only for new pool nodes.
  - 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.

- **GPU virtualization** (supported in 2.0.5 and later versions): Enable GPU virtualization to support the segmentation and isolation for the compute power and GPU memory of a single GPU.

Figure 3-93 Enabling GPU Virtualization



If the Volcano add-on has not been installed in the cluster, GPU virtualization cannot be enabled. Click **One-click installation** to install it. To configure the Volcano add-on parameters during installation, click **Custom Installation**. For details, see **3.14.13 Volcano Scheduler**.

If the Volcano add-on has been installed in the cluster but its version does not support GPU virtualization, click **Upgrade** to upgrade it. To configure the Volcano add-on parameters during installation, click **Custom Upgrade**. For details, see **3.14.13 Volcano Scheduler**.

NOTE

After GPU virtualization is enabled, select **Virtualization nodes are compatible with GPU sharing mode**, that is, **default GPU scheduling in Kubernetes** is supported. This capability requires that the version of gpu-device-plugin is 2.0.10 or later and the version of Volcano is 1.10.5 or later.

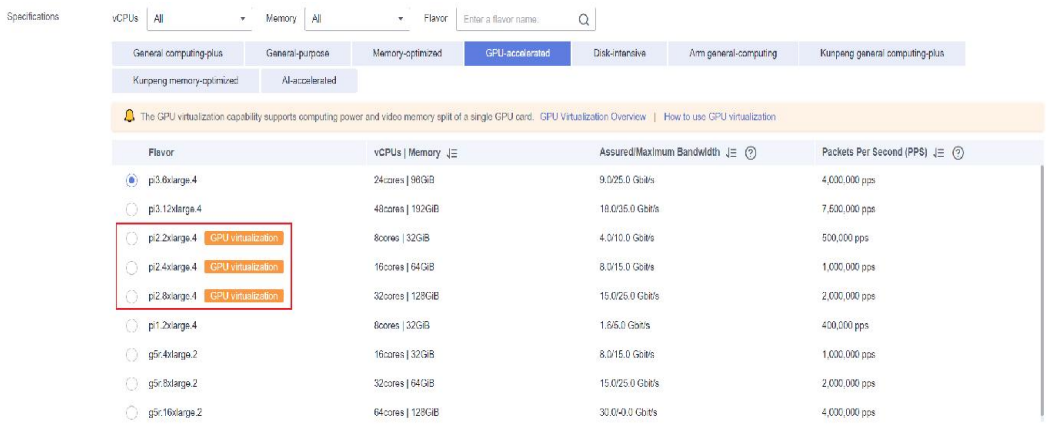
- If you enable compatibility, the **nvidia.com/gpu** quota specified in workloads (the **nvidia.com/gpu** quota is set to a decimal fraction, for example, 0.5) is provided by GPU virtualization to implement GPU memory isolation. The GPU memory is allocated to containers based on the specified quota. For example, 8 GiB (0.5 x 16 GiB) GPU memory is allocated. The value of GPU memory must be an integer multiple of 128 MiB. Otherwise, the value is automatically rounded down to the nearest integer. If **nvidia.com/gpu** resources have been used in the workload before compatibility is enabled, the resources will not be provided by GPU virtualization but the entire GPU.
- After compatibility is enabled, if you use the **nvidia.com/gpu** quota, it is equivalent to enabling GPU memory isolation. The **nvidia.com/gpu** quota can share a GPU with workloads in GPU memory isolation mode, but cannot share a GPU with workloads in compute and GPU memory isolation mode. In addition, **Constraints** on GPU virtualization must be followed.
- If compatibility is disabled, the **nvidia.com/gpu** quota specified in the workload only affects the scheduling result. It does not require GPU memory isolation. That is, although the **nvidia.com/gpu** quota is set to 0.5, you can still view complete GPU memory in the container. In addition, workloads using **nvidia.com/gpu** resources and workloads using virtualized GPU memory cannot be scheduled to the same node.
- If you deselect **Virtualization nodes are compatible with GPU sharing mode**, running workloads will not be affected, but workloads may fail to be scheduled. For example, if compatibility is disabled, the workload using **nvidia.com/gpu** resources are still in the GPU memory isolation mode. As a result, the GPU cannot schedule workloads in compute and GPU memory isolation mode. You need to delete workloads using **nvidia.com/gpu** resources before rescheduling.

Step 4 Click **Install**.

----End

Step 2: Create a GPU Node

Create nodes that support GPU virtualization in the cluster to use the GPU virtualization function. For details, see **3.3.4 Creating a Node** or **3.4.2 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 **Add-ons**.

Step 2 Find the Volcano add-on on the right and click **Edit**.

Step 3 On the displayed page, modify the advanced settings.

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**).

A complete example is as follows:

```
{
  "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",
            // Set the priority of the node with the fewest allocated resources to 0.
            "arguments": {
```



```
                    "leastrequested.weight": 0
                }
            }
        ]
    },
    {
        "plugins": [
            {
                "name": "cce-gpu-topology-predicate"
            },
            {
                "name": "cce-gpu-topology-priority"
            },
            {
                "name": "xgpu"
            },
            // Add the bin packing add-on, and specify the weights of xGPU resources.
            {
                "name": "binpack",
                "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"
            }
        ]
    }
  ]
},
"tolerations": [
    {
        "effect": "NoExecute",
        "key": "node.kubernetes.io/not-ready",
        "operator": "Exists",
        "tolerationSeconds": 60
    },
    {
        "effect": "NoExecute",
        "key": "node.kubernetes.io/unreachable",
        "operator": "Exists",
        "tolerationSeconds": 60
    }
]
}
```

----End

3.6.3.2.3 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 connect to the cluster. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Constraints

- A single GPU can be virtualized into a maximum of 20 xGPU devices.
- After GPU virtualization is used, init containers are not supported.
- 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 nodes.
- 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.

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.

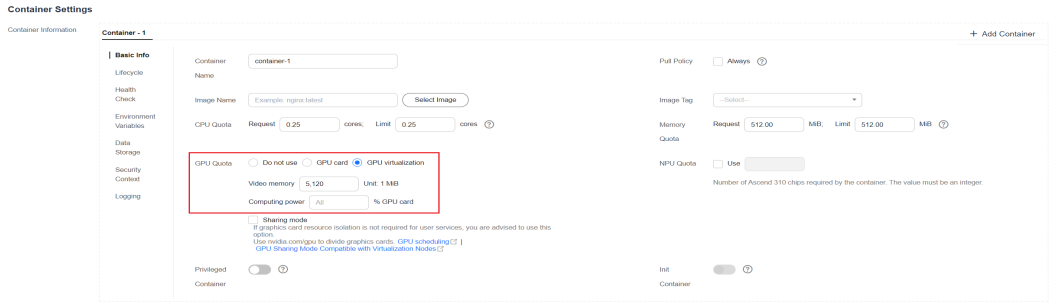
Choose **Container Settings > Basic Info** and configure the GPU quota.

- **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 3-94 Configuring the xGPU quota



This section describes how to use GPU virtualization. For details about other parameters, see **3.5 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 connect to 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 3-121 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.
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

3.6.3.2.4 Supporting Kubernetes' Default GPU Scheduling

After GPU virtualization is enabled, the target GPU node does not support the workloads that use **Kubernetes' default GPU scheduling** by default, which are workloads using nvidia.com/gpu resources. If there are workloads using nvidia.com/gpu resources in your cluster, you can enable the GPU node to support GPU sharing in the gpu-device-plugin configuration so that the GPU node can support Kubernetes' default GPU scheduling.

- If you enable compatibility, the **nvidia.com/gpu** quota specified in workloads (the **nvidia.com/gpu** quota is set to a decimal fraction, for example, 0.5) is provided by GPU virtualization to implement GPU memory isolation. The GPU memory is allocated to containers based on the specified quota. For example, 8 GiB (0.5 x 16 GiB) GPU memory is allocated. The value of GPU memory must be an integer multiple of 128 MiB. Otherwise, the value is automatically rounded down to the nearest integer. If **nvidia.com/gpu** resources have been used in the workload before compatibility is enabled, the resources will not be provided by GPU virtualization but the entire GPU.
- After compatibility is enabled, if you use the **nvidia.com/gpu** quota, it is equivalent to enabling GPU memory isolation. The **nvidia.com/gpu** quota can share a GPU with workloads in GPU memory isolation mode, but cannot share a GPU with workloads in compute and GPU memory isolation mode. In addition, **Constraints** on GPU virtualization must be followed.

- If compatibility is disabled, the **nvidia.com/gpu** quota specified in the workload only affects the scheduling result. It does not require GPU memory isolation. That is, although the **nvidia.com/gpu** quota is set to 0.5, you can still view complete GPU memory in the container. In addition, workloads using **nvidia.com/gpu** resources and workloads using virtualized GPU memory cannot be scheduled to the same node.
- If you deselect **Virtualization nodes are compatible with GPU sharing mode**, running workloads will not be affected, but workloads may fail to be scheduled. For example, if compatibility is disabled, the workload using **nvidia.com/gpu** resources are still in the GPU memory isolation mode. As a result, the GPU cannot schedule workloads in compute and GPU memory isolation mode. You need to delete workloads using **nvidia.com/gpu** resources before rescheduling.

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.

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**.

Step 2 Locate **CCE AI Suite (NVIDIA GPU)** on the right and click **Install**.

If the add-on has been installed, click **Edit**.

Step 3 Configure the add-on. For details, see **Installing the add-on**.

After GPU virtualization is enabled, you can configure the **nvidia.com/gpu** field to enable or disable the function of supporting Kubernetes' default GPU scheduling.

Step 4 Click **Install**.

----End

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:
  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 x 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 x 128 MiB = 2304 MiB).

----End

3.6.3.3 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 **3.14.17 Cloud Native Cluster Monitoring** add-on has been installed in the cluster.
- The **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on has been installed in the cluster, and the add-on version is 2.0.10 or later.
- The **3.14.13 Volcano Scheduler** add-on 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.

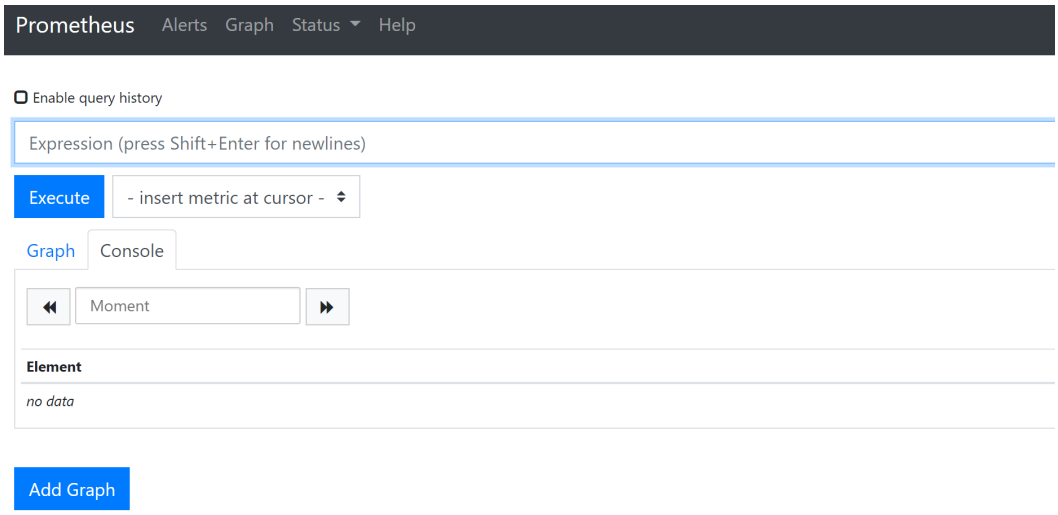
Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

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
```

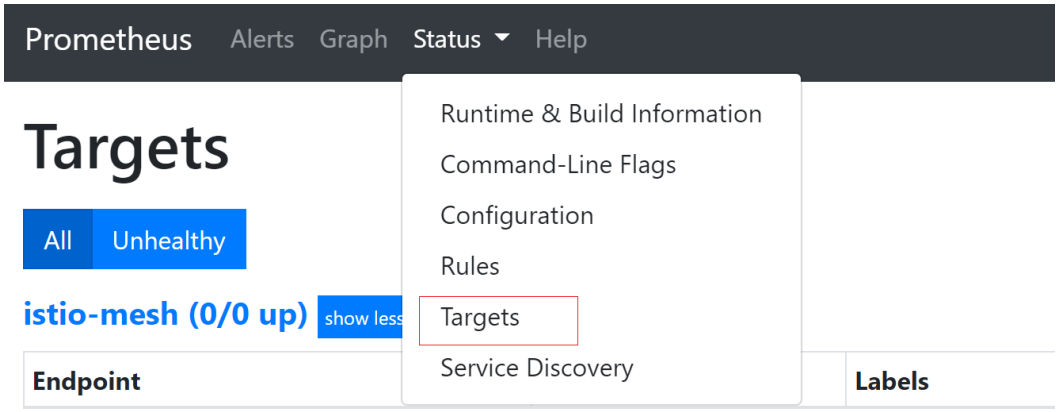
Step 3 After the Service is created, visit ***Public IP address of the load balancer.Service port*** to access Prometheus.

Figure 3-95 Accessing Prometheus



Step 4 Choose **Status** > **Targets** to view the targets monitored by Prometheus.

Figure 3-96 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.

Figure 3-97 Viewing GPU metrics

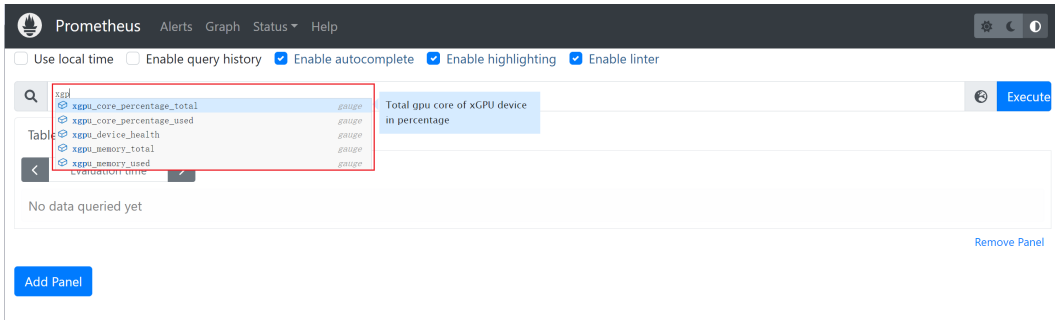


Table 3-122 Basic GPU monitoring metrics

Type	Metric	Monitoring Level	Description
Utilization	cce_gpu_utilization	GPU cards	GPU compute usage
	cce_gpu_memory_utilization	GPU cards	GPU memory usage
	cce_gpu_encoder_utilization	GPU cards	GPU encoding usage
	cce_gpu_decoder_utilization	GPU cards	GPU decoding usage
	cce_gpu_utilization_process	GPU processes	GPU compute usage of each process
	cce_gpu_memory_utilization_process	GPU processes	GPU memory usage of each process
	cce_gpu_encoder_utilization_process	GPU processes	GPU encoding usage of each process
	cce_gpu_decoder_utilization_process	GPU processes	GPU decoding usage of each process
Memory	cce_gpu_memory_used	GPU cards	Used GPU memory
	cce_gpu_memory_total	GPU cards	Total GPU memory
	cce_gpu_memory_free	GPU cards	Free GPU memory
	cce_gpu_bar1_memory_used	GPU cards	Used GPU BAR1 memory
	cce_gpu_bar1_memory_total	GPU cards	Total GPU BAR1 memory
Frequency	cce_gpu_clock	GPU cards	GPU clock frequency
	cce_gpu_memory_clock	GPU cards	GPU memory frequency
	cce_gpu_graphics_clock	GPU cards	GPU frequency
	cce_gpu_video_clock	GPU cards	GPU video processor frequency
Physical status	cce_gpu_temperature	GPU cards	GPU temperature
	cce_gpu_power_usage	GPU cards	GPU power

Type	Metric	Monitoring Level	Description
	cce_gpu_total_energy_consumption	GPU cards	Total GPU energy consumption
Bandwidth	cce_gpu_pcie_link_bandwidth	GPU cards	GPU PCIe bandwidth
	cce_gpu_nvlink_bandwidth	GPU cards	GPU NVLink bandwidth
	cce_gpu_pcie_throughput_rx	GPU cards	GPU PCIe RX bandwidth
	cce_gpu_pcie_throughput_tx	GPU cards	GPU PCIe TX bandwidth
	cce_gpu_nvlink_utilization_counter_rx	GPU cards	GPU NVLink RX bandwidth
	cce_gpu_nvlink_utilization_counter_tx	GPU cards	GPU NVLink TX bandwidth
Memory isolation page	cce_gpu_retired_pages_sbe	GPU cards	Number of isolated GPU memory pages with single-bit errors
	cce_gpu_retired_pages_dbe	GPU cards	Number of isolated GPU memory pages with dual-bit errors

Table 3-123 xGPU metrics

Metric	Monitoring Level	Description
xgpu_memory_total	GPU processes	Total xGPU memory
xgpu_memory_used	GPU processes	Used xGPU memory
xgpu_core_percentage_total	GPU processes	Total xGPU cores
xgpu_core_percentage_used	GPU processes	Used xGPU cores

Metric	Monitoring Level	Description
gpu_schedule_policy	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	GPU cards	Health status of an xGPU device. Options: • **0**: The xGPU device is healthy. • **1**: The xGPU device is not healthy.

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.

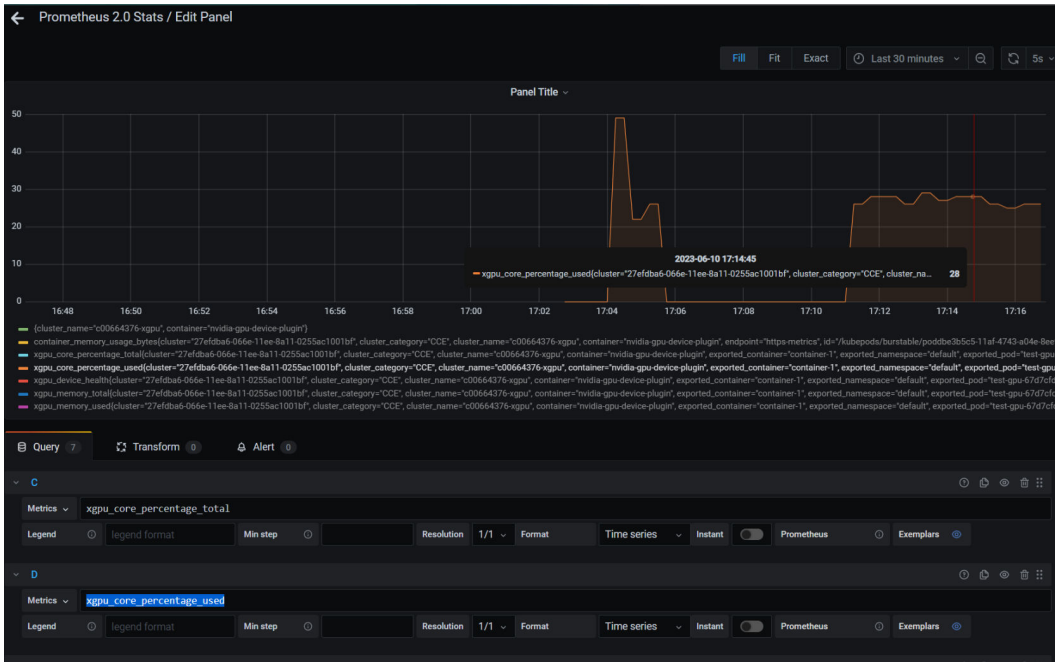
Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the cluster console. In the navigation pane, choose **Services & Ingresses**.

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 3-98 Viewing xGPU resources



----End

3.6.3.4 GPU-based HPA Practice

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

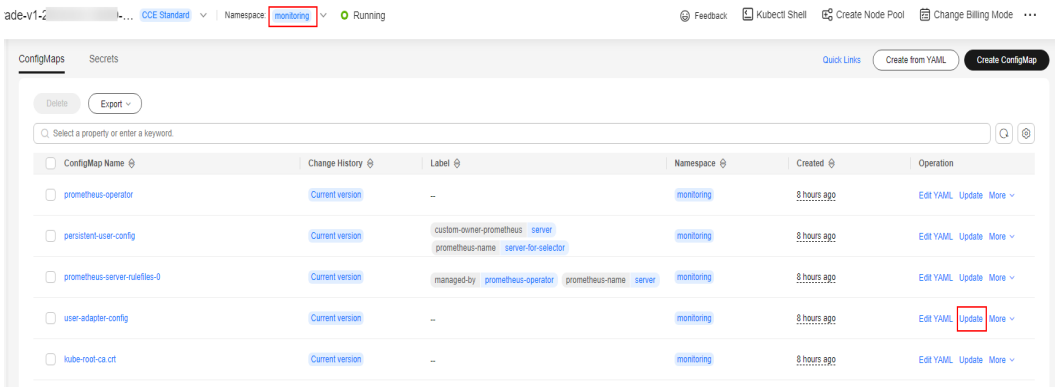
- A cluster is available, and there are GPU nodes and GPU related services running in the cluster.
- The **3.14.11 CCE AI Suite (NVIDIA GPU)** add-on has been installed in the cluster, and the add-on metrics API is working properly. You can log in to the GPU node and run the following command:
  ```
  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 **3.14.17 Cloud Native Cluster Monitoring** add-on of v3.9.5 or later has been installed in the cluster, and it is deployed in server 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 3-99 Updating a ConfigMap



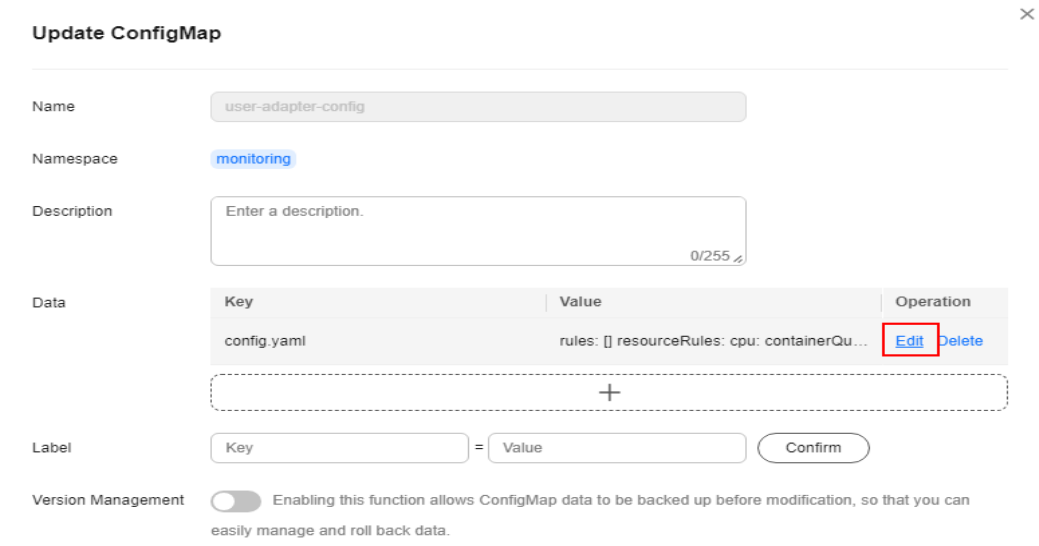
Step 3 On the **Update ConfigMap** page, click **Edit** in the **Operation** column of the **config.yaml** file in the **Data** pane. Then, add a custom metric collection rule under the **rules** field. Click **OK**.

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**.

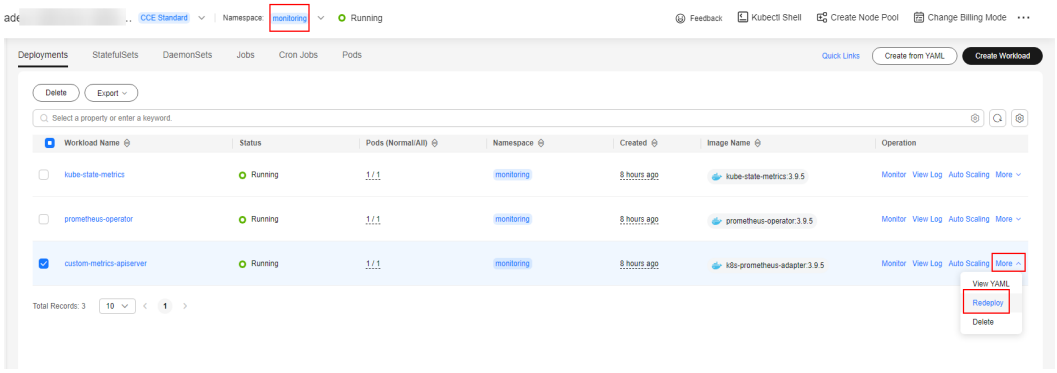
```
rules:
- seriesQuery: '{__name__=~"cce_gpu_memory_utilization",container!="",namespace!="",pod!=""}'
  seriesFilters: []
  resources:
    overrides:
      namespace:
        resource: namespace
      pod:
        resource: pod
  metricsQuery: sum(last_over_time(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<<.GroupBy>>)
```

Figure 3-100 Customizing a collection rule



Step 4 Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

Figure 3-101 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

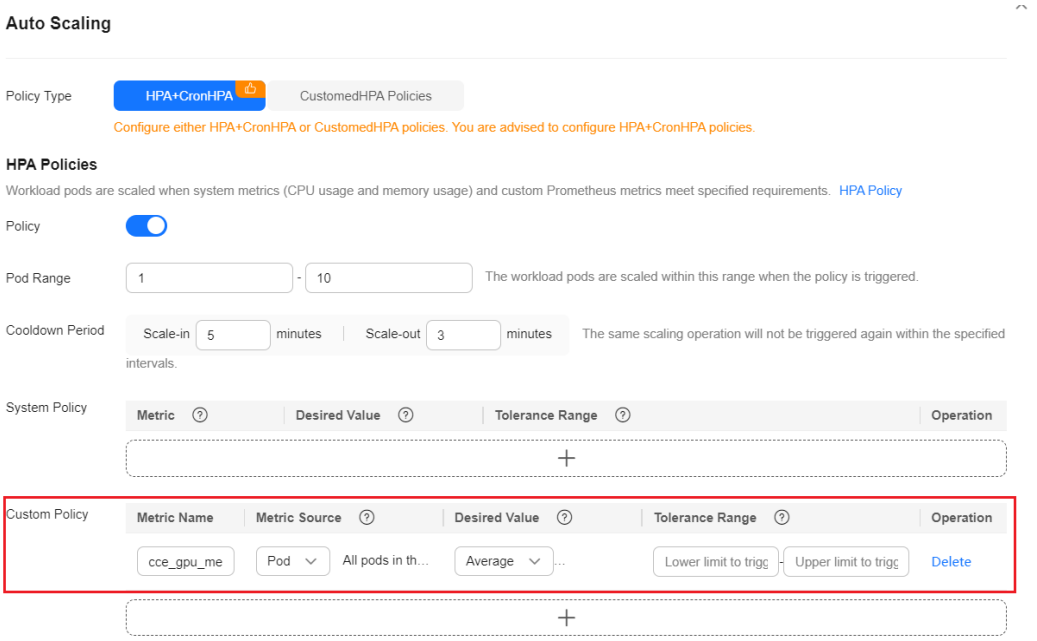
Creating an Auto Scaling Policy

Step 1 Choose **Workloads** in the navigation pane. Locate the target workload and choose **More** > **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.

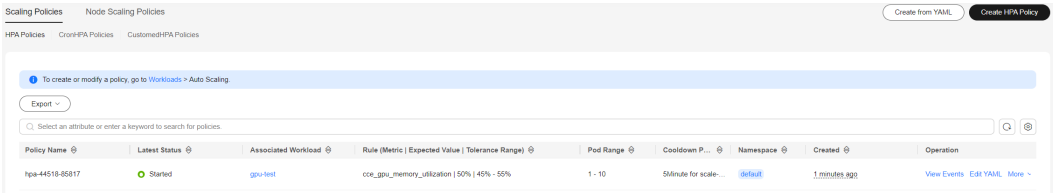
Figure 3-102 Selecting custom metrics



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 **3.12.2.2 HPA Policies**.

Step 3 Return to the **Scaling Policies** page and check whether the HPA policy has been created.

Figure 3-103 HPA policy created



----End

3.6.3.5 GPU Fault Handling

Prerequisites

3.14.18 Cloud Native Logging 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 3-124**.

Table 3-124 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.
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.

3.6.4 NPU Scheduling

You can use NPUs in CCE containers.

Prerequisites

- An NPU node has been created. For details, see **3.3.4 Creating a Node**.
- The huawei-npu has been installed. For details, see **3.14.12 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**.

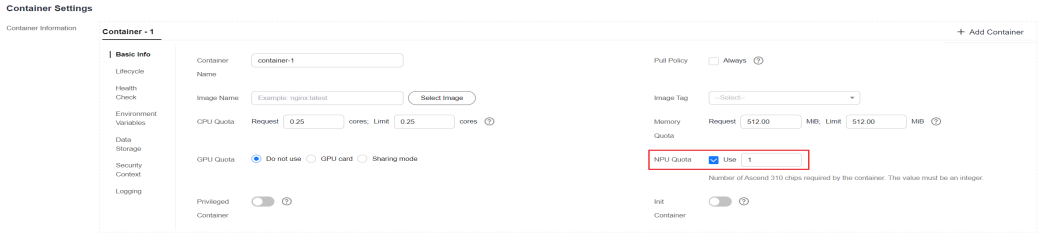
NOTE

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 3-104 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
```

3.6.5 Volcano Scheduling

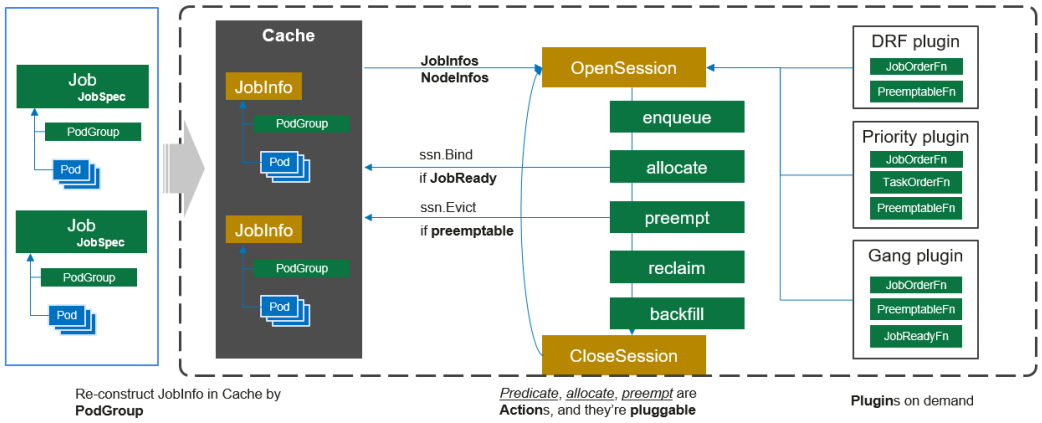
3.6.5.1 Overview

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.

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 3-105 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.

3.6.5.2 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**.

Constraints

When a large number of workloads are scheduled, Volcano prints a large number of logs. In this case, you can use Volcano with LTS. Otherwise, the disk space of the node where Volcano resides may be used up. For details, see **3.9.4.2 Collecting Container Logs**.

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:

```
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 3-125 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 and the allocated queue.

```
kubectl describe pod <pod_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
```

3.6.5.3 Resource Usage-based Scheduling

3.6.5.3.1 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **3.14.13 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

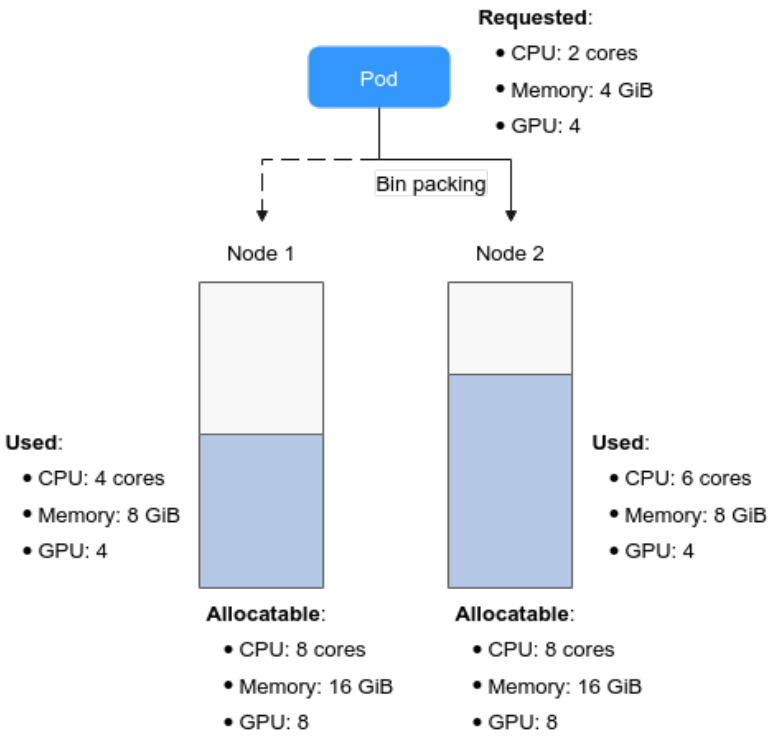
The bin packing add-on calculates the score of a node using the following formula:

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 3-106 Bin packing example



As shown in the figure, there are two nodes in the cluster. When pods need to be scheduled, the bin packing policy scores the two nodes separately.

1. The scoring for node 1 is as follows:

   Each resource is scored using the following formula: CPU.weight x (Requested + Used)/Allocatable

- CPU score: 1 x (2 + 4)/8 = 0.75
- Memory score: 1 x (4 + 8)/16 = 0.75
- GPU score: 2 x (4 + 4)/8 = 1

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

Score of node 1: 5 x (0.75 + 0.75 + 1)/(1 + 1 + 2) x 100 = 312.5

2. The scoring for node 2 is as follows:
   - CPU score: 1 x (2 + 6)/8 = 1
   - Memory score: 1 x (4 + 8)/16 = 0.75
   - GPU score: 2 x (4 + 4)/8 = 1

   Score of node 2: 5 x (1 + 0.75 + 1)/(1 + 1 + 2) x 100 = 343.75

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.

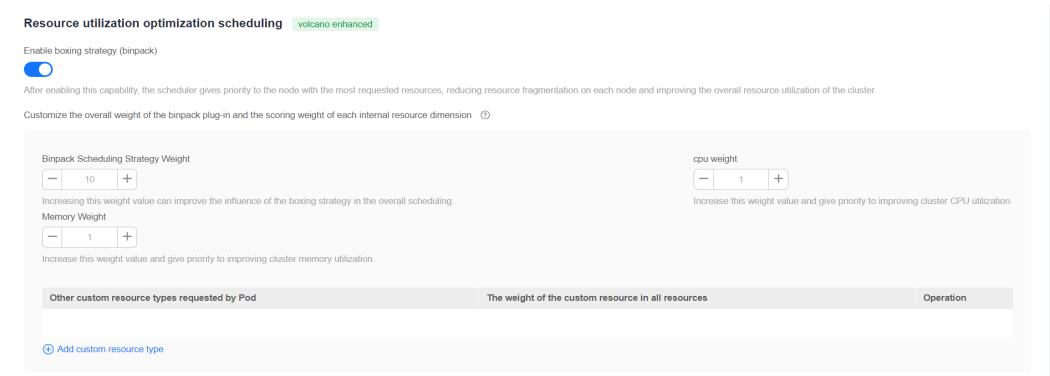
Step 2 Click the cluster name to access the cluster console. Choose **Settings** in the navigation pane. In the right pane, click the **Scheduling** tab.

Step 3 In the **Resource utilization optimization scheduling** area, modify the bin packing settings.

Table 3-126 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

Figure 3-107 Resource utilization optimization scheduling



Step 4 Click **Confirm**.

----End

3.6.5.3.2 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.

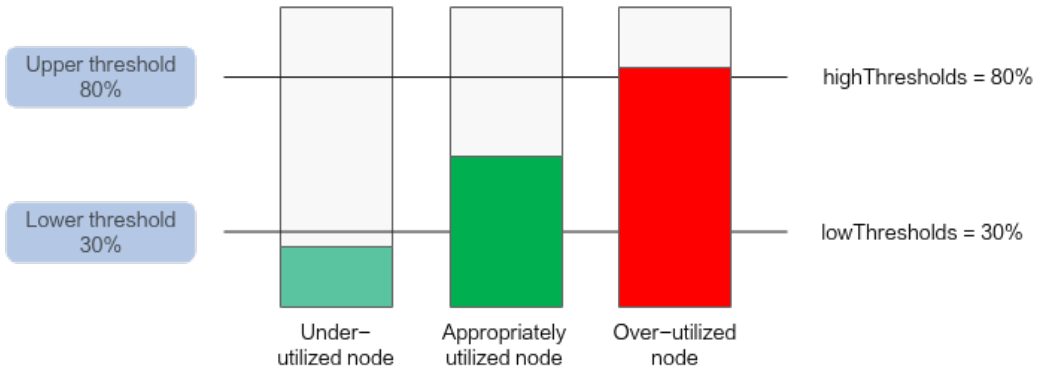
CCE has resolved this issue by using Volcano Scheduler to evict pods that do not comply with the configured policy so that pods can be rescheduled. In this way, the cluster load is balanced and resource fragmentation is minimized.

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 3-108 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see **3.14.13 Volcano Scheduler**.

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 In the navigation pane, choose **Add-ons**. Locate **Volcano Scheduler** on the right and click **Install** or **Edit**.

Step 3 In the **Parameters** area, modify **Advanced Settings** to configure the load-aware descheduling policy. The following shows a configuration example for Volcano 1.11.21 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 3-127 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 3-128**.

Table 3-128 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. If the node value 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 **3.6.5.3.1 Bin Packing**.

Step 2 In the navigation pane, choose **Add-ons**. Locate **Volcano Scheduler** on the right and click **Install** or **Edit**.

Step 3 In the **Parameters** area, modify **Advanced Settings** to configure the HighNodeUtilization policy.

```
{
  "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 3-129 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 3-130**.

Table 3-130 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`
	`}`

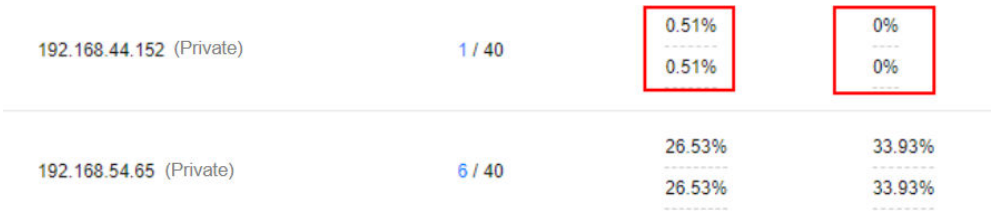
Step 4 Click **OK**.

----End

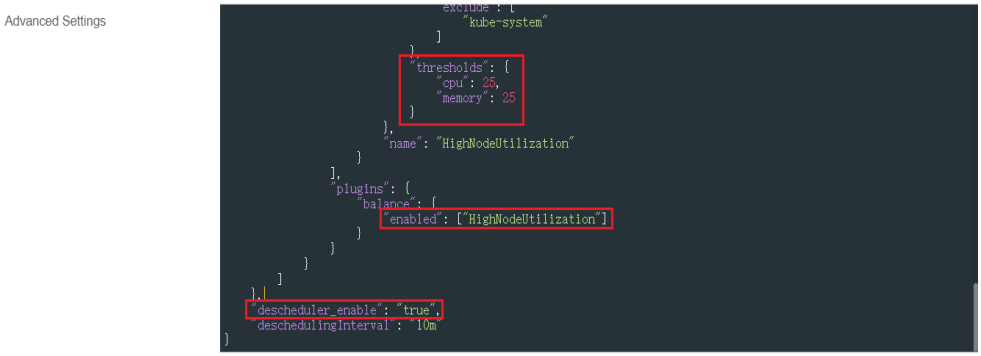
Use Cases

HighNodeUtilization

1. Check the nodes in a cluster. It is found that some nodes are under-utilized.



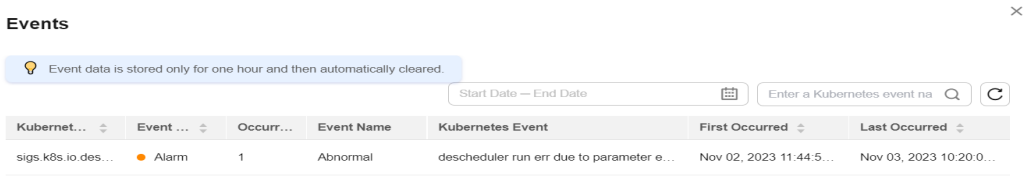
2. Edit the Volcano parameters to enable the descheduler and set the CPU and memory usage thresholds to **25**. When the CPU and memory usage of a node is less than 25%, pods on the node will be evicted.



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.

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.



3.6.5.3.3 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 formula for scoring a node 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

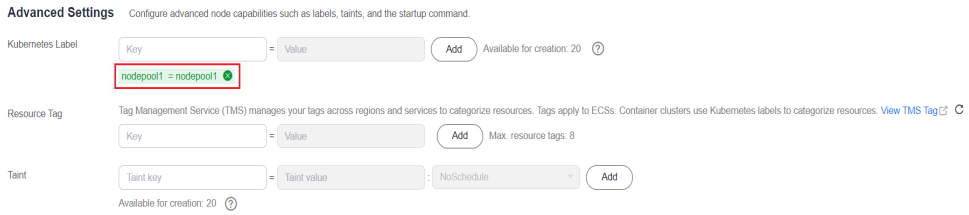
- A cluster of v1.19.16 or later is available. For details, see **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- Volcano of v1.11.5 or later has been installed in the cluster. For details, see **3.14.13 Volcano Scheduler**.

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 Choose **Add-ons** in the navigation pane, locate **Volcano Scheduler** on the right, click **Install** or **Edit**, and configure Volcano scheduler parameters in the **Parameters** area.


```
{
  "ca_cert": "",
  "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"
          }
        ]
      }
```



```
      ]
    },
    "server_cert": "",
    "server_key": ""
}
```

Step 3 Click **OK**.

----End

3.6.5.3.4 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on of v1.11.14 or later has been installed. For details, see **3.14.13 Volcano Scheduler**.
- The kube-prometheus-stack add-on has been installed and it works in server mode. For details, see **3.14.17 Cloud Native Cluster Monitoring**.

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 imbalancing.

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. Certain nodes in a cluster have been overloaded, but this could not been detected by the scheduler. This may greatly affect service stability.

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 imbalancing 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 **3.6.5.3.2 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 customize the weights of 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 x [(1 - CPU usage) x CPU weight + (1 - Memory usage) x 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

Configuring Load-aware Scheduling

Step 1 Obtain resource metrics by calling the metrics API and add custom metric collection rules.

After the kube-prometheus-stack add-on is installed, enable the function of automatically obtaining resource metrics through the metrics API. For details, see **Providing Resource Metrics Through the Metrics API**.

Add custom metric collection rules. For details, see **Creating an HPA Policy Using Custom Metrics**. In the following example rules, the rules in red are new ones and those in black are existing ones:

```
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**.

Step 2 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.

----End

3.6.5.3.5 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**.	3.6.5.3.1 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**.	
	tainttoleration.weight	Pods are scheduled to the node with a high taint tolerance. The parameter defaults to **3**.	

Scheduling Policy	Parameter	Description	Reference
	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**.	**3.6.5.6 NUMA Affinity Scheduling**
Load-aware scheduling (usage)	weight	After this function is enabled, the parameter defaults to **5**.	**3.6.5.3.4 Load-aware Scheduling**
Node pool affinity scheduling (nodepoolaffinity)	nodepoolaffinity.weight	After this function is enabled, the parameter defaults to **10000**.	**3.6.5.3.3 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 **3.6.5.3.1 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 **3.6.5.3.4 Load-aware Scheduling**.
2. Disable bin packing. For details, see **3.6.5.3.1 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 bot 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 **3.6.5.3.2 Descheduling**.

1. Enable load-aware scheduling and use the default policy weight **5**. For details, see **3.6.5.3.4 Load-aware Scheduling**.
2. Enable descheduling and configure the load-aware descheduling policy. For details, see **3.6.5.3.2 Descheduling**.
3. Disable bin packing. For details, see **3.6.5.3.1 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**

3.6.5.4 Priority-based Scheduling

3.6.5.4.1 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **3.14.13 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 3-131 lists the priority-based scheduling supported by CCE clusters.

Table 3-131 Priority-based scheduling

Scheduling Type	Description	Scheduler
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.	kube-scheduler or Volcano scheduler
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.	Volcano scheduler

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.

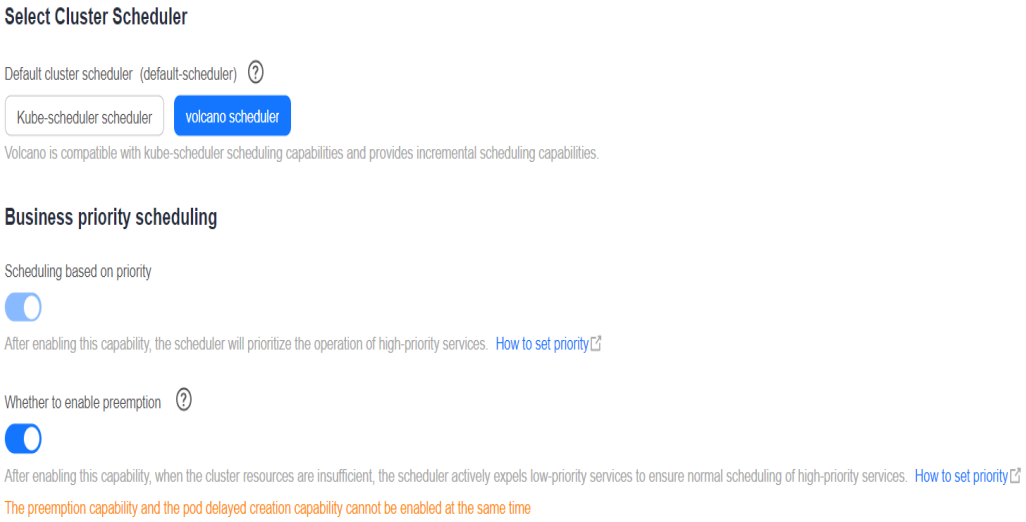
Step 3 In the **Business priority scheduling** area, configure priority-based scheduling.

- **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 3-109 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.

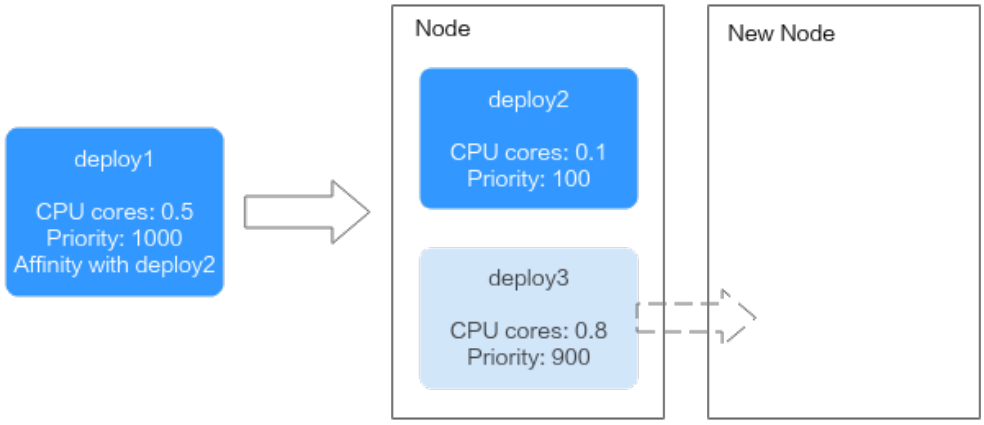
----End

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. For details, see **Inter-pod affinity on lower-priority pods**.

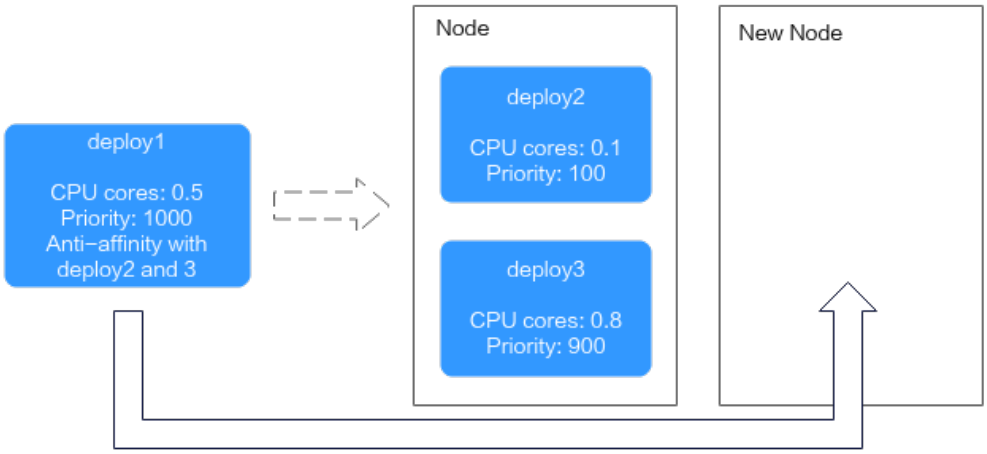
In inter-pod anti-affinity, if priority-based preemption is enabled and deploy1 is affinity with lower-priority deploy2, volcano-scheduler will evict deploy3 and schedule deploy1 to the node to ensure service O&M. The evicted deploy3 will be scheduled to the new node after the new node is ready.

Figure 3-110 Inter-pod affinity on lower-priority pods



In inter-pod anti-affinity, volcano-scheduler does not evict deploy2 and deploy3 to reduce the impact on other services. Instead, volcano-scheduler schedules deploy1 to the new node after the new node is ready.

Figure 3-111 Inter-pod anti-affinity on lower-priority pods



3.6.5.5 AI Performance-based Scheduling

3.6.5.5.1 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **3.14.13 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.

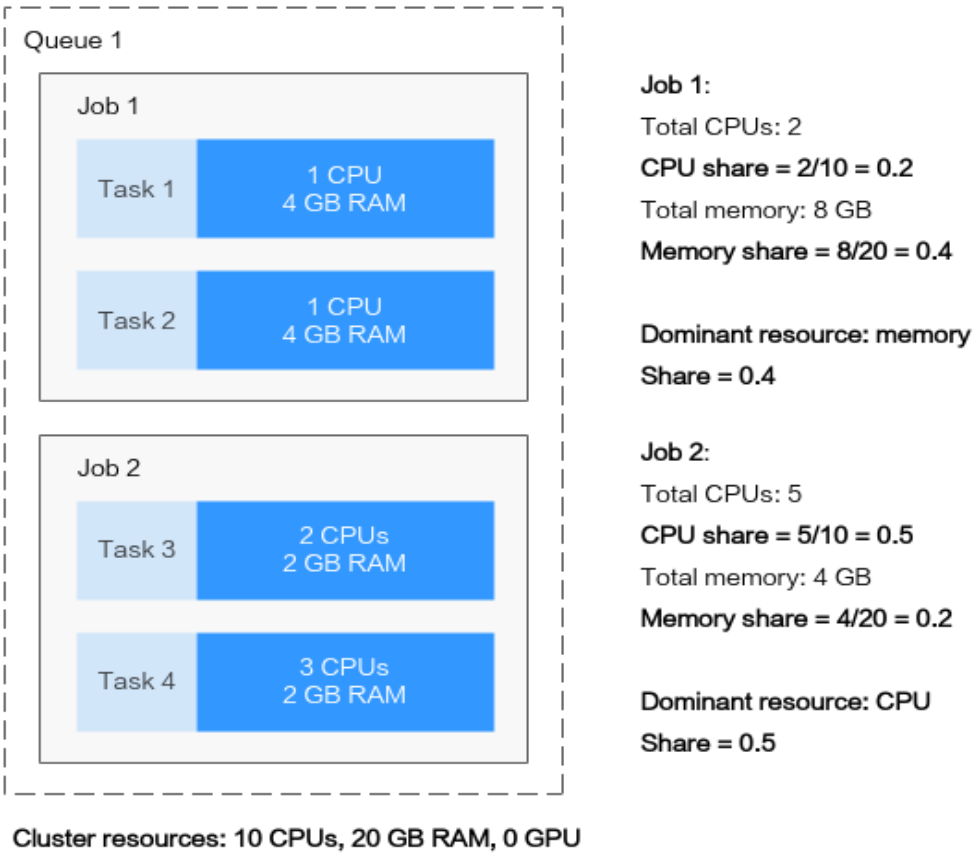
The share value of each job resource is calculated using the following formula:

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 3-112 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

3.6.5.5.2 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 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 **3.2.2.1 Buying a CCE Standard/Turbo Cluster**.
- The Volcano add-on has been installed. For details, see **3.14.13 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

3.6.5.6 NUMA Affinity Scheduling

Background

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. However, in workloads where CPU cache affinity and scheduling latency significantly affect workload performance, additional latency will occur when CPU cores are from different NUMA nodes. To resolve this issue, kubelet allows you to use Topology Manager to replace the CPU management policies to determine node allocation.

Both the CPU Manager and Topology Manager are kubelet components, but they have the following limitations:

- The scheduler is not topology-aware. Therefore, the workload may be scheduled on a node and then fail on the node due to the Topology Manager. This is unacceptable for TensorFlow jobs. If any worker or ps failed on node, the job will fail.
- The managers are node-level that results in an inability to match the best node for NUMA topology in the whole cluster.

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.

For more information, see **https://github.com/volcano-sh/volcano/blob/master/docs/design/numa-aware.md**.

Application Scope

- CPU resource topology scheduling
- Pod-level topology policies

Pod Scheduling Prediction

After a topology policy is configured for pods, Volcano predicts matched nodes based on the topology policy. 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.

Volcano Topology Policy	Node Scheduling	
	1. Filter nodes with the same policy.	**2. Check whether node's CPU topology meets the policy requirements.**
none	No filtering: • **none**: schedulable • **best-effort**: schedulable • **restricted**: schedulable • **single-numa-node**: schedulable	None
best-effort	Filter the nodes with the **best-effort** topology policy. • **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.

Volcano Topology Policy	Node Scheduling	
	1. Filter nodes with the same policy.	**2. Check whether node's CPU topology meets the policy requirements.**
restricted	Filter the nodes with the **restricted** topology policy. • **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	Filter the nodes with the **single-numa-node** topology policy. • **none**: unschedulable • **best-effort**: unschedulable • **restricted**: unschedulable • **single-numa-node**: schedulable	Pods can only be scheduled to a single NUMA node.

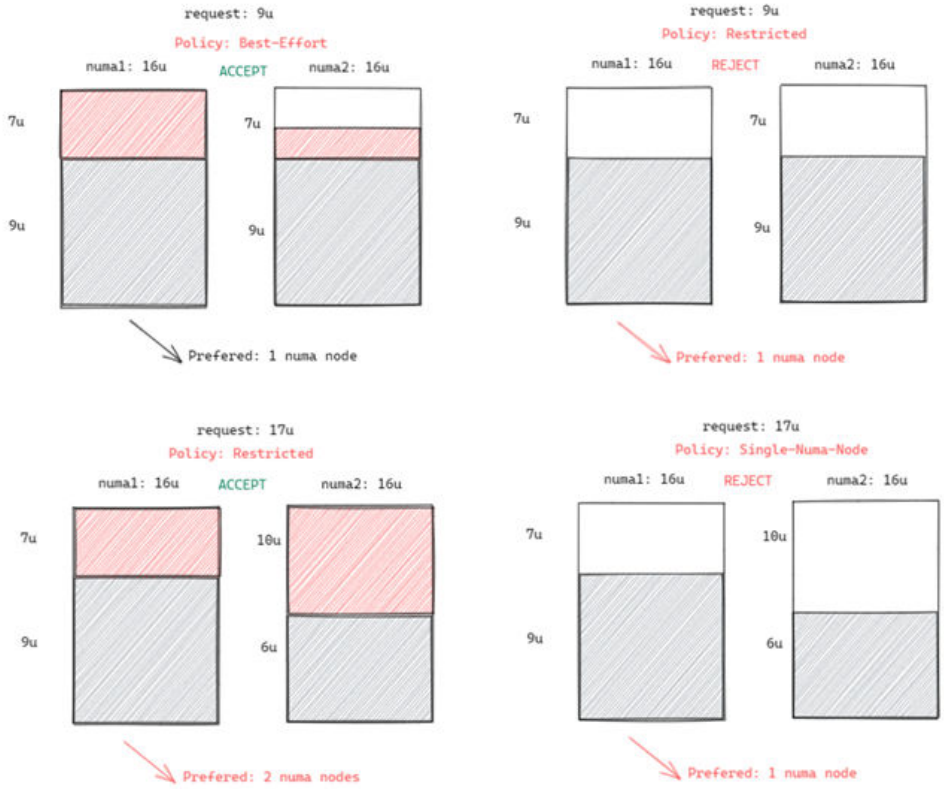
For example, two NUMA nodes provide resources, each with a total of 32 CPU cores. The following table lists resource allocation.

Worker Node	Node Topology Policy	Total CPU Cores on NUMA Node 1	Total CPU Cores on NUMA Node 2
Node 1	best-effort	16	16
Node 2	restricted	16	16
Node 3	restricted	16	16
Node 4	single-numa-node	16	16

Figure 3-113 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 9 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 a total of 9 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 3-113 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. For details, see **Enabling the CPU Management Policy**.

Step 2 Configure a CPU topology policy.

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 Prediction**.

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. In the navigation pane, choose **Add-ons**. On the right of the page, locate the **Volcano** add-on and click **Edit**. In the **Parameters** area, configure Volcano scheduler parameters.


```
{
  "ca_cert": "",
  "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"
          },
          {
            // add this also enable resource_exporter
            "name": "numa-aware",
            // the weight of the NUMA Aware Plugin
            "arguments": {
              "weight": "10"
            }
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
      }
    ]
```



```
  },
  "server_cert": "",
  "server_key": ""
}
```

Volcano earlier than 1.7.1

1. The **resource_exporter_enable** parameter is enabled for the Volcano add-on to collect node NUMA information.

   ```
   {
     "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
   ```

2. 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

Using Volcano to Configure NUMA Affinity Scheduling

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.

3.6.5.7 Application Scaling Priority Policies

Application scaling priority policies control scaling of applications in clusters. 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.

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.
- The application scaling priority policy function only applies to Deployments (including **ReplicaSets**). Workloads' **spec.schedulerName** or the default cluster scheduler must be set 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 and workload-balancer-score** annotation will be used to determine which pod to scale-in.

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:
    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:
    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 3-132 Key parameters of a Balancer CR

Field	Description	Type	Remarks
metadata.name	Name	String	This field is mandatory.
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.

Field	Description	Type	Remarks

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`
			• 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`

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 3-133 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 new **BalancerPolicyTemplate** CR.

```
kubectl create -f new-balancerpolicytemplate.yaml
```

Step 3 Modify **default-balancer**. You can also create a new **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

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.

3.6.6 Cloud Native Hybrid Deployment

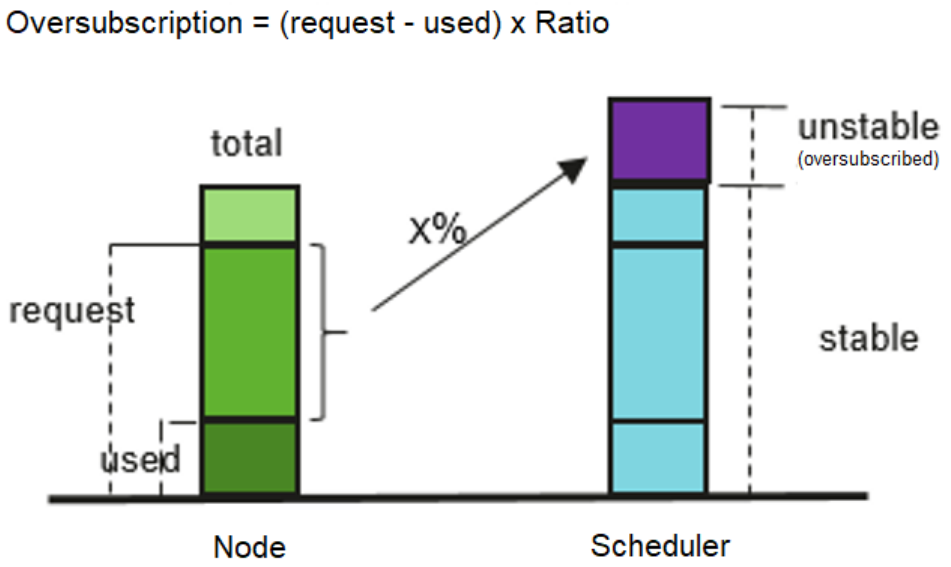
3.6.6.1 Dynamic Resource Oversubscription

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 3-114 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 isolation is provided by kernels.

  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 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:

  If only hybrid deployment is used, configure the label **volcano.sh/colocation=true** for the node and delete the node label **volcano.sh/oversubscription** or set its value to **false**.

  If the label **volcano.sh/colocation=true** is configured for a node, hybrid deployment is enabled. If the label **volcano.sh/oversubscription=true** is configured, resource oversubscription is enabled. The following table lists the available feature combinations after hybrid deployment or resource oversubscription is enabled.

Hybrid Deployment Enabled (volcano.sh/colocation=true)	Resource Oversubscription Enabled (volcano.sh/oversubscription=true)	Resource Oversubscription	When Offline Pod Eviction Triggered (Using Annotations to Configure Limits)
No	No	No	None
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%.

Hybrid Deployment Enabled (volcano.sh/colocation=true)	Resource Oversubscription Enabled (volcano.sh/oversubscription=true)	Resource Oversubscription	When Offline Pod Eviction Triggered (Using Annotations to Configure Limits)
Yes	Yes	Yes	The actual resource usage of a node exceeds the upper limit.

kubelet Oversubscription

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 CCE 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, you need to 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 3-134**.
- 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 **cpu-manager-policy** is set to static core binding on a node, do not assign the QoS class of Guaranteed to offline pods. If core binding is required, change the pods to online pods. Otherwise, offline pods may occupy the CPUs of online pods, causing online pod startup failures, and offline pods fail to be started although they are successfully scheduled.
- If **cpu-manager-policy** is set to static core binding on a node, do not bind cores to all online pods. Otherwise, online pods occupy all CPU or memory resources, leaving 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 3-134**.

Ensure that you have correctly configure labels because the scheduler does not check the add-on and node configurations.

Table 3-134 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 Configure the Volcano add-on.

1. Use kubectl to access the cluster.
2. Install the Volcano add-on and add the oversubscription add-on to **volcano-scheduler-configmap**. Ensure that the add-on configuration does not contain the overcommit add-on. If **- name: overcommit** exists, delete this configuration. In addition, set **enablePreemptable** and **enableJobStarving** of the gang add-on to **false** and configure a preemption action.

```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
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 2 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 Configurations** page, enable **Node oversubscription feature (over-subscription-resource)** and click **OK**.

Step 3 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 3-135**. 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 3-135 Node oversubscription annotations

Name	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. After eviction is triggered, the scheduling starts again when the CPU usage of a node is lower than the specified value. The default value is **30**, indicating that scheduling starts 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%.
volcano.sh/evicting-memory-low-watermark	Lower limit for memory usage. After eviction is triggered, the scheduling starts again when the memory usage of a node is lower than the specified value. The default value is **30**, indicating that the scheduling starts again when the memory usage of a node is less 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 4 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: production
preemptionPolicy: PreemptLowerPriority
value: 999999
---
apiVersion: scheduling.k8s.io/v1
description: Used for low priority pods
kind: PriorityClass
metadata:
  name: testing
preemptionPolicy: PreemptLowerPriority
value: -99999

EOF
```

Step 5 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 label
    spec:
      schedulerName: volcano              # Volcano is used.
      priorityClassName: testing          # Configure the testing 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: production   # Configure the production priorityClass.
      ...
```


Step 6 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 mCPUs 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: testing         # Configure the testing 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: production        # Configure the 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: production         # Configure the 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 2343 mCPUs 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.

3.6.6.2 CPU Burst

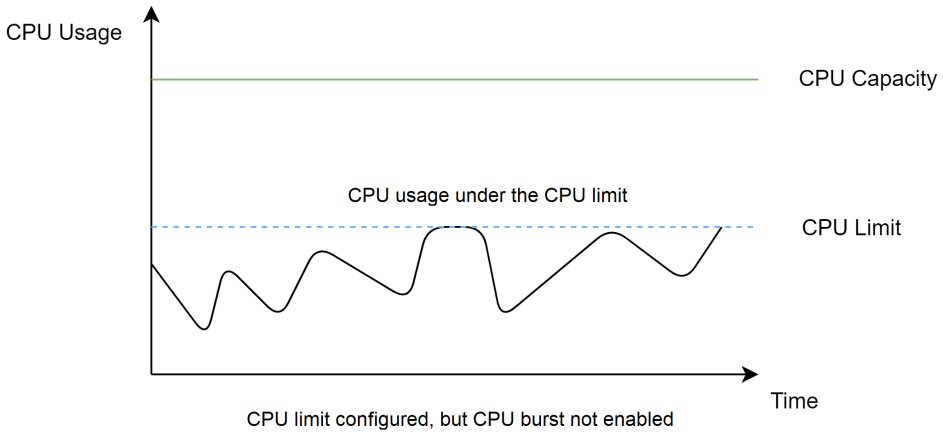
If a CPU limit is set for a container in the pod, the CPU usage of the container cannot exceed the limit. Frequent CPU traffic limiting affects service performance and increases the long-tail response latency, especially for latency-sensitive services.

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.

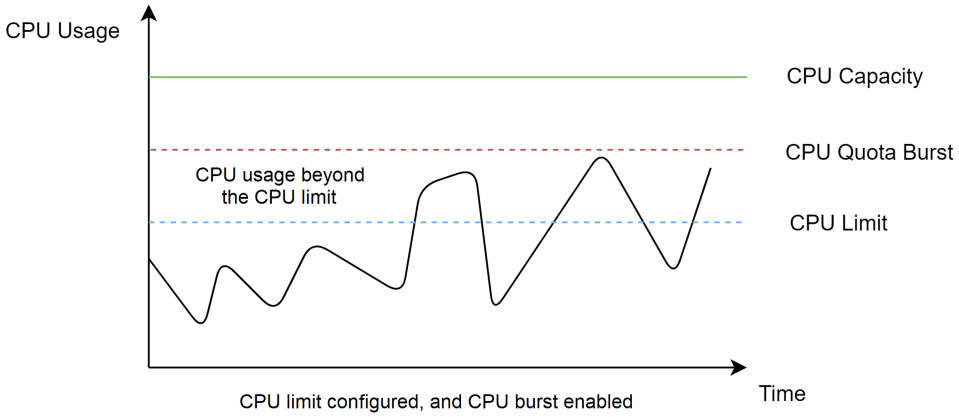
- If CPU burst is not enabled, the CPU quota for a container cannot exceed the limit, and the accumulated burst resources cannot be used.

  Figure 3-115 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 3-116 CPU burst enabled



Constraints

- Cluster version: CCE Turbo cluster v1.23.5-r0 or later
- OS version: Huawei Cloud EulerOS 2.0
- The Volcano add-on of v1.9.0 or later must be installed in the cluster, and the hybrid deployment function must be enabled by setting **colocation_enable** in the advanced settings to **true**.

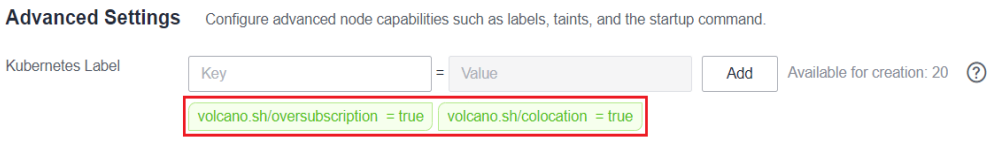
Procedure

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

Step 2 In the navigation pane on the left, choose **Nodes**. Click the **Node Pools** tab. When creating or updating a node pool, enable hybrid deployment of online and offline services in **Advanced Settings**.

- volcano.sh/oversubscription=true
- volcano.sh/colocation=true

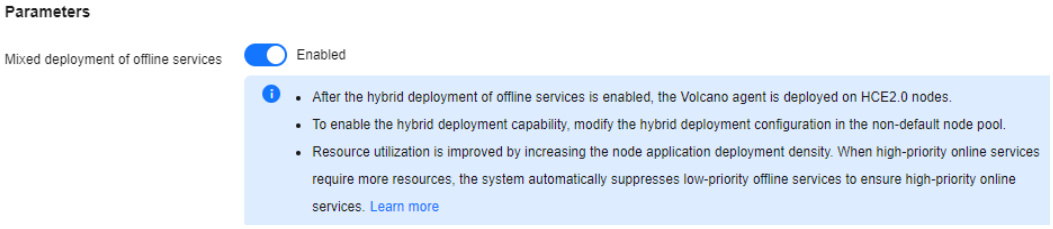
Figure 3-117 Node label settings



Step 3 In the navigation pane, choose **Add-ons** and click **Install** of the Volcano add-on. On the **Install Add-on** page, enable hybrid deployment in the **Parameters** area. For details about the installation, see **3.14.13 Volcano Scheduler**.

If the Volcano add-on has been installed, click **Edit** to view or modify the parameter **colocation_enable**.

Figure 3-118 Enabling hybrid deployment of online and offline services



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 4 Verify that CPU burst has been enabled.

After confirming that the Volcano add-on is working, edit the parameter **configmap** of **volcano-agent-configuration** in the namespace **kube-system**. If **enable** is set to **true**, CPU burst is enabled. If **enable** is set to **false**, CPU burst is disabled.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Example:

```
...
data:
  colocation-config: |
    {
        "globalConfig":{
            "cpuBurstConfig":{
                "enable":true
            },
...
```

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 5 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.

Annotation	Mandatory	Description
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 6 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 3-136 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

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
Enabled	456 μs	0	0	469	3.7s

----End

3.6.6.3 Egress Network Bandwidth Guarantee

Egress network bandwidth guarantee is implemented by setting network priorities. It has the following advantages:

- 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.

Constraints

To use egress network bandwidth guarantee, the following requirements must be met:

- Only nodes running Huawei Cloud EulerOS 2.0 are supported.
- Only CCE Turbo clusters of v1.23 or later are supported.
- The Volcano add-on of v1.9.0 or later must be installed in the cluster, and the hybrid deployment function must be enabled by setting **colocation_enable** in the advanced settings to **true**.
- Before enabling, modifying, or disabling egress network bandwidth guarantee, ensure that the Volcano add-on is working.
- For pods that have been running on the node before the Volcano add-on is installed, manually restart the pods after enabling network bandwidth guarantee so that the feature can take effect.
- Uninstalling the Volcano add-on or disabling the hybrid deployment function (that is, setting **colocation_enable** in the advanced settings to **false**) does not affect the existing egress network bandwidth guarantee on the node. To disable this feature, see **Disabling Egress Network Bandwidth Guarantee**.
- 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 on the left, choose **Nodes**. Click the **Node Pools** tab. When creating or updating a node pool, enable hybrid deployment of online and offline services in **Advanced Settings**.

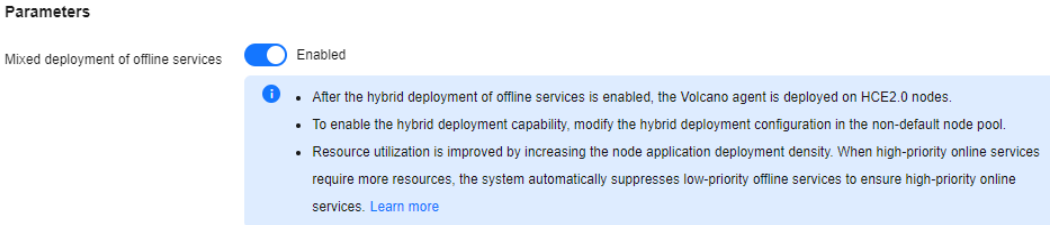
- volcano.sh/oversubscription=true
- volcano.sh/colocation=true

Figure 3-119 Node label settings

Step 3 In the navigation pane, choose **Add-ons** and click **Install** of the Volcano add-on. On the **Install Add-on** page, enable hybrid deployment in the **Parameters** area. For details about the installation, see **3.14.13 Volcano Scheduler**.

If the Volcano add-on has been installed, click **Edit** to view or modify the parameter **colocation_enable**.

Figure 3-120 Enabling hybrid deployment of online and offline services



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 4 (Optional) Modify parameters for egress network bandwidth guarantee.

After confirming that the Volcano add-on is working, edit the parameter **configmap** of **volcano-agent-configuration** in the **kube-system** namespace. If

enable is set to **true** (default value), egress network bandwidth guarantee is enabled, and related parameters can be modified.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Example:

```
...
data:
  colocation-config: |
    {
        "globalConfig":{
            "cpuBurstConfig":{
                "enable":true
            },
            "networkQosConfig":{
                "enable":true,
                "onlineBandwidthWatermarkPercent":80,
                "offlineLowBandwidthPercent":10,
                "offlineHighBandwidthPercent":40
            },
...
```

NOTE

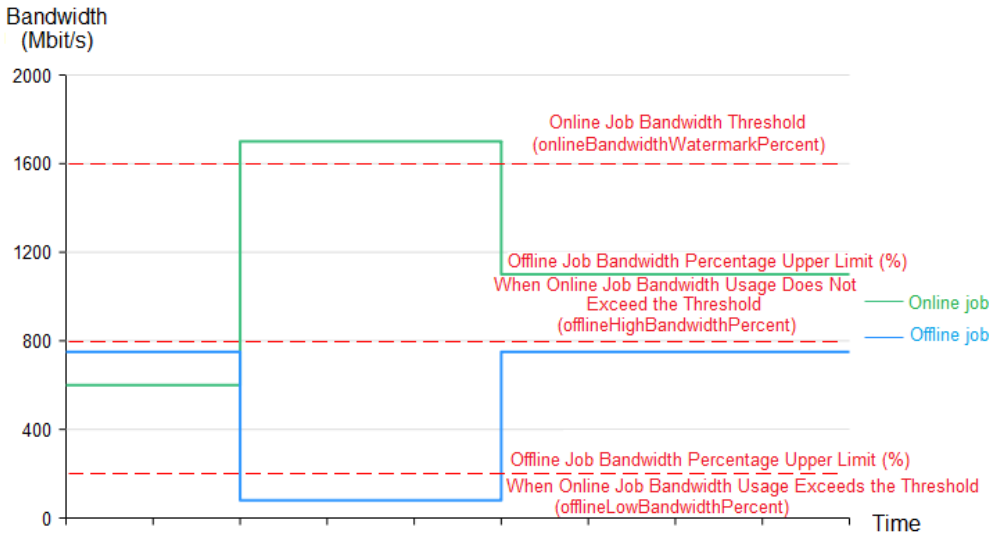
The modified parameters take effect for all nodes running Huawei Cloud EulerOS 2.0 in the cluster.

Table 3-137 networkQosConfig parameters

Name	Description	Default Value	Configuration Range
enable	Specifies whether to enable the egress network bandwidth guarantee feature.	true	true or false
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.

Name	Description	Default Value	Configuration Range
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	
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	

Figure 3-121 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.

Step 5 To disable egress network bandwidth guarantee, after confirming that the Volcano add-on is working, run the following command to edit the parameter **configmap** of **volcano-agent-configuration** in the namespace **kube-system**. Set **enable** to **false**.

```
kubectl edit configmap -nkube-system volcano-agent-configuration
```

Modify the following parameters:

```
...
data:
  colocation-config: |
    {
        "globalConfig":{
            "cpuBurstConfig":{
                "enable":true
            },
            "networkQosConfig":{
                "enable":false,
                "onlineBandwidthWatermarkPercent":80,
                "offlineLowBandwidthPercent":10,
                "offlineHighBandwidthPercent":40
            },
...
```

----End

3.7 Network

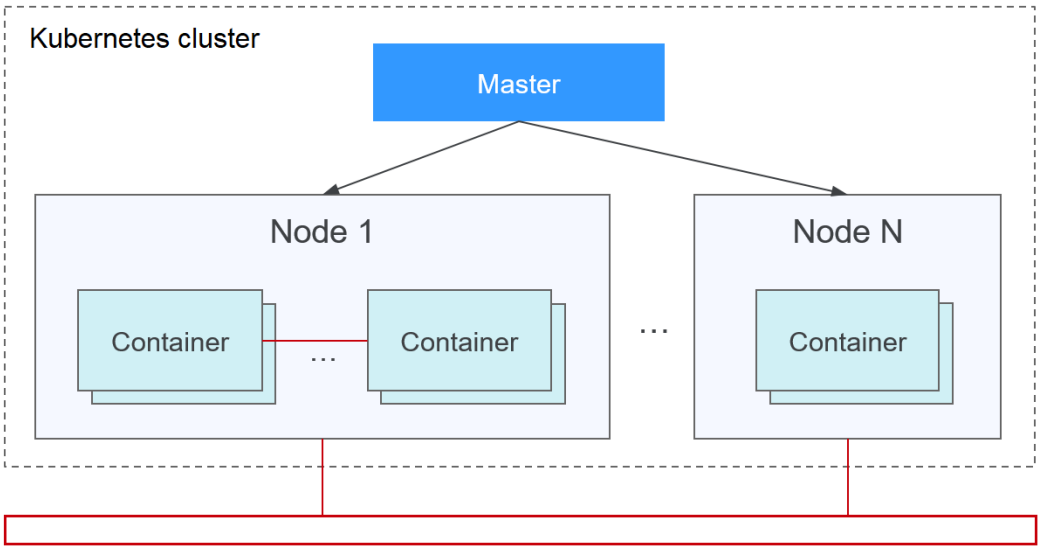
3.7.1 Overview

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 containers 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 uses VPC routing to integrate with the underlying network. This network model applies to performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. This networking model is free from tunnel encapsulation overhead and outperforms the container tunnel network model. 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 2.0 network 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 **3.7.2.1 Overview**.

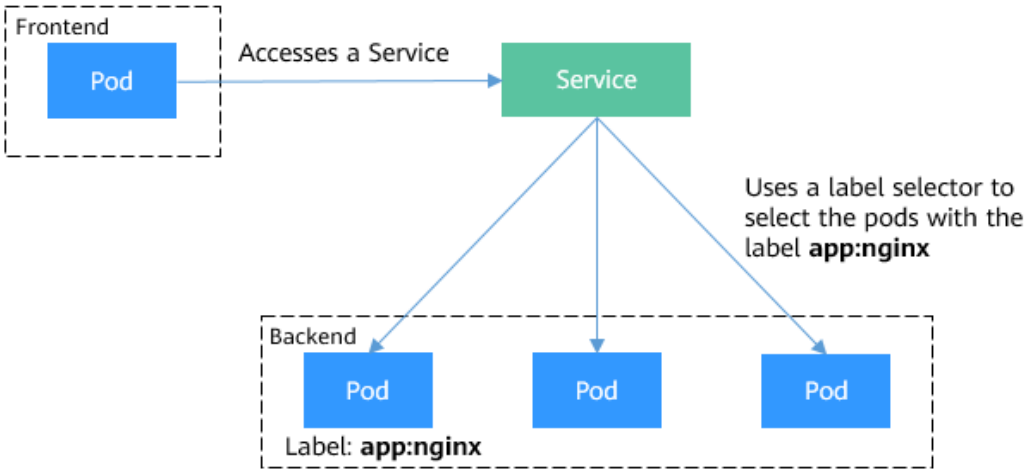
- **Service Network**

  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 3-122 Accessing pods through a Service



You can configure the following types of Services:

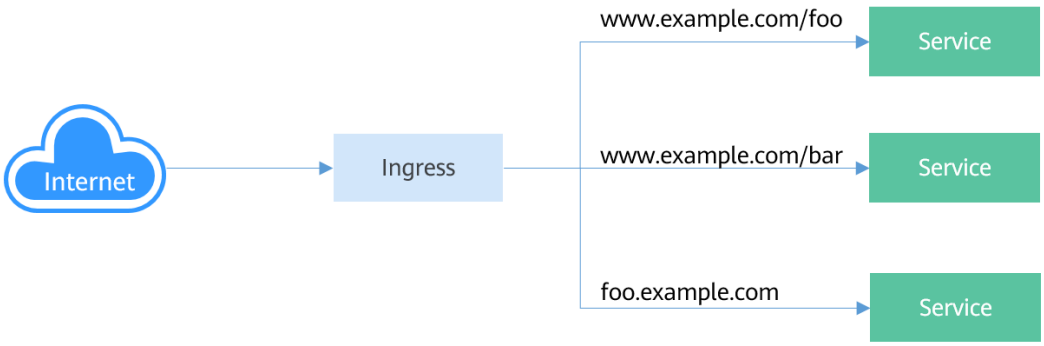
- ClusterIP: used to make the Service only reachable from within a cluster.
- NodePort: used for access from outside a cluster. A NodePort Service is accessed through the port on the node.
- LoadBalancer: used for access from outside a cluster. It is an extension of NodePort, to which a load balancer 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 **3.7.3.1 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 3-123 Ingress and Service



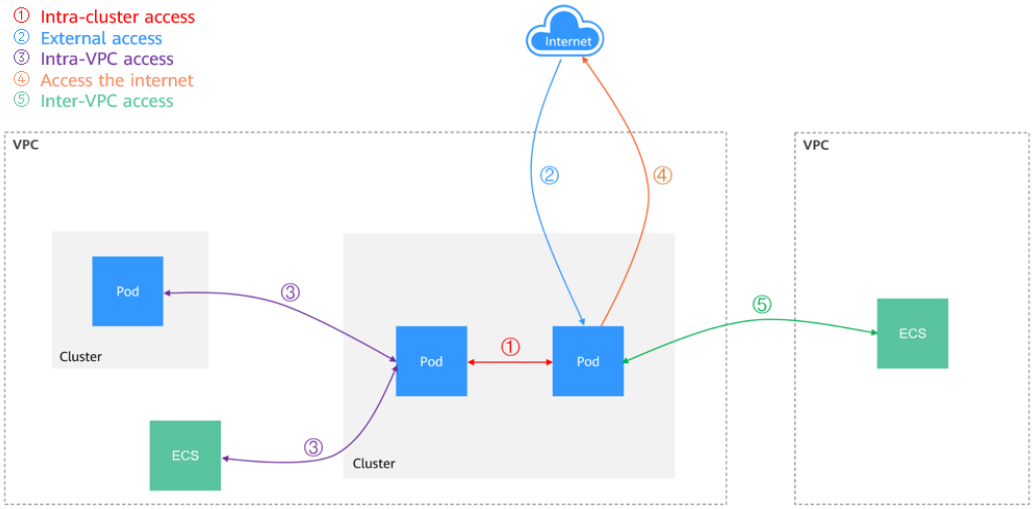
For details about the ingress, see **3.7.4.1 Overview**.

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 **3.7.8 Configuring Intra-VPC Access**.
  - Accessing a public network: Assign an EIP to the node where the workload runs (when the VPC network or tunnel network model is used), bind an EIP to the pod IP address (when the Cloud Native Network 2.0 model is used), or configure SNAT rules through the NAT gateway. For details, see **3.7.9 Accessing the Internet from a Container**.

Figure 3-124 Network access diagram



3.7.2 Container Network Models

3.7.2.1 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:

- **Tunnel network**
- **VPC network**
- **Cloud Native Network 2.0**

Network Model Comparison

Table 3-138 describes the differences of network models supported by CCE.

CAUTION

After a cluster is created, the network model cannot be changed.

Table 3-138 Network model comparison

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Application scenarios	• Common container service scenarios • Scenarios that do not have high requirements on network latency and bandwidth	• Scenarios that have high requirements on network latency and bandwidth • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.	• Scenarios that have high requirements on network latency, bandwidth, and performance • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.
Core technology	OVS	IPvlan and VPC route	VPC ENI/sub-ENI
Applicable clusters	CCE standard cluster	CCE standard cluster	CCE Turbo cluster
Network isolation	Kubernetes native NetworkPolicy for pods	No	Pods support security group isolation.

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Passthrough networking	No	No	Yes
IP address management	• The container CIDR block is allocated separately. • CIDR blocks are divided by node and can be dynamically allocated (CIDR blocks can be dynamically added after being allocated.)	• The container CIDR block is allocated separately. • CIDR blocks are divided by node and statically allocated (the CIDR block cannot be changed after a node is created).	The container CIDR block is divided from the VPC subnet and does not need to be allocated separately.
Network performance	Performance loss due to VXLAN encapsulation	No tunnel encapsulation. Cross-node packets are forwarded through VPC routers, delivering performance equivalent to that of the host network.	The container network is integrated with the VPC network, eliminating performance loss.

Dimension	Tunnel Network	VPC Network	Cloud Native Network 2.0
Networking scale	A maximum of 2000 nodes are supported.	Suitable for small- and medium-scale networks due to the limitation on VPC routing 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 routing tables (including the default and custom ones). Therefore, the cluster scale is limited by the VPC routing tables. For details about routing tables, see **Constraints**.	A maximum of 2000 nodes are supported.

NOTICE

1. The scale of a cluster that uses the VPC network model is limited by the custom routes of the VPC. Therefore, estimate the number of required nodes before creating a cluster.
2. The scale of a cluster that uses the Cloud Native Network 2.0 model depends on the size of the VPC subnet CIDR block selected for the network attachment definition. Before creating a cluster, evaluate the scale of your cluster.
3. By default, VPC routing network supports direct communication between containers and hosts in the same VPC. If a peering connection policy is configured between the VPC and another VPC, the containers can directly communicate with hosts on the peer VPC. In addition, in hybrid networking scenarios such as Direct Connect and VPN, communication between containers and hosts on the peer end can also be achieved with proper planning.
4. Do not change the mask of the primary CIDR block on the VPC after a cluster is created. Otherwise, the network will be abnormal.

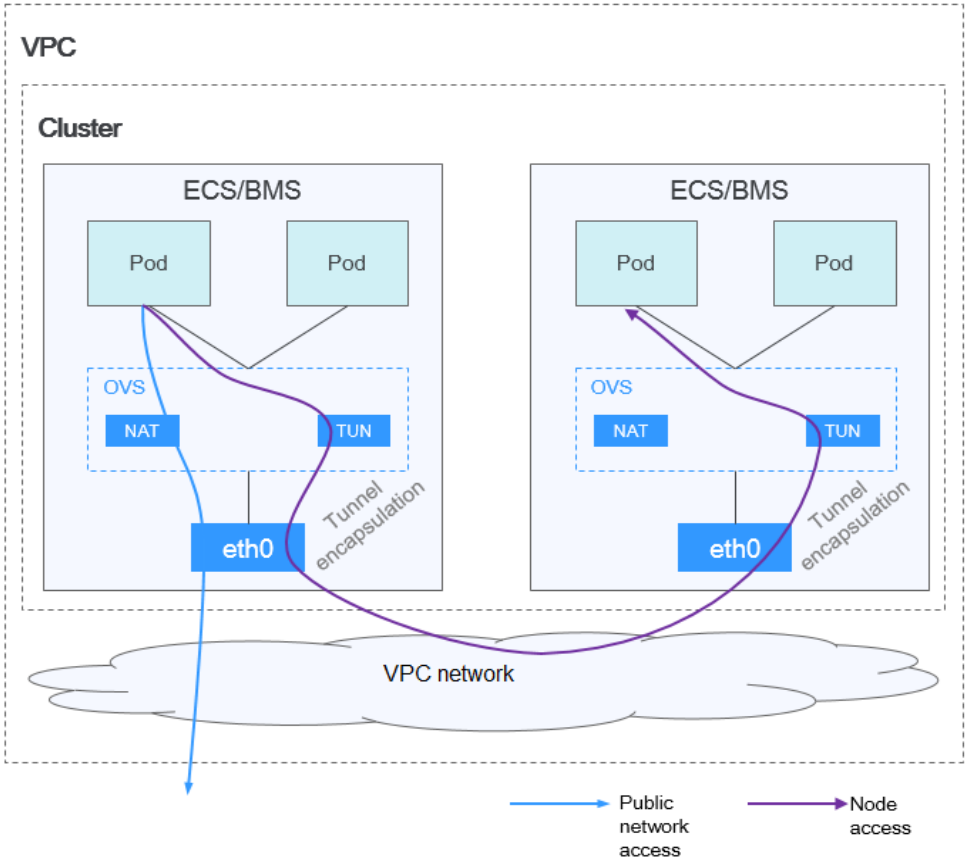
3.7.2.2 Container Tunnel Network

Container Tunnel Network Model

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. Though at some costs of performance, packet encapsulation and tunnel transmission enable higher interoperability and compatibility with advanced features (such as network policy-based isolation) for most common scenarios.

Figure 3-125 Container tunnel network



Pod-to-pod communication

- On the same node: Packets are directly forwarded via the OVS bridge on the node.
- Across nodes: Packets are encapsulated in the OVS bridge and then forwarded to the peer node.

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 elastic ENIs, and creation speed).
- Network isolation is supported. For details, see **3.7.6.3.1 Network Policies**.
- Bandwidth limits are supported.
- Large-scale networking is supported.

Disadvantages

- High encapsulation overhead, complex networking, and low performance
- Pods cannot directly use functionalities such as EIPs and security groups.
- External networks cannot be directly connected to container IP addresses.

Applicable Scenarios

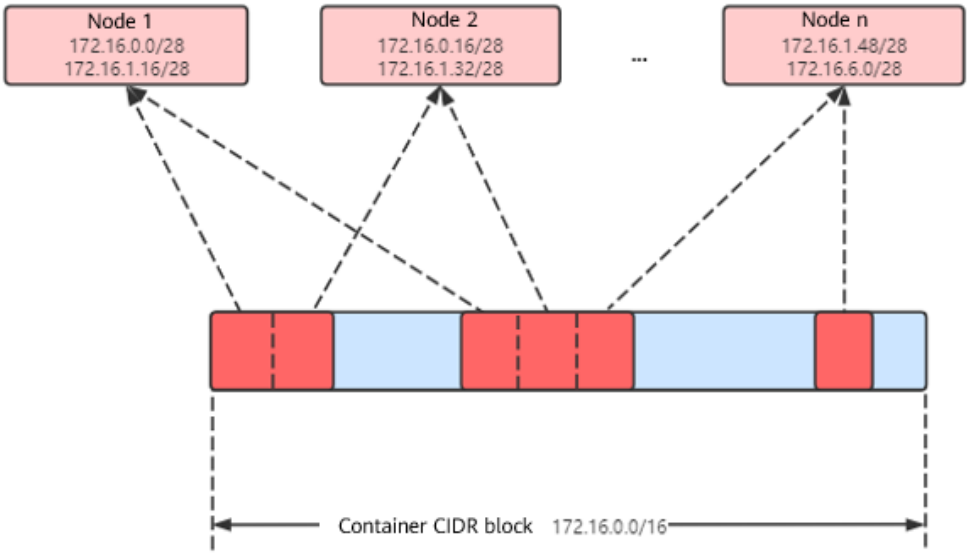
- Low requirements on performance: As the container tunnel network requires additional VXLAN tunnel encapsulation, it has about 5% to 15% of performance loss when compared with the other two container network models. Therefore, the container tunnel network applies to the scenarios that do not have high 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:

- The container CIDR block is allocated separately, which is irrelevant to the node CIDR block.
- 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.
- The container CIDR block cyclically allocates CIDR blocks to new nodes or existing nodes in sequence.
- Pods scheduled to a node are cyclically allocated IP addresses from one or more CIDR blocks allocated to the node.

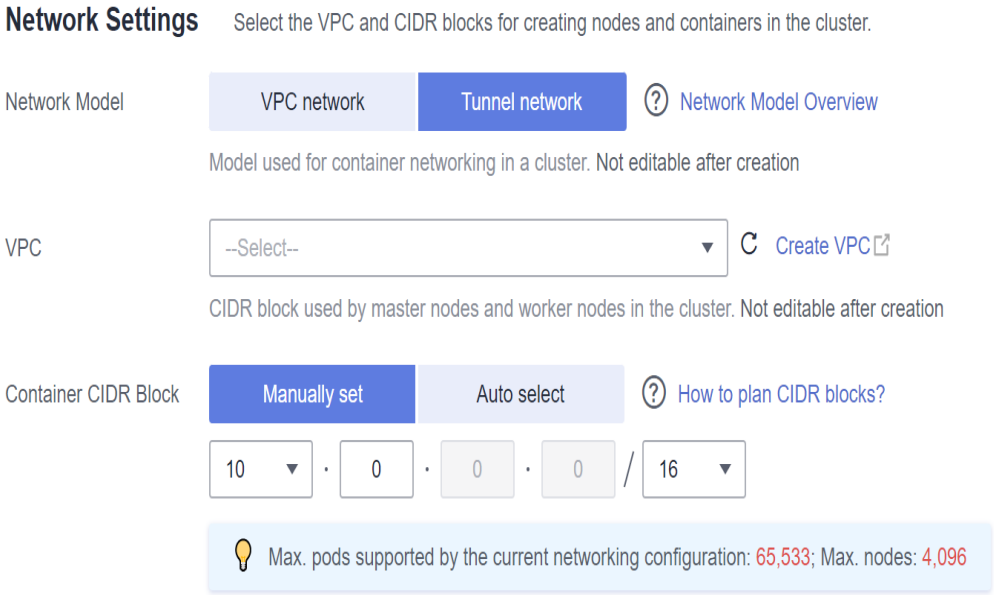
Figure 3-126 IP address allocation of the container tunnel network



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. If 16 IP addresses are allocated to a node at a time, a maximum of 4096 (65536/16) nodes can be created in the cluster. This is an extreme case. If 4096 nodes are created, a maximum of 16 pods can be created for each node because only 16 IP CIDR block\s are allocated to each node. In addition, the number of nodes that can be created in a cluster also depends on the node network and cluster scale.

Figure 3-127 Selecting a network model (when creating the cluster)



Recommendation for CIDR Block Planning

As described in **Cluster Network Structure**, network addresses in a cluster can be divided into three parts: node network, container network, and service network. When planning network addresses, consider the following aspects:

- The three CIDR blocks cannot overlap. Otherwise, a conflict occurs. All subnets (including those created from the secondary CIDR block) 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 node 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 **3.3.8.3 Maximum Number of Pods That Can Be Created on a Node**.

Example of Container Tunnel Network Access

Create a cluster that uses the container tunnel network model. Create a Deployment in the cluster.

```
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
```

View the created pod.

```
$ kubectl get pod -owide
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>
```

In this case, the IP address of the pod cannot be directly accessed outside the cluster in the same VPC. This is a feature of the container tunnel network.

However, the pod can be accessed from a node in the cluster or in the pod. As shown in the following figure, the pod can be accessed directly from the container.

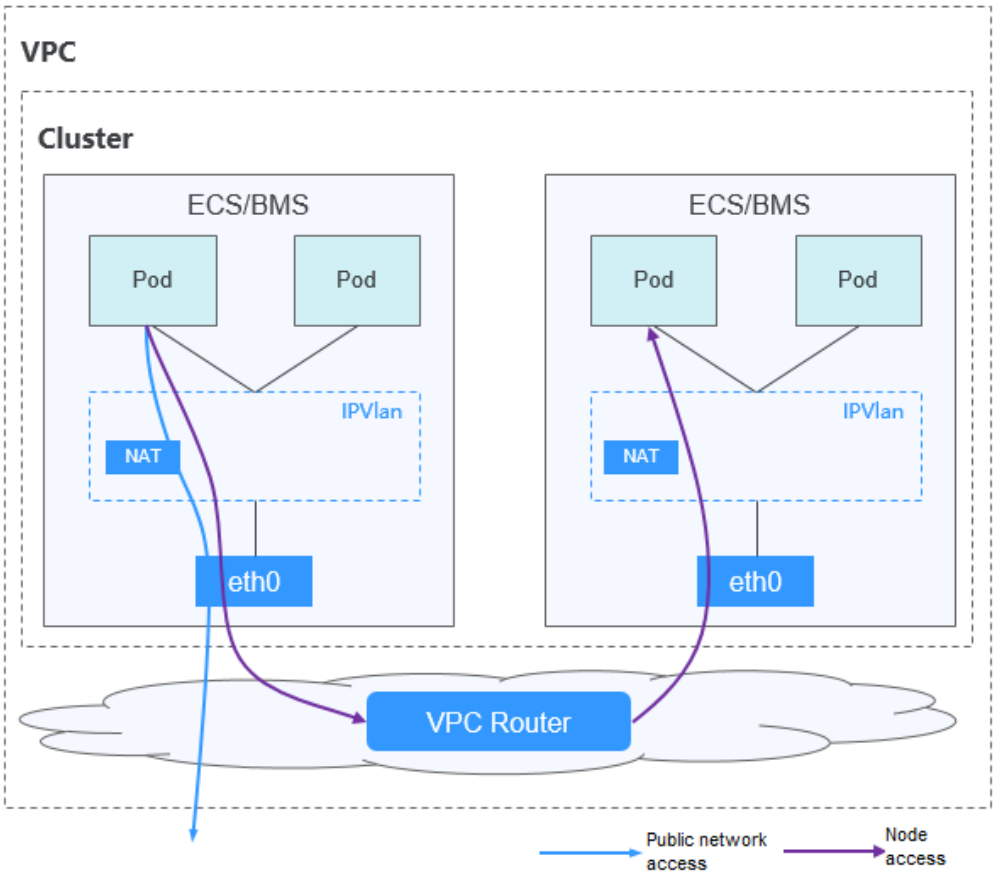
```
$ kubectl exec -it example-5bdc5699b7-5rvq4 -- curl 10.0.0.21
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</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>
```

3.7.2.3 VPC Network

Model Definition

The VPC network uses VPC routing to integrate with the underlying network. This network model is suitable for performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the VPC route quota. Each node is assigned a CIDR block of a fixed size. This networking model is free from tunnel encapsulation overhead and outperforms the container tunnel network model. 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 ECSs in the same VPC outside the cluster.

Figure 3-128 VPC network model



Pod-to-pod communication

- On the same node: Packets are directly forwarded through IPvlan.
- Across nodes: Packets are forwarded to the default gateway through default routes, and then to the peer node via the VPC routes.

Advantages and Disadvantages

Advantages

- No tunnel encapsulation is required, so network problems are easy to locate and the performance is high.
- In the same VPC, the external network of the cluster can be directly connected to the container IP address.

Disadvantages

- The number of nodes is limited by the VPC route quota.
- Each node is assigned a CIDR block of a fixed size, which leads to a waste of IP addresses in the container CIDR block.
- Pods cannot directly use functionalities such as EIPs and security groups.

Applicable Scenarios

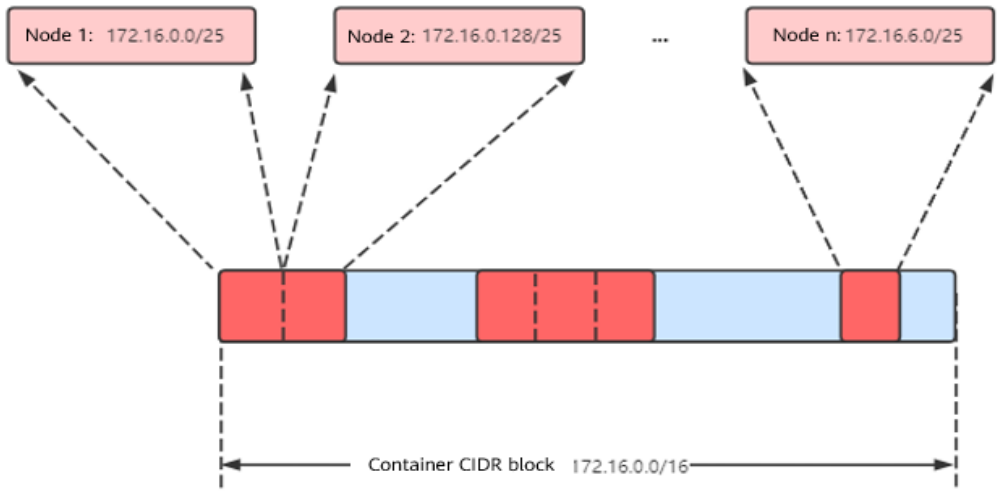
- High performance requirements: As no tunnel encapsulation is required, the VPC network model delivers the performance close to that of a VPC network when compared with the container tunnel network model. Therefore, the VPC network model applies to scenarios that have high requirements on performance, such as AI computing and big data computing.
- Small- and medium-scale networks: Due to the limitation on VPC routing 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 allocates container IP addresses according to the following rules:

- The container CIDR block is allocated separately.
- IP addresses are allocated by node. One CIDR block with a fixed size (which is configurable) is allocated to each node in a cluster from the container CIDR block.
- The container CIDR block cyclically allocates CIDR blocks to new nodes in sequence.
- Pods scheduled to a node are cyclically allocated IP addresses from CIDR blocks allocated to the node.

Figure 3-129 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 the 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. In addition, the number of nodes that can be created in a cluster also depends on the node network and cluster scale.

Recommendation for CIDR Block Planning

As described in **Cluster Network Structure**, network addresses in a cluster can be divided into three parts: node network, container network, and service network. When planning network addresses, consider the following aspects:

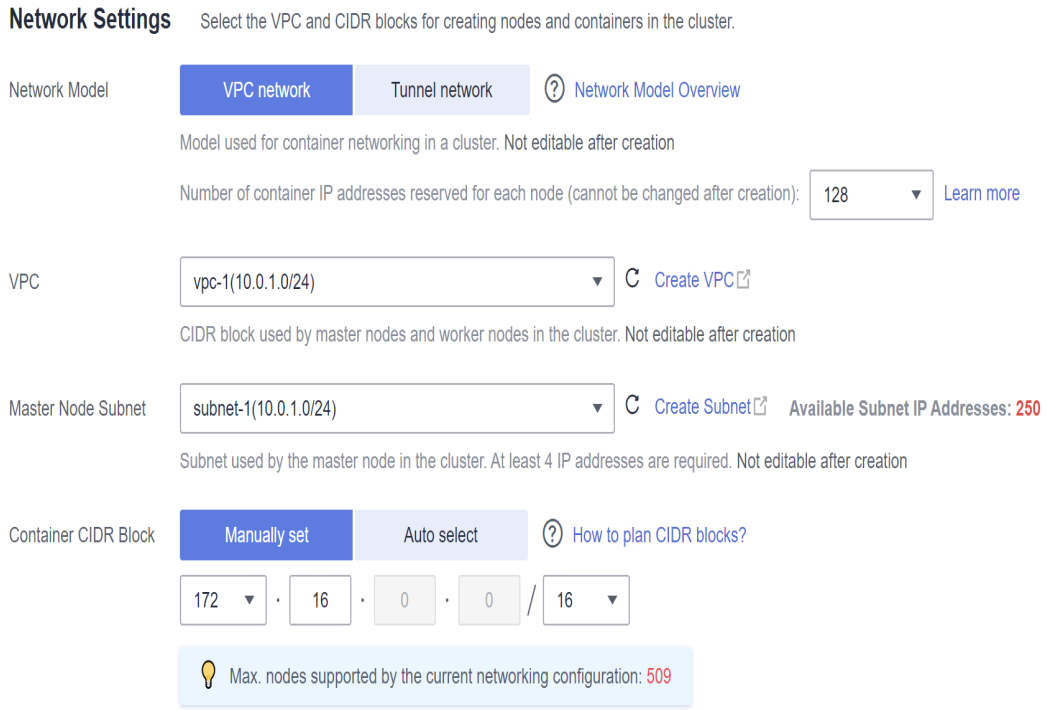
- The three CIDR blocks cannot overlap. Otherwise, a conflict occurs. All subnets (including those created from the secondary CIDR block) in the VPC where the cluster resides cannot conflict with the container and Service CIDR blocks.
- Ensure that each CIDR block has sufficient IP addresses.
  - The IP addresses in the node 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 **3.3.8.3 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 node subnet must be greater than 200. Otherwise, nodes cannot be created due to insufficient IP addresses.

The container CIDR block is 10.0.0.0/16, 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 the 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, including the three master nodes.

Figure 3-130 Setting the container CIDR block (when creating the cluster)



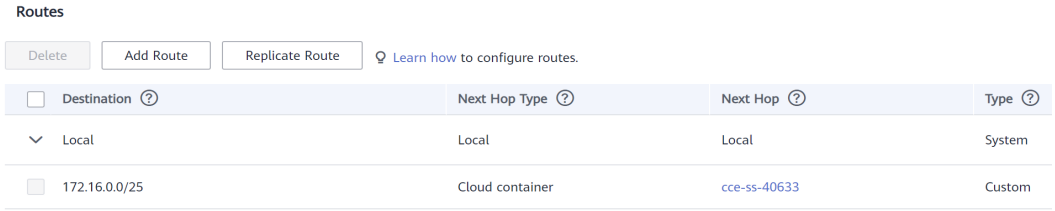
Example of VPC Network Access

Create a cluster using the VPC network model. The cluster contains one node.

```
$ kubectl get node
NAME          STATUS   ROLES    AGE   VERSION
192.168.0.99  Ready    <none>   9d    v1.17.17-r0-CCE21.6.1.B004-17.37.5
```

Check the VPC routing table. The destination address 172.16.0.0/25 is the container CIDR block allocated to the node, and the next hop is the corresponding node. When the container IP address is accessed, the VPC route forwards the access request to the next-hop node. This indicates that the VPC network model uses VPC routes.

Figure 3-131 Routes



Create a Deployment in the cluster.

```
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
```

Check the pod.

```
$ kubectl get pod -owide
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>
```

In this case, if you access the IP address of the pod from an ECS (outside the cluster) in the same VPC, the access is successful. This is a feature of VPC

networking. Pods can be directly accessed from any node locating outside of the cluster and in the same VPC as that of the pods using the pods' IP addresses.

Pods can be accessed from nodes or pods in the same cluster. In the following example, you can directly access the pods in the container.

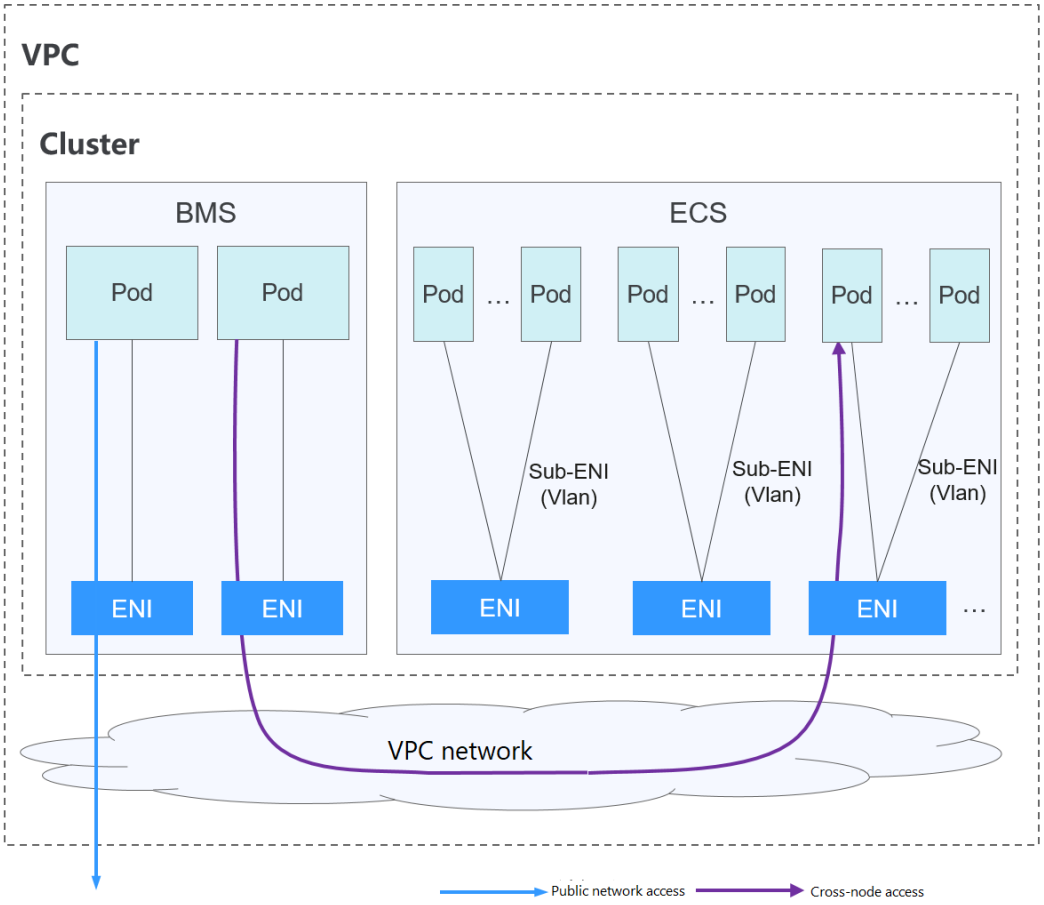
```
$ kubectl exec -it example-86b9779494-l8qrw -- curl 172.16.0.7
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</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>
```

3.7.2.4 Cloud Native 2.0 Network

Model Definition

Developed by CCE, Cloud Native 2.0 network deeply integrates Elastic Network Interfaces (ENIs) and sub-ENIs of Virtual Private Cloud (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.

Figure 3-132 Cloud Native 2.0 network



Pod-to-pod communication

- Pods on BMS nodes use ENIs, whereas pods on ECS nodes use Sub-ENIs. Sub-ENIs are attached to ENIs through VLAN sub-interfaces.
- On the same node: Packets are forwarded through the VPC ENI or sub-ENI.
- Across nodes: Packets are forwarded through the VPC ENI or sub-ENI.

Constraints

This network model is available only to CCE Turbo clusters.

Advantages and Disadvantages

Advantages

- As the container network directly uses VPC, it is easy to locate network problems and provide the highest performance.
- External networks in a VPC can be directly connected to container IP addresses.
- The load balancing, security group, and EIP capabilities provided by VPC can be directly used by pods.

Disadvantages

The container network directly uses VPC, which occupies the VPC address space. Therefore, you must properly plan the container CIDR block before creating a cluster.

Application Scenarios

- High performance requirements and use of other VPC network capabilities: Cloud Native Network 2.0 directly uses VPC, which delivers almost the same performance as the VPC network. Therefore, it applies to scenarios that have high requirements on bandwidth and latency, such as live streaming and e-commerce flash sale.
- Large-scale networking: Cloud Native Network 2.0 supports a maximum of 2000 ECS nodes and 100,000 containers.

Container IP Address Management

In the Cloud Native Network 2.0 model, 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.

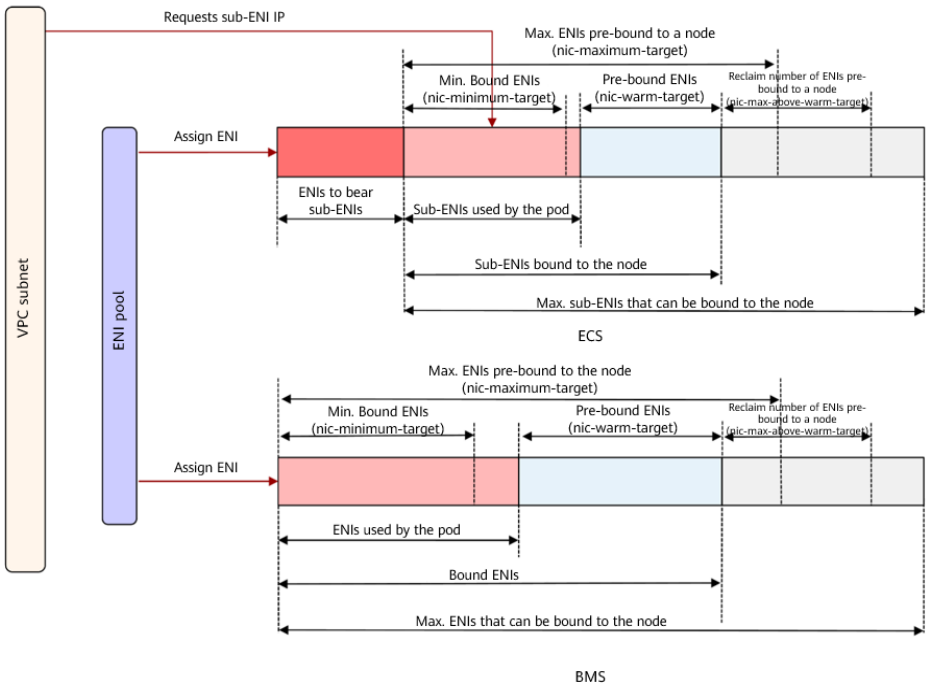
Table 3-139 Comparison between ENI pre-binding policies

Policy	Dynamic ENI Pre-binding Policy (Default)	Threshold-based ENI Pre-binding Policy
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 3-133 Dynamic ENI pre-binding policy



CCE provides four parameters for the dynamic ENI pre-binding policy. Set these parameters properly.

Table 3-140 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)**

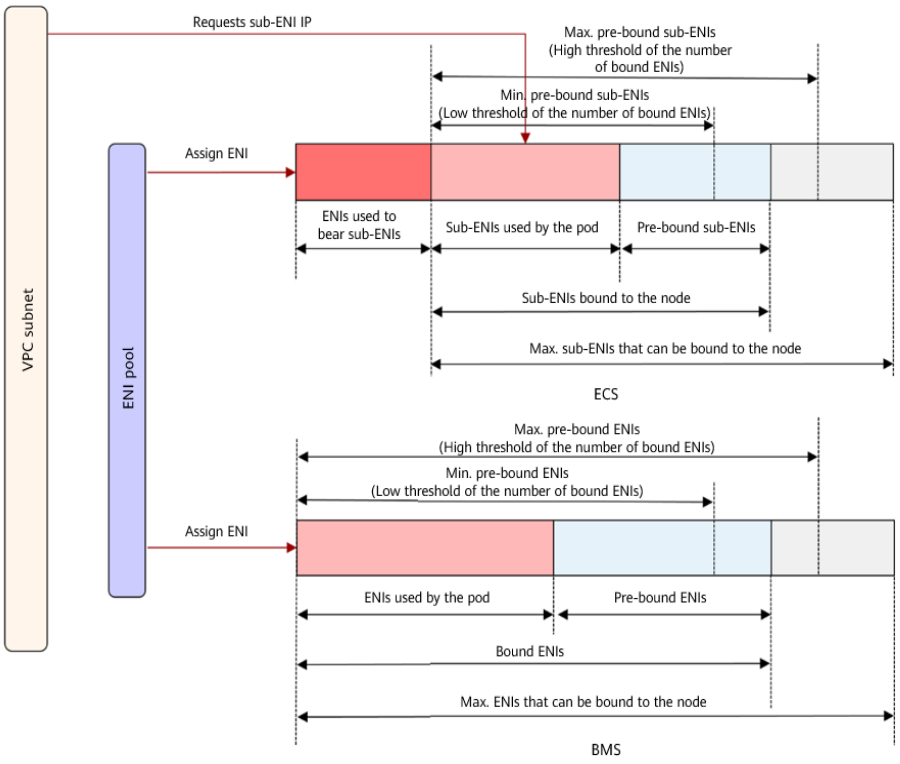
The number of pre-binding ENIs on the node remains in the following range:

- **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 ENI (the earliest unused one) is preferentially allocated from the pool. If no idle ENI is available, a newsub-ENI is 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 3-134 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 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

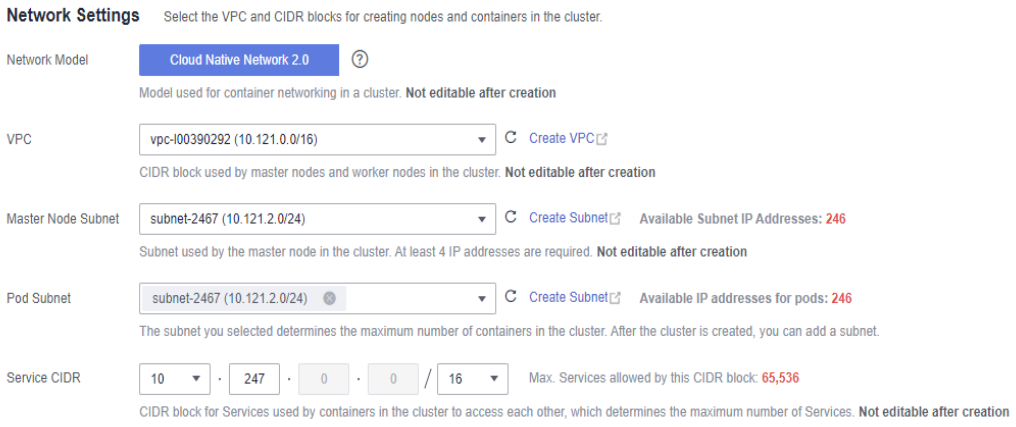
As described in **Cluster Network Structure**, network addresses in a cluster can be divided into three parts: node network, container network, and service network. When planning network addresses, consider the following aspects:

- The three CIDR blocks cannot overlap. Otherwise, a conflict occurs. All subnets (including those created from the secondary CIDR block) 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 node 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 the Cloud Native Network 2.0 model, the container CIDR block and node CIDR block share the network 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 resources.

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 3-135 Setting the CIDR blocks (when creating the cluster)

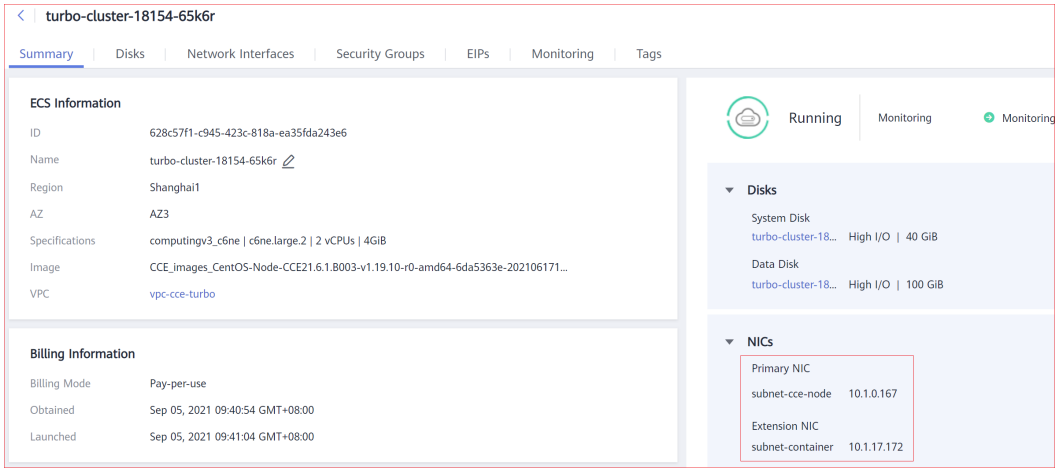


Example of Cloud Native Network 2.0 Access

Create a CCE Turbo cluster, which contains three ECS nodes.

Access the details page of one node. You can see that the node has one primary ENI and one extended ENI, and both of them are ENIs. The extended ENI belongs to the container CIDR block and is used to mount a sub-ENI to the pod.

Figure 3-136 Node ENIs



Create a Deployment in the cluster.

```
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
```

View the created pod.

```
$ kubectl get pod -owide
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 mounted to the ENI (extended ENI) of the node.

For example, the extended ENI of node 10.1.0.167 is 10.1.17.172. On the **Network Interfaces** page of the Network Console, you can see that three sub-ENIs are mounted to the extended ENI 10.1.17.172, which is the IP address of the pod.

Figure 3-137 Pod ENIs

In the VPC, the IP address of the pod can be successfully accessed.

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 3-141 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**).
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.

3.7.3 Service

3.7.3.1 Overview

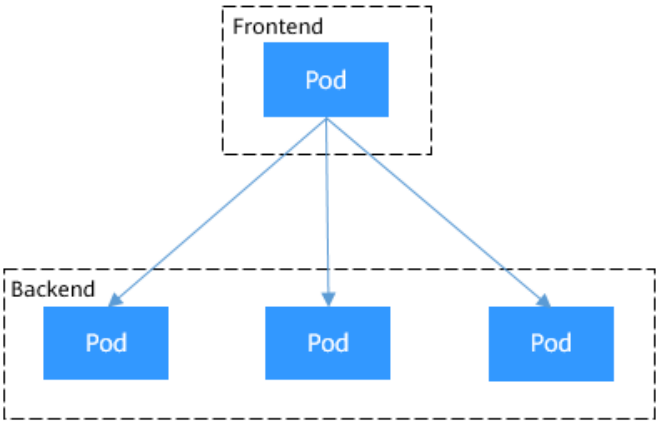
Direct Access to a Pod

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 3-138**. 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 3-138 Inter-pod access

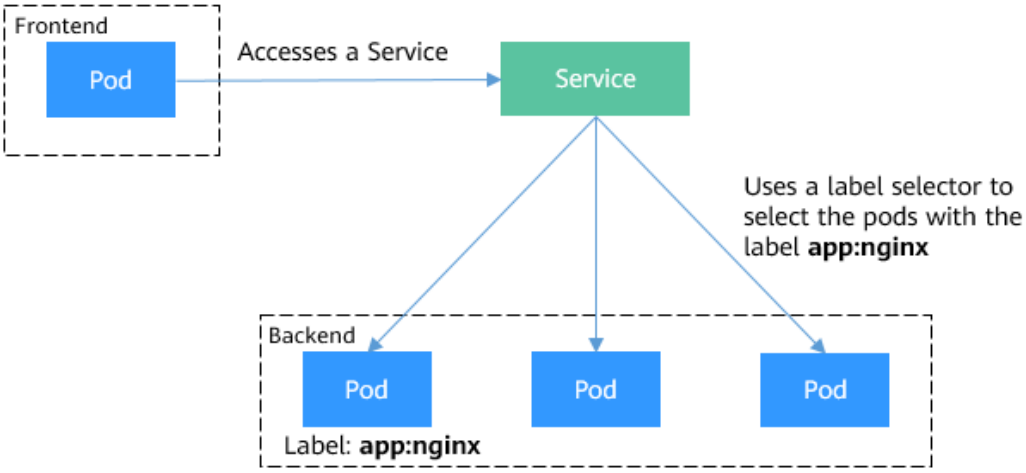


Using Services for Pod Access

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 3-139**.

Figure 3-139 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 insides 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:

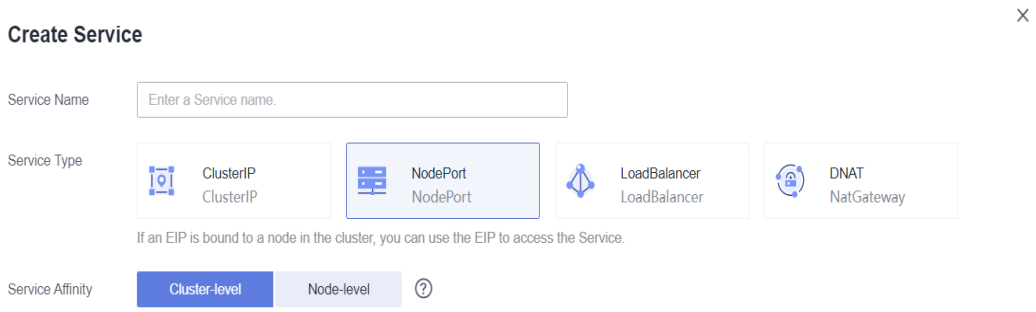
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
```

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 3-142 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 need 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 a Cloud Native 2.0 network model, 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)
nginx-ingress add-on connected with a dedicated load balancer (**Local**)	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 **3.7.3.4.13 Enabling Passthrough Networking for LoadBalancer Services**.

NOTE

- After passthrough networking is configured for a dedicated load balancer, in a CCE standard cluster, pods that run on the same node as the workload and pods that run on the same node cannot be accessed through the LoadBalancer Service.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the pass-through settings of Service connected to the same ELB 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
```

3.7.3.2 ClusterIP

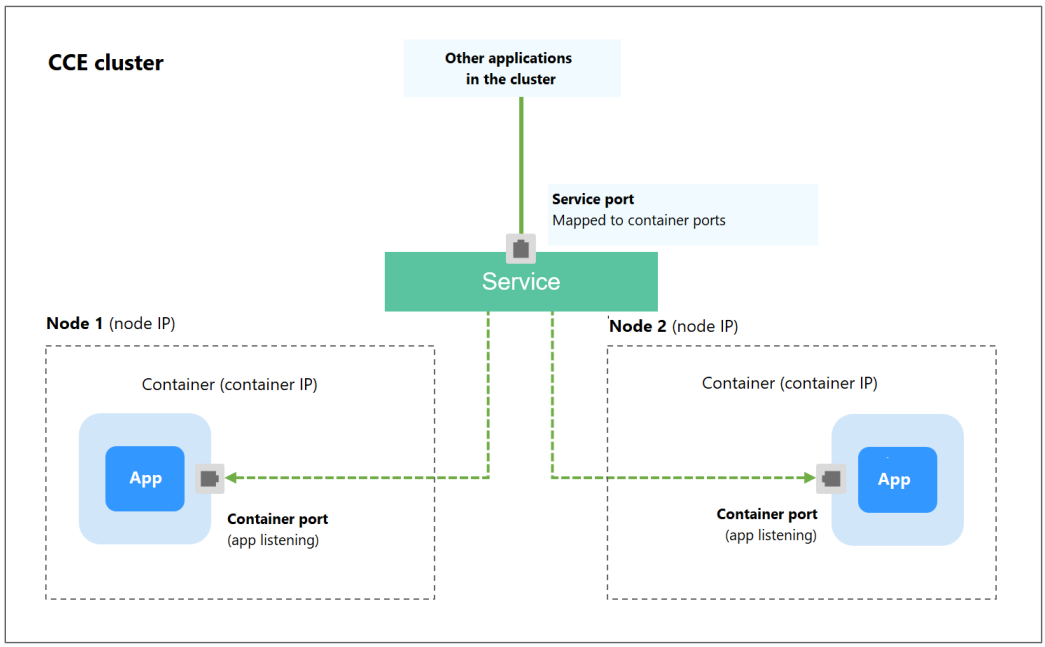
Scenario

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 3-140 shows the mapping relationships between access channels, container ports, and access ports.

Figure 3-140 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 to which the workload belongs.
- **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.

Step 4 Click **OK**.

----End

Setting the Access Type Using kubectl

You can run kubectl commands to configure Service access. This section uses an Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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>
...
```

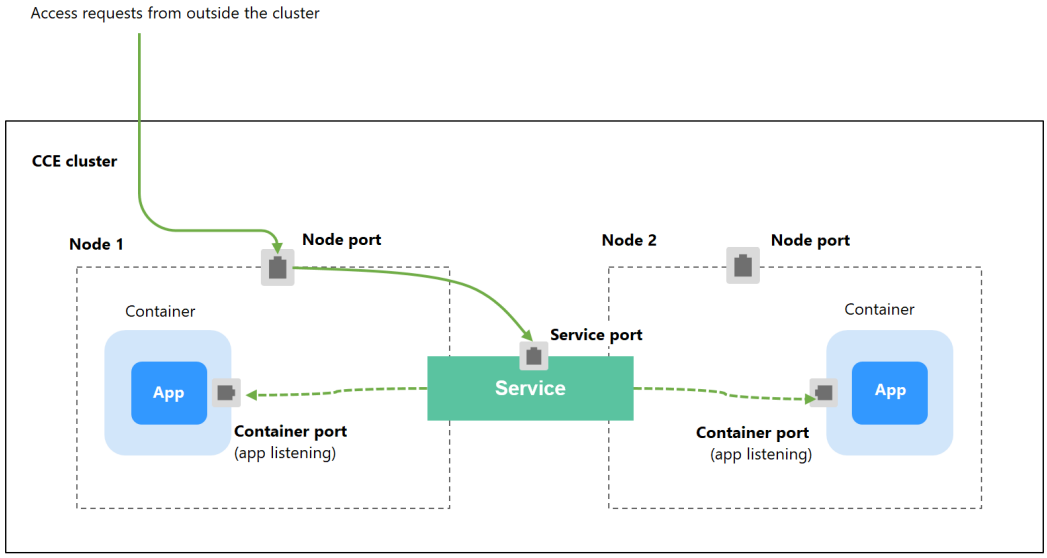
----End

3.7.3.3 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 3-141 NodePort access



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 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**.
- **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 run kubectl commands to configure Service access. This section uses an Nginx workload as an example to describe how to set a NodePort Service using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

3.7.3.4 LoadBalancer

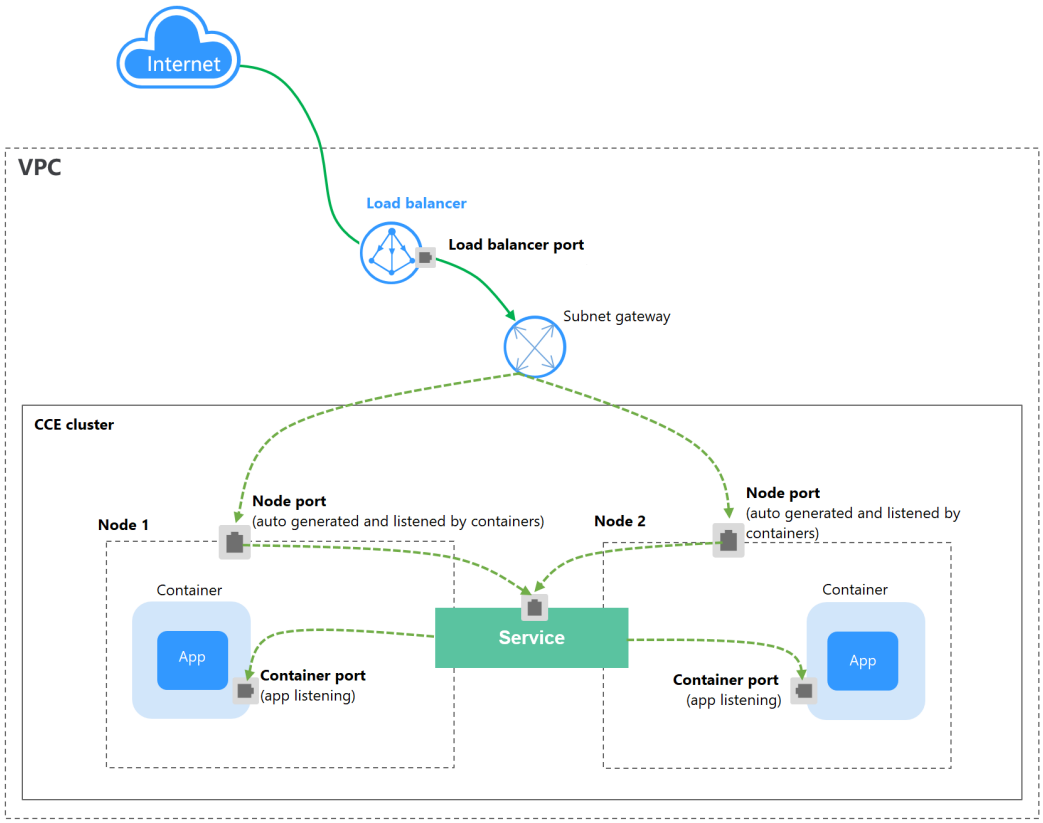
3.7.3.4.1 Creating a LoadBalancer Service

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.

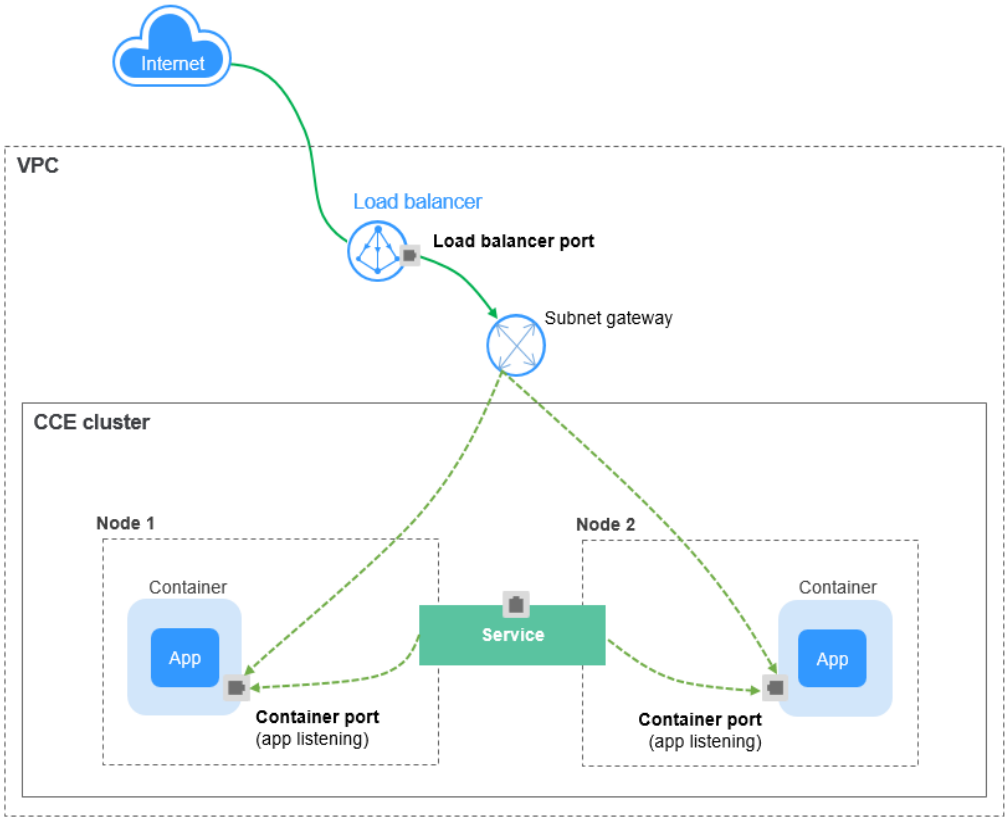
Figure 3-142 LoadBalancer



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 3-143 Passthrough networking



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 the 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 a Cloud Native 2.0 network model, 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.
- A dedicated load balancer must be of the network type (TCP/UDP) and support private networks (with a private IP address). If the Service needs to support HTTP, the dedicated load balancers must be of the network (TCP/UDP) or application load balancing (HTTP/HTTPS) type.

- In a CCE cluster, if the cluster-level affinity is configured for a LoadBalancer Service, requests are distributed to the node ports of each node using SNAT when entering the cluster. The number of node ports cannot exceed the number of available node ports on the node. If the service affinity is at the node level (Local), there is no such constraint. In a CCE Turbo cluster, this constraint applies to shared load balancers, but not dedicated ones. Use dedicated load balancers in CCE Turbo clusters.
- 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.
- In a cluster using the IPVS proxy mode, if the ingress and Service use the same ELB load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer connected to the ingress. Use different load balancers for the ingress and Service.

Creating a LoadBalancer 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 parameters.

- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **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**.
- **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).**
- **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/TLS)**, **Application (HTTP/HTTPS)**, or **Network (TCP/UDP/TLS) & 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 3-143**.

Table 3-143 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/Application-oriented 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. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency.

You can click  in the **Set ELB** area and configure load balancer parameters in the **Set ELB** dialog box.

– **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.

  - **Type**: This function is disabled by default. You can select **Source IP address**. Source IP address-based sticky session means that access requests from the same IP address are forwarded to the same backend server.

    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 **3.7.3.4.11 Configuring Health Check on Multiple Service Ports**.

Table 3-144 Health check parameters

Parameter	Description
Protocol	When the protocol of **Port** is set to TCP, the TCP and HTTP are supported. When the protocol of **Port** is set to UDP, the UDP 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.
  - **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**: 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 3-144**.

    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.
    - **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.
- **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**.
- **Access Control**
  - **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.
- **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.

Configuration	Description	Restrictions
Rewrite X-Forwarded-Host	If this function is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the client request body 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.
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.

Configuration	Description	Restrictions
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 **3.7.3.4.2 Using Annotations to Balance Load**.

Step 4 Click **OK**.

----End

Using kubectl to Create a Service (Using an Existing Load Balancer)

You can configure Service access when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

Before enabling sticky session, ensure that the following conditions are met:

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

```
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 (min)
    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 3-145**.

For more annotations and examples related to advanced functions, see **3.7.3.4.2 Using Annotations to Balance Load**.

Table 3-145 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	ID of an enhanced 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**.
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: • **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.

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. • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter 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 3-146** object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave blank this parameter 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 at the same time.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.health-check-option	No	Table 3-147 object	ELB health check configuration items.

Table 3-146 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 3-147 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

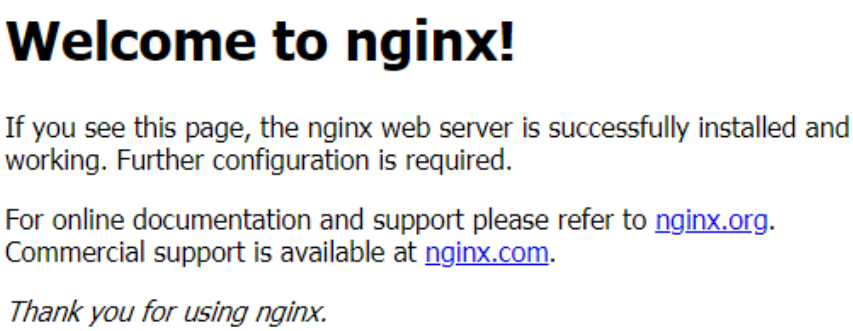
If information similar to the following is displayed, the access type has been set, and the workload is accessible.

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.247.0.1      <none>         443/TCP        3d
nginx        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 3-144 Accessing Nginx through the LoadBalancer Service



----End

Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can configure Service access when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

Before enabling sticky session, ensure that the following conditions are met:

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes. For details, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

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 (min)
    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 (min)
    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 3-148**.

For more annotations and examples related to advanced functions, see **3.7.3.4.2 Using Annotations to Balance Load**.

Table 3-148 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: • **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**.
kubernetes.io/elb.autocreate	Yes	**elb.autocreate** object	Whether to automatically create a load balancer associated with the Service. **Example** • If a public network load balancer will be automatically created, set this parameter to the following value: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • If a private network load balancer will be automatically created, set this parameter to the following value: {"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 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?**

Parameter	Mandatory	Type	Description
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.
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. • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter 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.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.session-affinity-option	No	Table 3-146 object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave blank this parameter 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 at the same time.
kubernetes.io/elb.health-check-option	No	Table 3-147 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. A tag is in the format of "key=value". Use commas (,) to separate multiple tags.

Table 3-149 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**

Parameter	Mandatory	Type	Description
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.
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.

Parameter	Mandatory	Type	Description
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 **querying 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 **querying 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 **querying 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.

Parameter	Mandatory	Type	Description
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	Specifies the 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

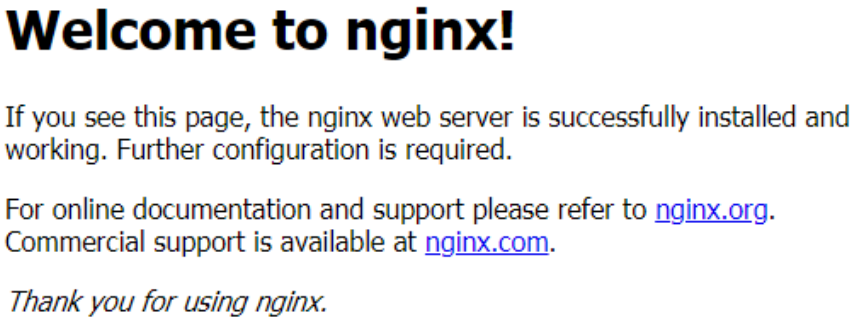
If information similar to the following is displayed, the access type has been set, and the workload is accessible.

```
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.247.0.1       <none>         443/TCP        3d
nginx        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 3-145 Accessing Nginx through the LoadBalancer Service



----End

3.7.3.4.2 Using Annotations to Balance Load

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**

Interconnection with ELB

Table 3-150 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. The value can be: • **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 3-163	Mandatory **when load balancers are automatically created**. **Example:** • If a public network load balancer will be automatically created, set this parameter to the following value: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • If a private network load balancer will be automatically created, set this parameter to the following value: {"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 later than v1.11.7-r0.	Mandatory for clusters earlier than v1.11.7-r0 Discarded in clusters later than v1.11.7-r0
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 3-151 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. • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter 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 3-166	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 (min)
spec:
  selector:
    app: nginx
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```


Health Check

Table 3-152 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 blank this parameter 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 at the same time.	v1.9 or later
kubernetes.io/elb.health-check-option	**Table 3-164**	ELB health check configuration items.	v1.9 or later
kubernetes.io/elb.health-check-options	**Table 3-165**	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 **3.7.3.4.11 Configuring Health Check on Multiple Service Ports**.

HTTP or HTTPS

Table 3-153 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, see **3.7.3.4.3 Configuring an HTTP or HTTPS Service**.

SNI

Table 3-154 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 **3.7.3.4.4 Configuring SNI for a Service**.

Dynamic Adjustment of the Weight of the Backend ECS

Table 3-155 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 This parameter applies only to clusters of v1.21 or later and is invalid in passthrough networking.	v1.21 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.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 3-156 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, see **3.7.3.4.13 Enabling Passthrough Networking for LoadBalancer Services**.

Blocklist/Trustlist

Table 3-157 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**. **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
```

Host Network

Table 3-158 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 3-159 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, see **3.7.3.4.7 Configuring Timeout for a Service**.

Resource Tags

Table 3-160 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, see **Using kubectl to Create a Service (Automatically Creating a Load Balancer)**.

HTTP/2

Table 3-161 Annotations of 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, see **3.7.3.4.5 Configuring HTTP/2 for a Service**.

Enabling GZIP

Table 3-162 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: all file formats • GZIP: 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, see **3.7.3.4.9 Configuring GZIP Data Compression for a Service**.

Parameters for Automatically Creating a Load Balancer

Table 3-163 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**

Parameter	Mandatory	Type	Description
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.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. • **PER**: dedicated bandwidth

Parameter	Mandatory	Type	Description
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 **querying 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 **querying 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 **querying 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	Specifies the 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 3-164 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 3-165 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 3-166 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**

3.7.3.4.3 Configuring an HTTP or HTTPS Service

Constraints

- Only clusters of v1.19.16 or later support HTTP or HTTPS.

Table 3-167 Scenarios where a load balancer supports HTTP or HTTPS

ELB Type	Application scenario	Whether to Support HTTP or HTTPS	Description
Shared load balancer	Interconnecting with an existing load balancer	Yes	None
	Automatically creating a load balancer	Yes	None
Dedicated load balancer	Interconnecting with an existing load balancer	Yes (A YAML file is required.)	• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10 and 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 scenario	Whether to Support HTTP or HTTPS	Description
	Automatically creating a load balancer	Yes (A YAML file is required.)	• For versions earlier than v1.19.16-r50, v1.21.11-r10, v1.23.9-r10, v1.25.4-r10 and 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 **Creating a LoadBalancer Service**.

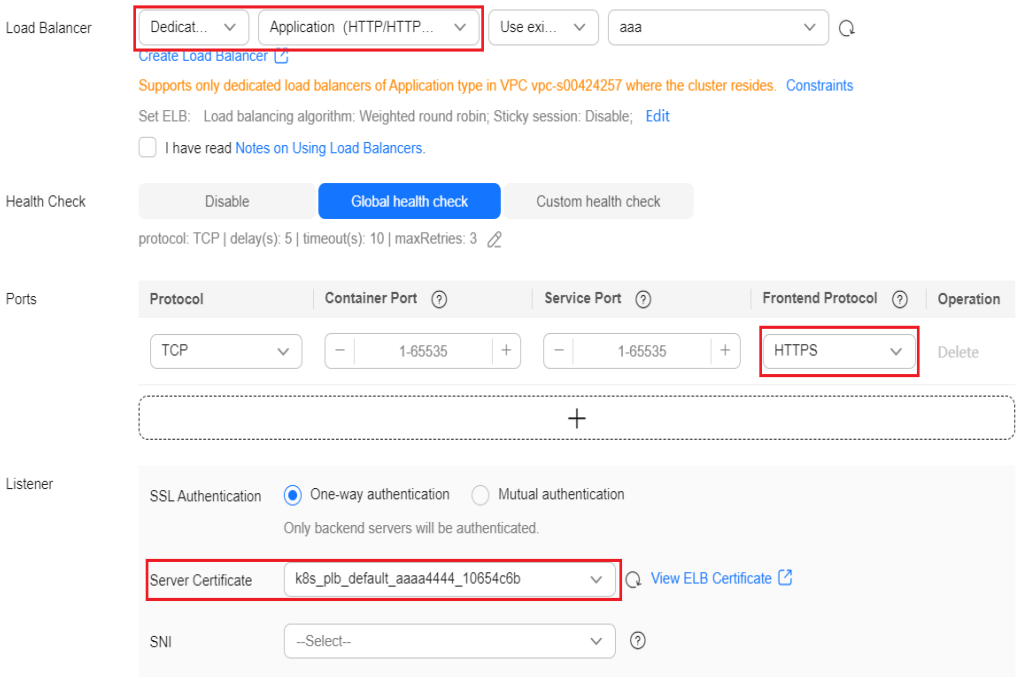
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an 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 3-143**.
- **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.
    - **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**.

    NOTE

    If multiple HTTPS Services are released, all listeners will use the same certificate configuration.

Figure 3-146 HTTP/HTTPS



Step 4 Click **OK**.

----End

Using kubectl

If a Service is HTTP/HTTPS-compliant, add the following annotations:

- **kubernetes.io/elb.protocol-port**: "https:443,http:80"

  The value of **protocol-port** must be the same as the port in the **spec.ports** field of the Service. The format is *Protocol:Port*. The port matches the one in the **service.spec.ports** field and is released as the corresponding protocol.

- **kubernetes.io/elb.cert-id**: "17e3b4f4bc40471c86741dc3aa211379"

  cert-id indicates the certificate ID in ELB certificate management. When **https** is configured for **protocol-port**, the certificate of the ELB listener will be set to the server certificate. When multiple HTTPS Services are released, they will use the same certificate.

The following is a configuration example for automatically creating a dedicated load balancer, in which key configurations are marked in red:

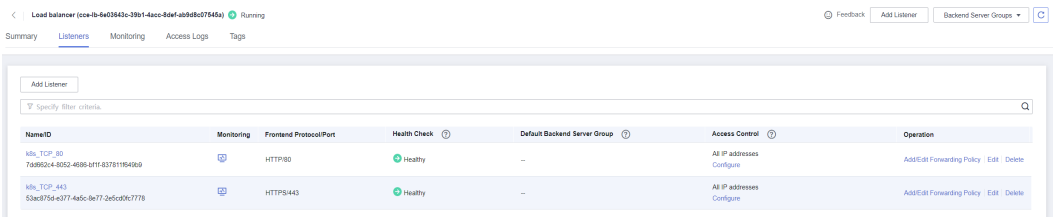
- Different ELB types and cluster versions have different requirements on flavors. For details, see **Table 3-167**.
- The two ports in **spec.ports** must correspond to those in **kubernetes.io/elb.protocol-port**. In this example, ports 443 and 80 are enabled with HTTPS and HTTP, respectively.

```
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
```

Use the preceding example configurations to create a Service. In the new ELB load balancer, you can see that the listeners on ports 443 and 80 are created.



3.7.3.4.4 Configuring SNI for a 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

If the SNI or the target annotation is deleted, the ELB configuration will not be modified.

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 **3.2.3.1 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**.

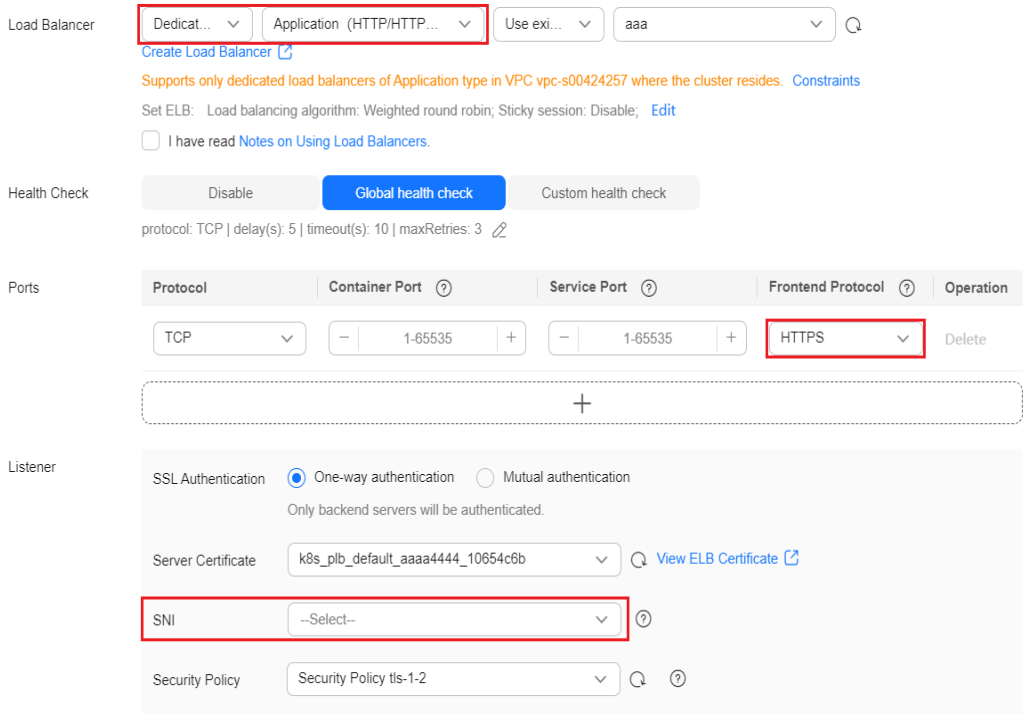
Step 3 Configure Service parameters. In this example, only mandatory parameters required for using SNI are listed. For details about how to configure other parameters, see **Creating a LoadBalancer Service**.

- **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 3-143**.
- **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.
    - **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.

Figure 3-147 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 3-168 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.

3.7.3.4.5 Configuring HTTP/2 for a 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.
- If the advanced configuration for enabling HTTP/2 or the target annotation is deleted, the ELB configuration will not be modified.

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 **3.2.3.1 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**.

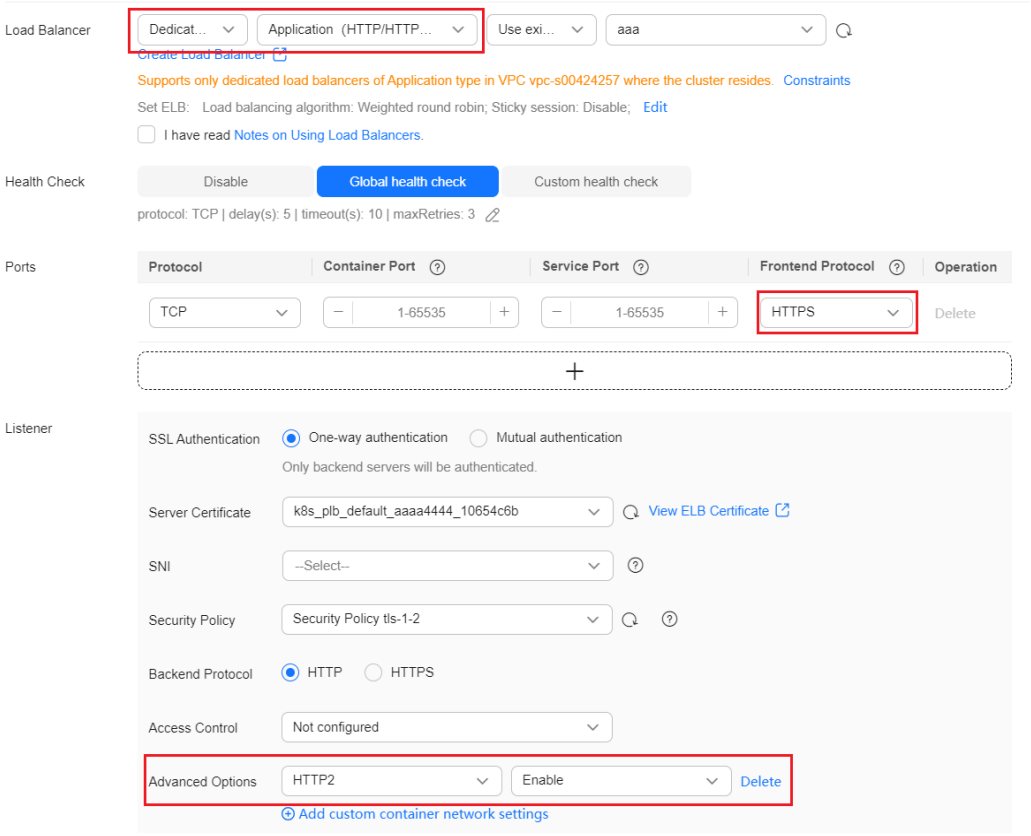
Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Creating a LoadBalancer Service**.

- **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 3-143**.
- **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
    - **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 3-148 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 3-169 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.

3.7.3.4.6 Configuring HTTP/HTTPS Headers for a 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**.
- If an HTTP/HTTPS header or the target annotation is deleted, the ELB configuration will not be modified.

Table 3-170 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 body 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 **3.2.3.1 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**.

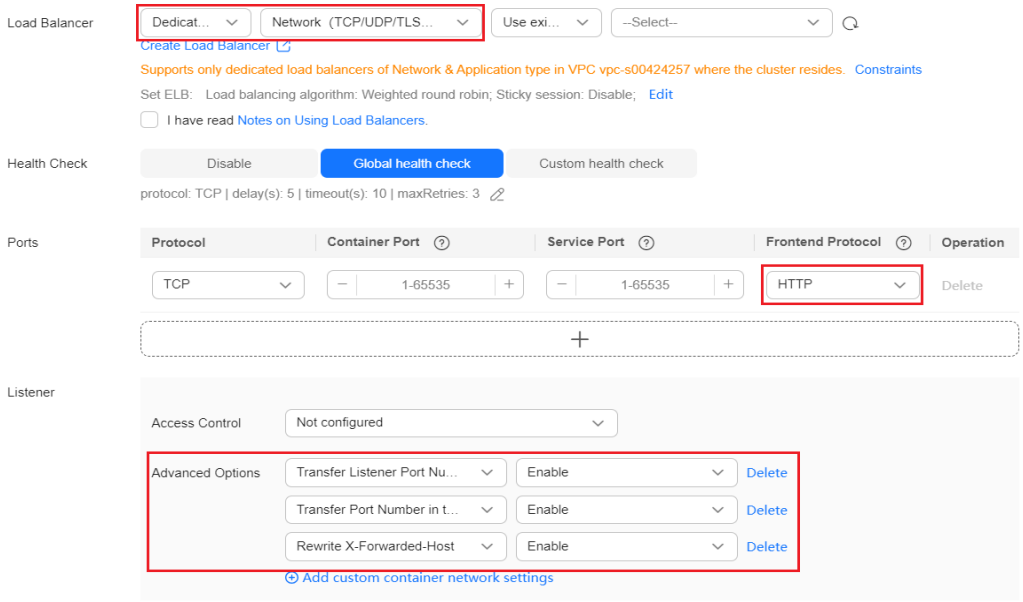
Step 3 Configure Service parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **Creating a LoadBalancer Service**.

- **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 3-143**.
- **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 body 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 3-149 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.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 3-171 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.

3.7.3.4.7 Configuring Timeout for a 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.

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.	

- If you delete the timeout configuration during a Service update, the timeout configuration on the existing listeners will be retained.

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**.

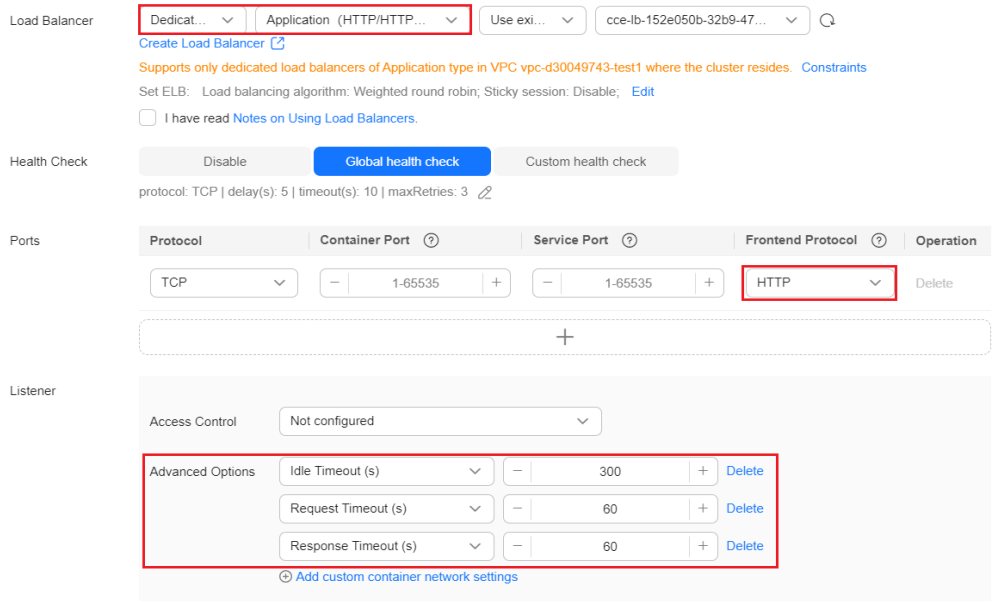
Step 3 Configure Service parameters. In this example, only mandatory parameters required for configuring timeout are listed. For details about how to configure other parameters, see **Creating a LoadBalancer Service**.

- **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 3-143**.

- **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 3-150 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.keepalive_timeout: '300'  # Timeout setting for client connections
  name: nginx
spec:
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```


Table 3-172 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**.

3.7.3.4.8 Configuring TLS for a 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 **3.2.3.1 Connecting to a Cluster Using kubectl**.

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 **Creating a LoadBalancer Service**.

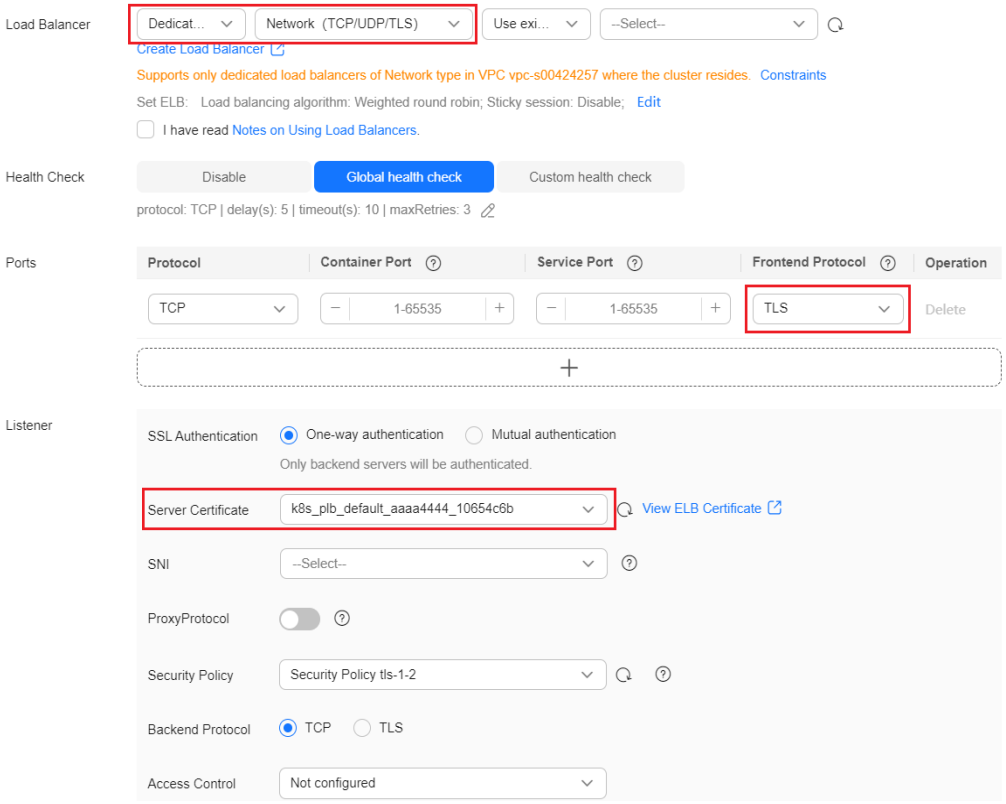
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an 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 3-143**.
- **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.
    - **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.
  - **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**.

Figure 3-151 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: 443
    nodePort: 0
    port: 443
    protocol: TCP
type: LoadBalancer
loadBalancerIP: **.**.**.**
```

Table 3-173 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.

3.7.3.4.9 Configuring GZIP Data Compression for a 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.

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 **3.2.3.1 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 **Creating a LoadBalancer Service**.

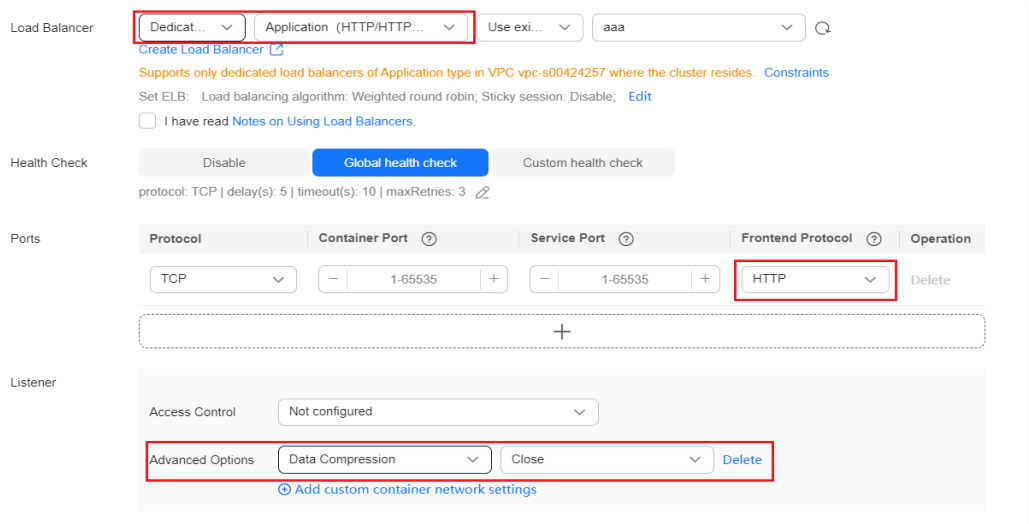
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Selector**: Add a label and click **Confirm**. The Service will use this label to select pods. You can also click **Reference Workload Label** to use the label of an 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 3-143**.

- **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 3-152 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 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.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 3-174 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.

3.7.3.4.10 Configuring a Blocklist/Trustlist Access Policy for a Service

When using a LoadBalancer Service, you can configure a trustlist or blocklist to specify the IP addresses that are allowed or denied to access 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.

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 3-143**.
- **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**
  - **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.

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 3-175 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**. **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.

3.7.3.4.11 Configuring Health Check on Multiple Service Ports

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.

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 3-176 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

3.7.3.4.12 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.

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.

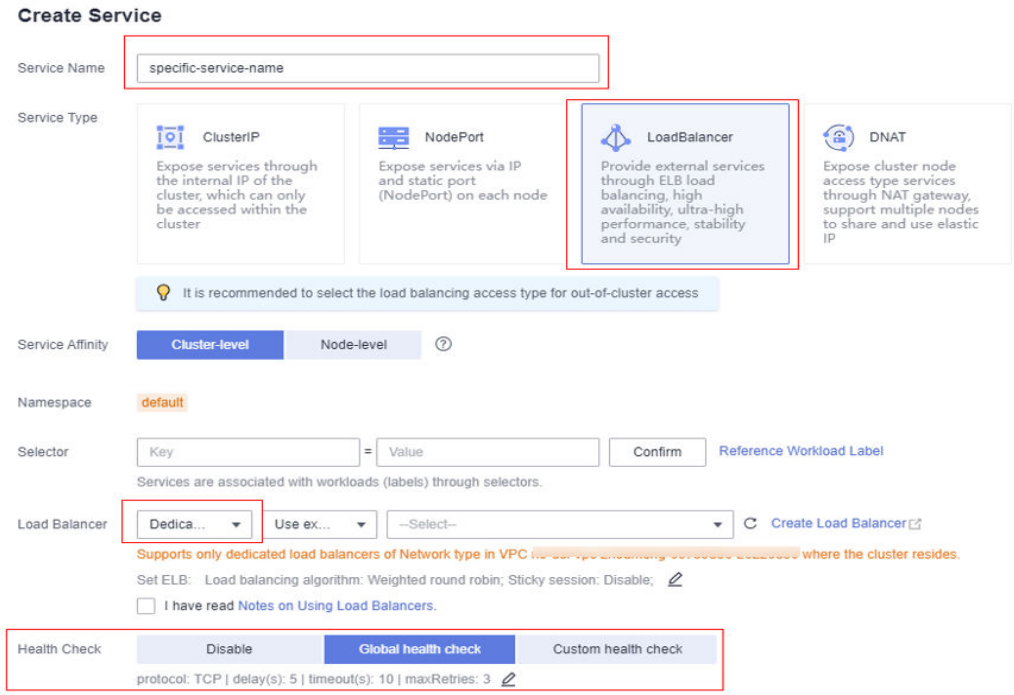
An event is displayed as follows:

```
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 3-153 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

3.7.3.4.13 Enabling Passthrough Networking for LoadBalancer Services

Background

A Kubernetes cluster can publish applications running on a group of pods as Services, which provide unified layer-4 access entries. For a Loadbalancer Service, kube-proxy configures the LoadbalanceIP in **status** of the Service to the local forwarding rule of the node by default. When a pod accesses the load balancer from within the cluster, the traffic is forwarded within the cluster instead of being forwarded by the load balancer.

kube-proxy is responsible for intra-cluster forwarding. kube-proxy has two forwarding modes: iptables and IPVS. iptables is a simple polling forwarding mode. IPVS has multiple forwarding modes but it requires modifying the startup parameters of kube-proxy. Compared with iptables and IPVS, load balancers provide more flexible forwarding policies as well as health check capabilities.

Solution

CCE supports passthrough networking. You can configure the **annotation** of **kubernetes.io/elb.pass-through** for the Loadbalancer Service. Intra-cluster access to the Service load balancer address is then forwarded to backend pods by the load balancer.

- 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, pass-through 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.

Constraints

- After passthrough networking is configured for a dedicated load balancer, in a CCE standard cluster, pods that run on the same node as the workload and pods that run on the same node cannot be accessed through the LoadBalancer Service.
- Passthrough networking is not supported for clusters of v1.15 or earlier.
- In IPVS network mode, the pass-through settings of Service connected to the same ELB 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 Nginx image to create a Deployment.

```
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
```

Step 2 For a LoadBalance Service type, set **kubernetes.io/elb.pass-through** to **true**. In this example, a shared load balancer named **james** is automatically created.

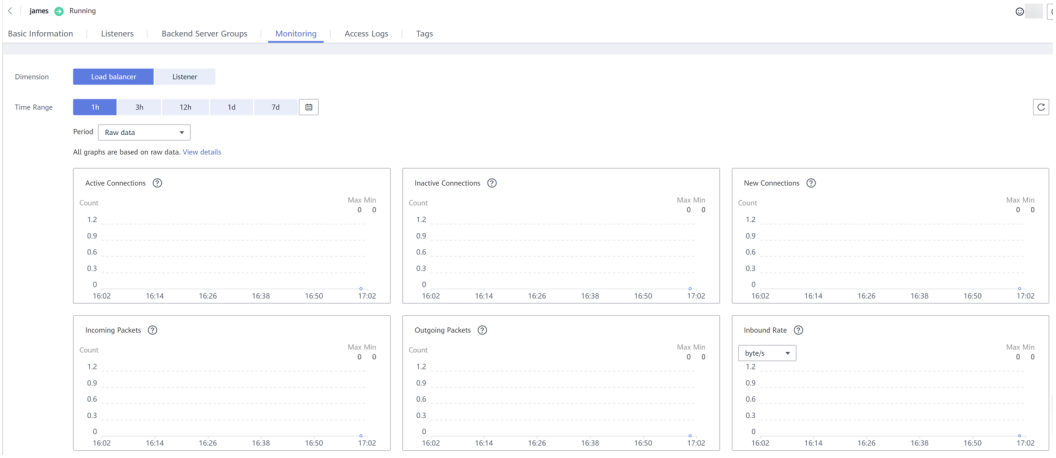
For details about how to create a LoadBalancer Service, see **LoadBalancer**.

```
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
```

----End

Verification

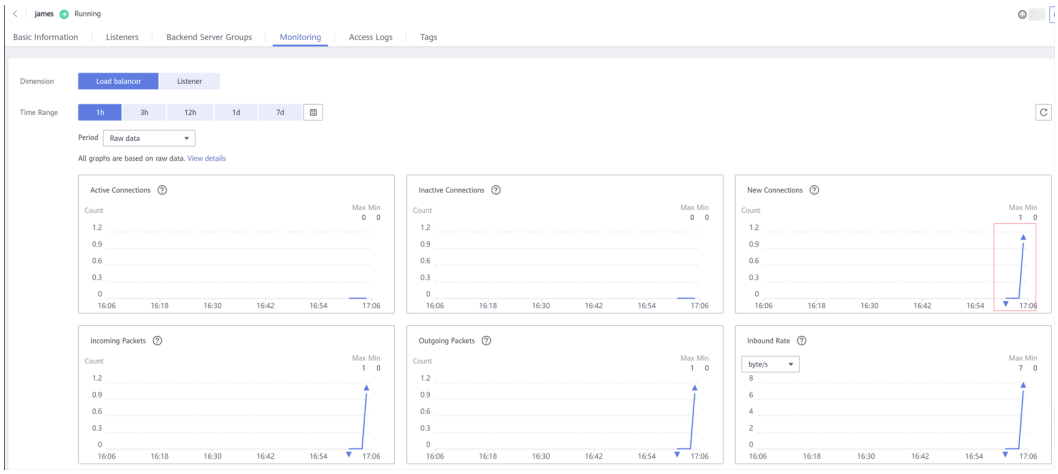
Check the ELB load balancer corresponding to the created Service. The load balancer name is **james**. The number of ELB connections is **0**, as shown in the following figure.



Use kubectl to connect to the cluster, go to an Nginx container, and access the ELB address. The access is successful.

```
# kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7c4c5cc6b5-vpncx   1/1     Running   0          9m47s
nginx-7c4c5cc6b5-xj5wl   1/1     Running   0          9m47s
# kubectl exec -it nginx-7c4c5cc6b5-vpncx -- /bin/sh
# curl 120.46.141.192
<!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>
```

Wait for a period of time and view the ELB monitoring data. A new access connection is created for the ELB, indicating that the access passes through the ELB load balancer as expected.



3.7.3.4.14 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.

Procedure

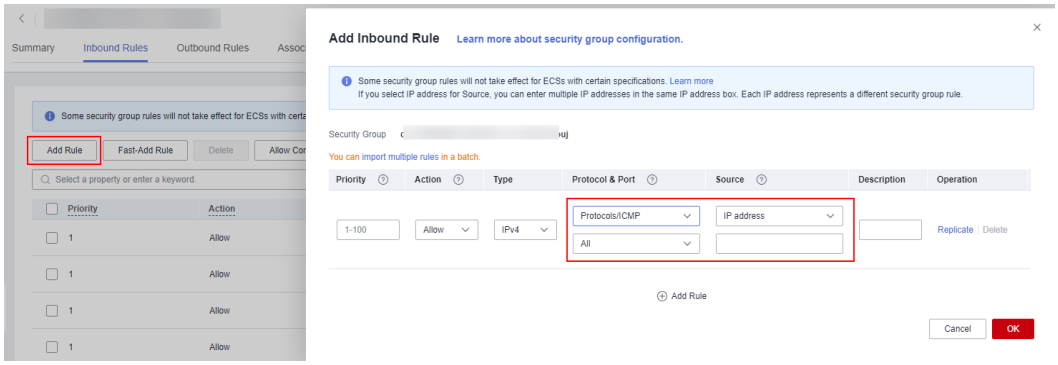
Step 1 Log in to the CCE console, choose **Service List** > **Networking** > **Virtual Private Cloud**, and choose **Access Control** > **Security Groups** in the navigation pane.

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

Cluster Type	ELB Type	Security Group	Protocol & Port	Allowed Source CIDR Block
	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 3-154 Adding a security group rule



Step 3 Click **OK**.

----End

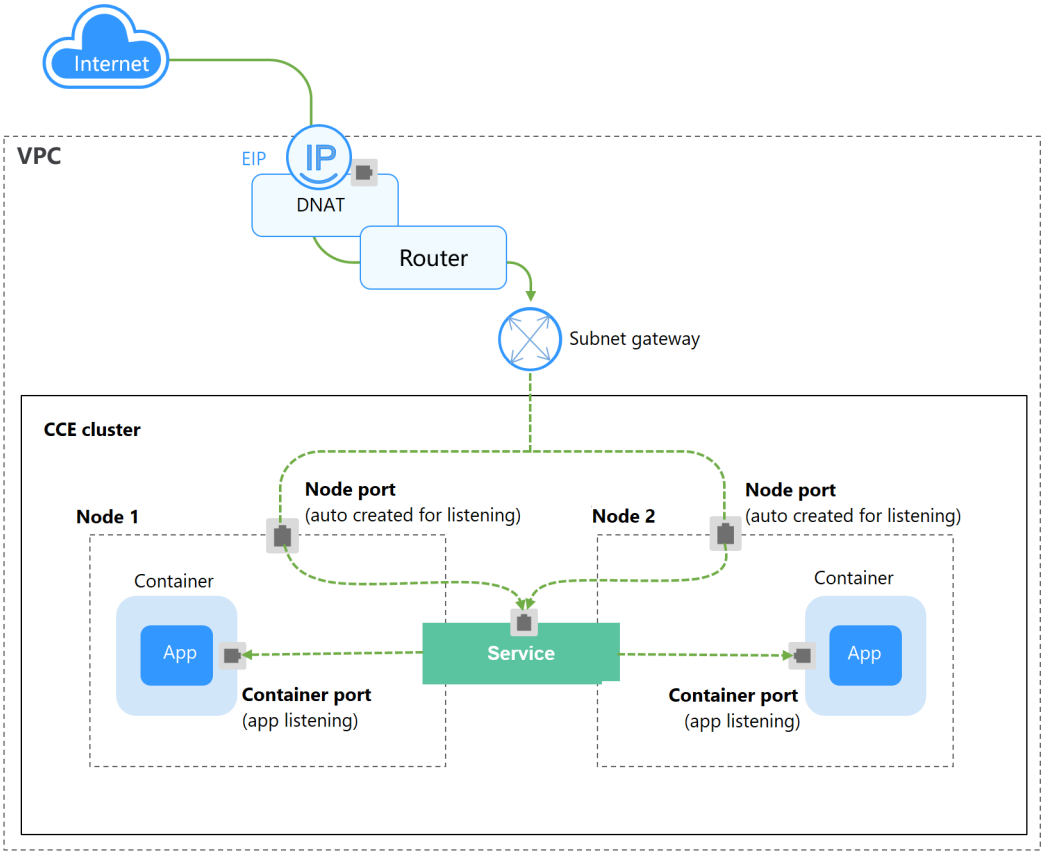
3.7.3.5 DNAT

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 3-155 DNAT

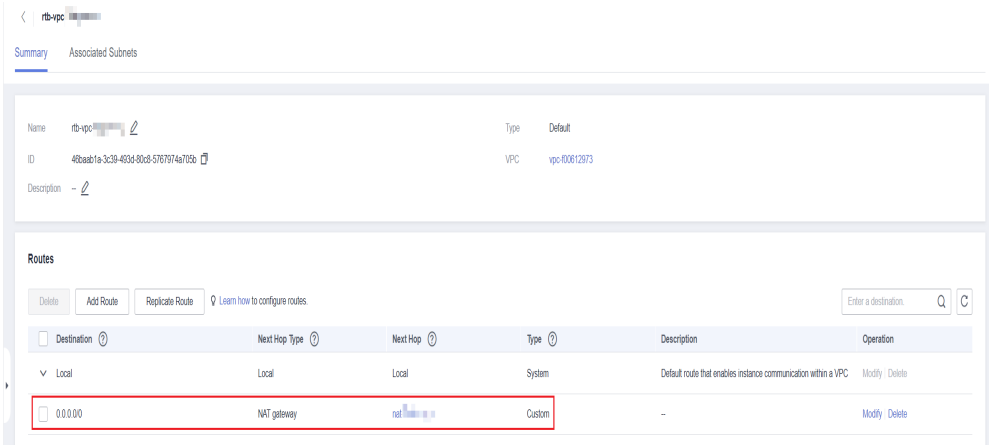


Constraints

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.



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**.
- **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).**
- **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 connect to the cluster. For details, see **3.2.3.1 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 3-100**.

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 3-177 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

3.7.3.6 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
```

3.7.4 Ingresses

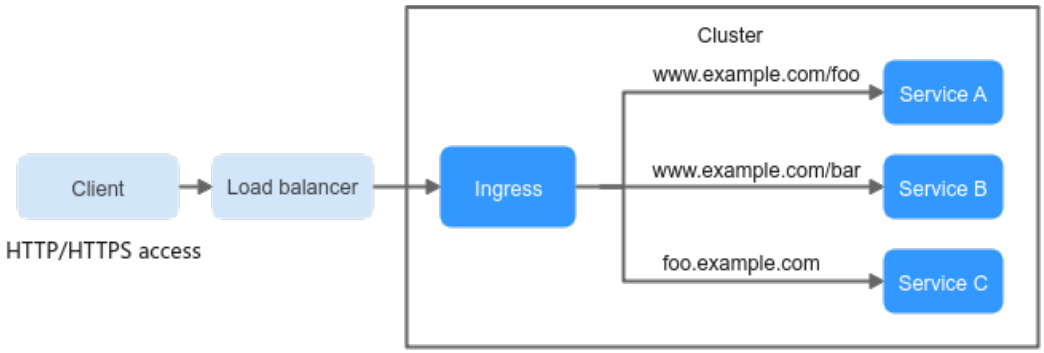
3.7.4.1 Overview

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 3-156**, you can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic.

Figure 3-156 Ingress diagram



The following describes the ingress-related definitions:

- Ingress object: a set of access rules that forward requests to specified Services based on domain names or URLs. It can be added, deleted, modified, and queried by calling APIs.
- Ingress Controller: an executor for request forwarding. 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.

Ingress Controllers provided by different vendors are implemented in different ways. Based on the types of load balancers, Ingress Controllers are classified into LoadBalancer Ingress Controller and Nginx Ingress Controller. Both of them are supported in CCE. LoadBalancer Ingress Controller forwards traffic through ELB. Nginx Ingress Controller uses the templates and images maintained by the Kubernetes community to forward traffic through the Nginx component.

Ingress Feature Comparison

Table 3-178 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.
	Dynamic loading is supported.	• Processes must be reloaded for non-backend endpoint changes, which 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 Ingresses**.

LoadBalancer Ingress Controller is deployed on a master node. All policies and forwarding behaviors are configured on the ELB side. 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 (CCE Turbo cluster + dedicated load balancer), 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-ingress. 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.

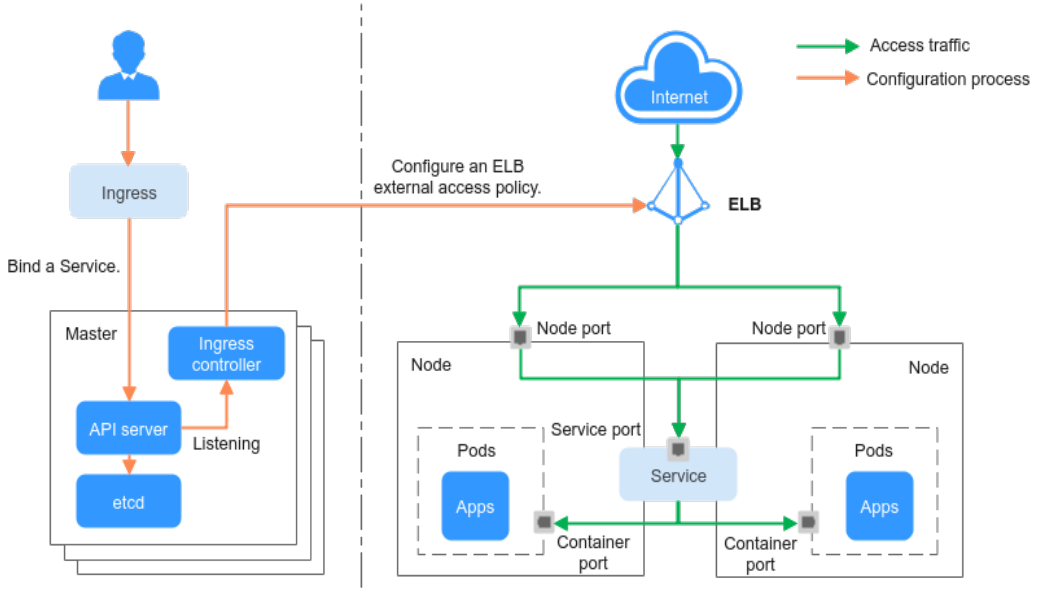
Working Rules of LoadBalancer Ingress Controller

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 corresponding Services using different URLs.

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). **Figure 3-157** shows the working rules of LoadBalancer Ingress Controller.

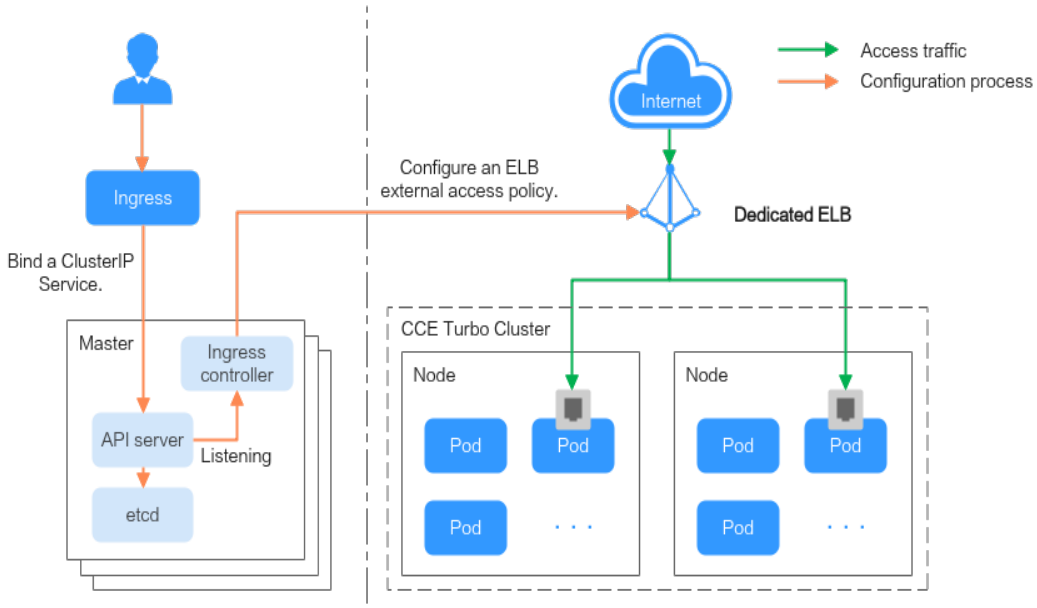
1. A user creates an ingress object and configures a traffic access rule in the ingress, including the load balancer, URL, SSL, and backend service port.
2. When Ingress Controller detects that the ingress object changes, it reconfigures the listener and backend server route on the ELB side according to the traffic access rule.
3. When a user accesses a workload, the traffic is forwarded to the corresponding backend service port based on the forwarding policy configured on ELB, and then forwarded to each associated workload through the Service.

Figure 3-157 Working rules of shared LoadBalancer ingresses in CCE standard and Turbo clusters



When you use **a dedicated load balancer in a CCE Turbo cluster**, pod IP addresses are allocated from the VPC and the load balancer can directly access the 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 3-158 Working rules of passthrough networking for dedicated LoadBalancer ingresses in CCE Turbo clusters



Working Rules of Nginx Ingress Controller

An Nginx ingress uses ELB as the traffic ingress. The **nginx-ingress** add-on is deployed in a cluster to balance traffic and control access.

NOTE

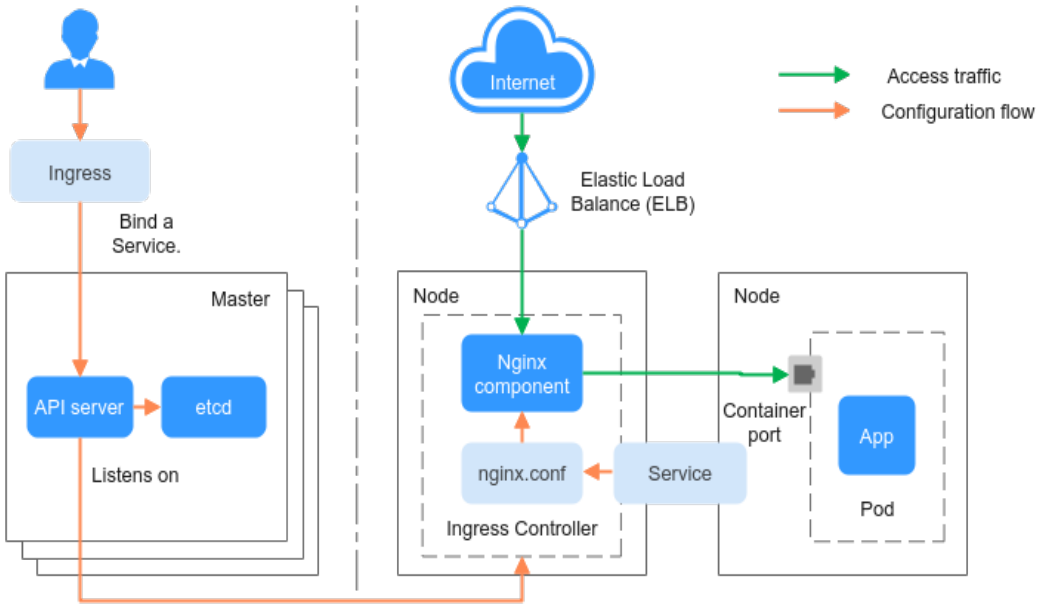
nginx-ingress uses the templates and images provided by the open-source community, and issues may occur during usage. CCE periodically synchronizes the community version to fix known vulnerabilities. Check whether your service requirements can be met.

You can visit the **open source community** for more information.

Nginx Ingress Controller is deployed on worker nodes through pods, which will result in O&M costs and Nginx component running overheads. **Figure 3-159** 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 3-159 Working rules of Nginx Ingress Controller



Services Supported by Ingresses

Table 3-179 lists the Services supported by LoadBalancer ingresses.

Table 3-179 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 (Failed to access the dedicated load balancers because no ENI is bound to the associated pod of the ClusterIP Service.)	Supported
CCE Turbo cluster	Shared load balancer	Not supported	Supported
	Dedicated load balancer	Supported	Not supported (Failed to access the dedicated load balancers because an ENI has been bound to the associated pod of the NodePort Service.)

Table 3-180 lists the services supported by Nginx ingresses.

Table 3-180 Services supported by Nginx ingresses

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

3.7.4.2 LoadBalancer Ingresses

3.7.4.2.1 Creating a LoadBalancer 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 **3.5.2.1 Creating a Deployment**, **3.5.2.2 Creating a StatefulSet**, or **3.5.2.3 Creating a DaemonSet**.
- **Services Supported by Ingresses** lists the Service types supported by LoadBalancer ingresses.

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.
- In a cluster using the IPVS proxy mode, if the ingress and Service use the same ELB load balancer, the ingress cannot be accessed from the nodes and containers in the cluster because kube-proxy mounts the LoadBalancer Service address to the ipvs-0 bridge. This bridge intercepts the traffic of the load balancer connected to the ingress. Use different load balancers for the ingress and Service.
- Do not connect an ingress and **a Service using 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 **3.14.7 Nginx Ingress Controller** add-on is installed. If this option is available, the nginx-ingress 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 **3.7.4.3.1 Creating Nginx Ingresses 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 3-181**.

Table 3-181 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/Application-oriented 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. The predefined tags are available to all resources that support tags. You can use predefined 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**
    - **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.
- **Certificate Source**: TLS secret and ELB server certificate are supported.
- **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.
  - **TLS secret**: For details about how to create a secret certificate, see **3.11.3 Creating a Secret**.
  - **ELB server certificate**: Use the certificate created in the ELB service.

  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 and 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 **Security Policy**.

  NOTE

  - **Security Policy** is available only when **HTTPS** is selected.
  - This function is supported only in clusters of v1.17.9 and 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 body 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.
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.	None

Configuration	Description	Restrictions
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.	None
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.	None
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 **Frontend 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**: Select whether to enable grayscale release.
  - **Do not turn on**: Grayscale release is disabled.
  - **Gray release by weight**: Request traffic is forwarded to the grayscale service by percentage.
  - **Gray release of request headers**: The request traffic whose header and header-value are matched is forwarded to the grayscale service.

- **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**: actual domain name. Ensure that the domain name has been registered and archived. Once a domain name rule is configured, you must use the domain name for access.
  - **URL 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.
  - **URL**: access path to be registered, 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. Services that do not meet search criteria are automatically filtered out.
  - **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 1,440 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. Configure a regular expression for the URL 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 **Creating an Ingress - Automatically Creating a Load Balancer** or **Creating an Ingress - Interconnecting with an Existing Load Balancer**.

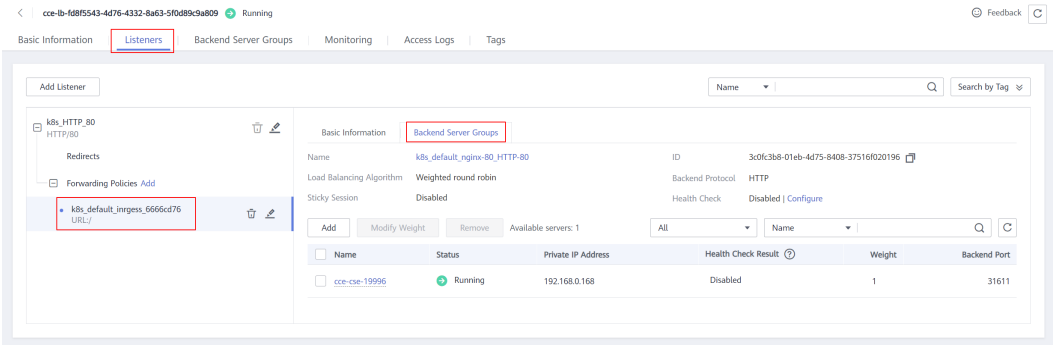
Step 4 Click **OK**. After the ingress is created, it is displayed in the ingress list.

On the ELB console, you can view the ELB automatically created through CCE. The default name is **cce-lb-ingress.UID**. Click the ELB name to access its details page. On the **Listeners** tab page, view the route settings of the ingress, including the URL, listener port, and backend server group port.

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.

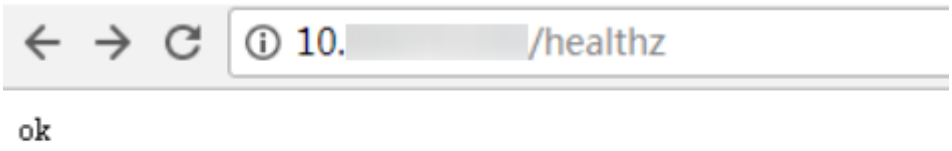
Figure 3-160 ELB routing configuration



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 3-161**.

Figure 3-161 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?**

3.7.4.2.2 Using kubectl to Create a LoadBalancer Ingress

Scenario

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 **Creating an Ingress - Automatically Creating a Load Balancer**.
- If a load balancer is available in the same VPC, perform the operation by referring to **Creating an Ingress - Interconnecting with an Existing Load Balancer**.

Prerequisites

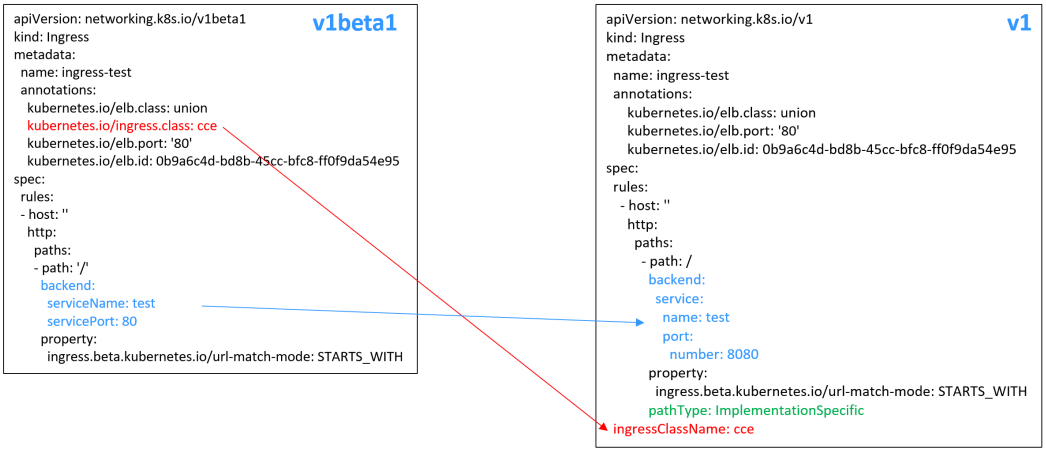
- An ingress provides network access for backend workloads. Ensure that a workload is available in a cluster. If no workload is available, deploy a sample Nginx workload by referring to **3.5.2.1 Creating a Deployment**, **3.5.2.2 Creating a StatefulSet**, or **3.5.2.3 Creating a DaemonSet**.
- **Services Supported by 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).

Ingress Description of networking.k8s.io/v1

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 changed from **kubernetes.io/ingress.class** in **annotations** to **spec.ingressClassName**.
- The format of **backend** is 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 (/).



Creating an Ingress - Automatically Creating a Load Balancer

The following describes how to run the kubectl command to automatically create a load balancer when creating an ingress.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 Description of networking.k8s.io/v1**.

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 version 1.21 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 3-182 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: • **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?**
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.

Parameter	Mandatory	Type	Description
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 3-183**. **Example** • If a public network load balancer will be automatically created, set this parameter to the following value: '{"type":"public","bandwidth_name":"cce-bandwidth-******","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • If a private network load balancer will be automatically created, set this parameter to the following value: {"type":"inner","name":"A-location-d-test"}
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 archived. Once a domain name rule is configured, 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.

Parameter	Mandatory	Type	Description
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
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 3-183 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 **querying 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 **querying 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 **querying 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	Specifies the 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
```

kubectl get ingress

If information similar to the following is displayed, the ingress has been created and the workload is accessible.

```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 4 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

Creating an Ingress - Interconnecting with an Existing Load Balancer

CCE allows you to connect to an existing load balancer when creating an ingress.

NOTE

- 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 3-184 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.

3.7.4.2.3 Configuring a LoadBalancer Ingress Using Annotations

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.

- **Interconnection with ELB**
- **Using HTTP/2**
- **Configuring ELB Certificates**
- **Interconnecting with HTTPS Backend Services**
- **Configuring Timeout for an Ingress**
- **Adding Resource Tags**
- **Blocklist/Trustlist**
- **Configuring a Custom Listening Port**
- **Enabling GZIP**

Interconnection with ELB

Table 3-185 Annotations for interconnecting with ELB

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.class	String	Select a proper load balancer type. The value can be: • **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/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 **3.7.4.2.2 Using kubectl to Create a LoadBalancer Ingress**.	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 interconnected**. 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

Parameter	Type	Description	Supported Cluster Version
kubernetes.io/elb.ip	String	Mandatory **when an existing load balancer is to be interconnected**. 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
kubernetes.io/elb.autocreate	**Table 3-194** Object	Mandatory **when load balancers are automatically created**. **Example** • If a public network load balancer will be automatically created, set this parameter to the following value: '{"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}' • If a private network load balancer will be automatically created, set this parameter to the following value: {"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 later than v1.11.7-r0.	Mandatory for clusters earlier than v1.11.7-r0 Discarded in clusters later than v1.11.7-r0

The following shows how to use the preceding annotations:

- Associate an existing load balancer. For details, see **Creating an Ingress - Interconnecting with an Existing Load Balancer**.
- Automatically create a load balancer. For details, see **Creating an Ingress - Automatically Creating a Load Balancer**.

Using HTTP/2

Table 3-186 Annotations of 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, see **3.7.4.2.7 Configuring HTTP/2 for a LoadBalancer Ingress**.

Configuring ELB Certificates

Table 3-187 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

For details, see **Using the ELB Certificate**.

Interconnecting with HTTPS Backend Services

Table 3-188 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

For details, see **3.7.4.2.11 Interconnecting LoadBalancer Ingresses with HTTPS Backend Services**.

Configuring Timeout for an Ingress

Table 3-189 Annotations of configuring ingress redirection rules

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**	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

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**	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, see **3.7.4.2.14 Configuring Timeout for a LoadBalancer Ingress**.

Adding Resource Tags

Table 3-190 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, see **Creating an Ingress - Automatically Creating a Load Balancer**.

Blocklist/Trustlist

Table 3-191 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**. **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

For details, see **3.7.4.2.17 Configuring a Blocklist/Trustlist Access Policy for a LoadBalancer Ingress**.

Configuring a Custom Listening Port

A custom listening port can be configured for an ingress. In this way, both ports 80 and 443 can be exposed.

Table 3-192 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. • Only newly created ingresses are allowed. Additionally, after multiple listening ports are configured, annotations cannot 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. • Advanced forwarding policies are not supported.	v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later

For example, if an existing ELB is used, the configuration is as follows:

```
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: test-https
  namespace: default
spec:
  ingressClassName: cce
  rules:
```



```
- host: xxx.com
  http:
    paths:
    - backend:
        service:
          name: test
          port:
            number: 8888
      path: /
      pathType: ImplementationSpecific
      property:
        ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

Enabling GZIP

Table 3-193 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

For details, see **3.7.4.2.15 Configuring GZIP Data Compression for a LoadBalancer Ingress**.

Parameters for Automatically Creating a Load Balancer

Table 3-194 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 **querying 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 **querying 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 **querying 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	Specifies the 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.

3.7.4.2.4 Configuring HTTPS Certificates for LoadBalancer Ingresses

Ingresses support TLS certificates and secure your Services with HTTPS.

You can use a TLS secret certificate configured in the cluster and the ELB certificate.

NOTE

If HTTPS is enabled for the same port of the same load balancer of multiple ingresses, you must select the same certificate.

Using a TLS Secret Certificate

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 **3.11.3 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
```

View 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

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.

The following uses the automatically created load balancer as an example. The YAML file is configured 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: foo.bar.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
```

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: foo.bar.com
    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 3-195 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.security_policy_id	No	String	The ID of the custom security group policy in 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 For details of cipher suites for each security policy, see **Table 3-196**.
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 **3.7.4.2.5 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 3-196 tls_ciphers_policy parameter description

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

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
```

View the created ingress.

kubectl get ingress

If information similar to the following is displayed, the ingress has been created and the workload is accessible.

```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 6 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 the ELB Certificate

To use the ELB certificate, you can specify the annotations **kubernetes.io/elb.tls-certificate-ids**.

NOTE

1. If you specify both the IngressTLS certificate and the ELB certificate, the latter is used.
2. CCE does not check whether the ELB certificate is valid. It only checks whether the certificate exists.
3. Only clusters of v1.19.16-r2, v1.21.5-r0, v1.23.3-r0, or later support the ELB certificate.

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: 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 3-197 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.

3.7.4.2.5 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.

NOTE

- This function is supported only in clusters of v1.15.11 and later.
- 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.

You can enable SNI when the preceding conditions are met. The following uses the automatic creation of a load balancer as an example. In this example, **sni-test-secret-1** and **sni-test-secret-2** are SNI certificates. The domain names specified by the certificates must be the same as those in the certificates.

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.top  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-1
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-2
  rules:
  - host: 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
```

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.top  # Domain name specified when a certificate is issued
    secretName: sni-test-secret-1
  - hosts:
      - example.com  # Domain name specified when a certificate is issued
```



```
  secretName: sni-test-secret-2
rules:
- host: example.com
  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
```

3.7.4.2.6 LoadBalancer Ingresses to Multiple Services

Ingresses can route to multiple backend Services based on different matching policies. The **spec** field in the YAML file is set as below. You can access **www.example.com/foo**, **www.example.com/bar**, and **foo.example.com/** to route to three different backend Services.

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.

```
...
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
```


3.7.4.2.7 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, you can add the following field to the ingress annotation to enable the use of HTTP/2:

```
kubernetes.io/elb.http2-enable: 'true'
```

NOTE

- An HTTPS-compliant load balancer supports HTTP/2.
- This function is available in clusters of version v1.23.13-r0, v1.25.8-r0, v1.27.5-r0, or v1.28.3-r0.
- If the advanced configuration for enabling HTTP/2 or the target annotation is deleted, the ELB configuration will not be modified.

The following shows an example YAML file where an existing load balancer is associated:

```
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: 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 3-198 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.

3.7.4.2.8 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.

Constraints

- The advanced forwarding policy of LoadBalancer ingresses is available only to dedicated load balancers.
- The advanced forwarding policy of LoadBalancer ingresses is available only in clusters of version v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.

Configuring a Rule for Redirecting an Ingress to a URL Using YAML

An ingress can be redirected to a URL using annotations. Example:

```
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: 8888
        property:
          ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
        pathType: ImplementationSpecific
ingressClassName: cce
```

Table 3-199 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**.

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
```

3.7.4.2.9 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**.

Constraints

- The advanced forwarding policy of LoadBalancer ingresses is available only to dedicated load balancers.
- The advanced forwarding policy of LoadBalancer ingresses is available only in clusters of version v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.

Configuring an Ingress URL Rewriting Rule Using YAML

URL rewriting rules of an ingress can be configured using annotations. The following is an example:

```
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: 8888
            property:
              ingress.beta.kubernetes.io/url-match-mode: REGEX
            pathType: ImplementationSpecific
  ingressClassName: cce
```


Table 3-200 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.

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**.

3.7.4.2.10 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.

Constraints

- The advanced forwarding policy of LoadBalancer ingresses is available only to dedicated load balancers.
- The advanced forwarding policy of LoadBalancer ingresses is available only in clusters of version v1.23.14-r0, v1.25.9-r0, v1.27.6-r0, v1.28.4-r0, or later.

Redirecting HTTP to HTTPS Using YAML

You can use annotations to redirect the requests of an ingress to a listener. The following is an example:

```
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: 8888
          property:
            ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
          pathType: ImplementationSpecific
ingressClassName: cce
```

Table 3-201 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.listen-ports	Yes	String	Multi-port listening configuration, which cannot be modified after being configured. Format: The value is a JSON string, for example: `[{"HTTP":80},{"HTTPS":443}]` Parameter: The port number ranges from 1 to 65535.
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. Either this annotation or the annotation of a grayscale release can be configured.

3.7.4.2.11 Interconnecting LoadBalancer Ingresses with HTTPS Backend Services

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
```

Constraints

- This function is available only in clusters of v1.23.8, v1.25.3, or later.
- 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.

Configuration Example

An ingress configuration example 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
```

3.7.4.2.12 Interconnecting LoadBalancer Ingresses with gRPC Backend Services

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
```

Constraints

- This function is available only in clusters of v1.23.10-r20, v1.25.5-r20, v1.27.2-r20, v1.28.1-r0, or later versions.
- Ingresses can interconnect with gRPC backend services only when dedicated load balancers are used. This function depends on ELB listeners and is available only in certain regions. Obtain these regions on the CCE console.
- When an ingress interconnects with a gRPC backend service, the ingress protocol must be HTTPS and HTTP/2 must be enabled.

Configuring gRPC Backend Services on the 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.

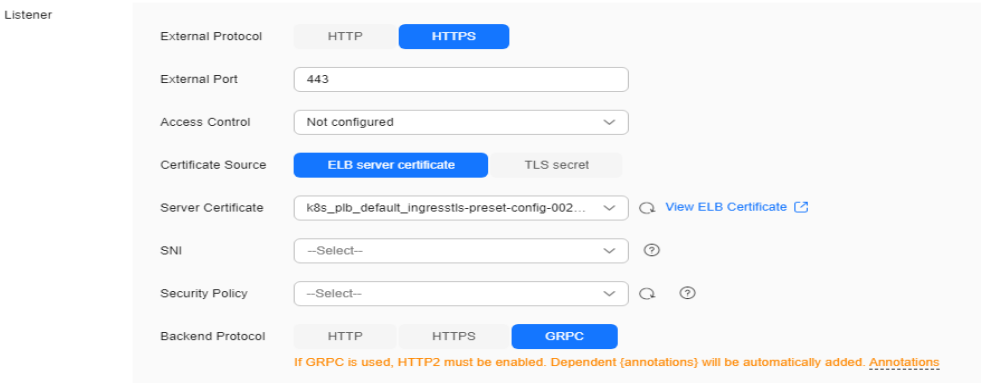
Step 3 Configure ingress parameters.

- **Name**: Customize the name of an ingress, for example, **ingress-demo**.

- **Load Balancer**: Select an interconnected load balancer or automatically create a load balancer. Only dedicated load balancers are supported.
- **Listener**: Set **External Protocol** to **HTTPS** and **Backend Protocol** to **GRPC**. Configure other parameters based on site requirements. For details, see **3.7.4.2.1 Creating a LoadBalancer Ingress on the Console**.

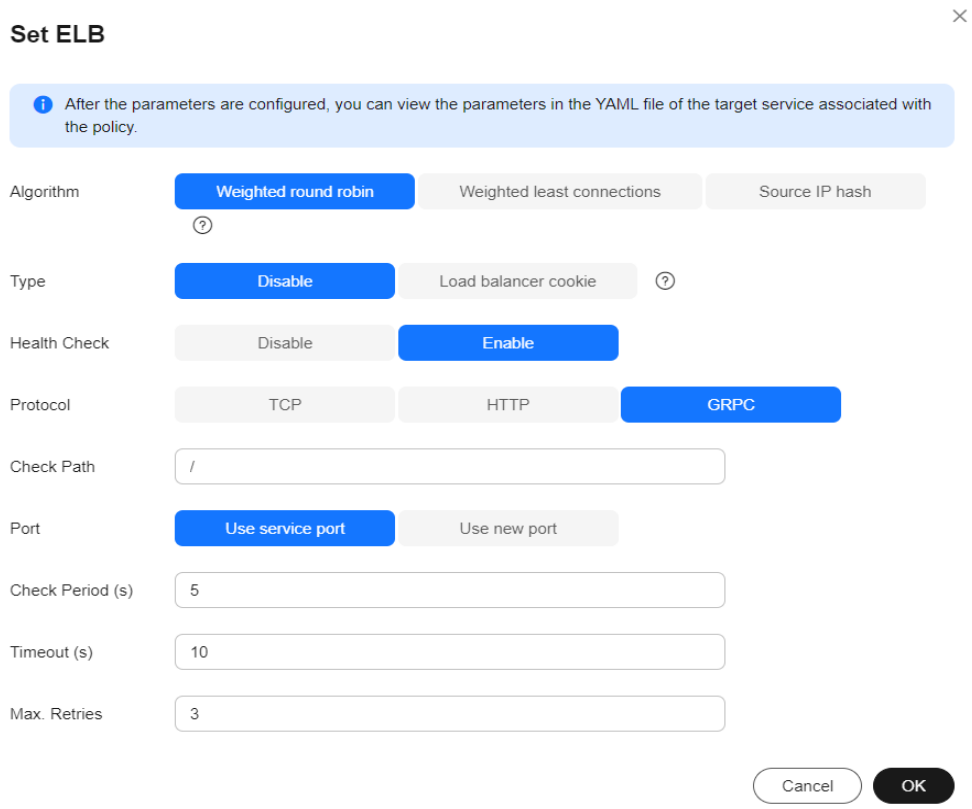
  NOTE

  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.



- **Forwarding Policy**: Select a destination Service so that requests can be forwarded to this Service for processing.

  (Optional) Enable health check and select gRPC.



Step 4 Click **OK**.

----End

Configuring gRPC Backend Services Using kubectl

An ingress configuration example 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
```

3.7.4.2.13 Configuring HTTP/HTTPS Headers 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**.
- If an HTTP/HTTPS header or the target annotation is deleted, the ELB configuration will not be modified.

Table 3-202 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.

Header	Feature	Description
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 body 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 **3.2.3.1 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.

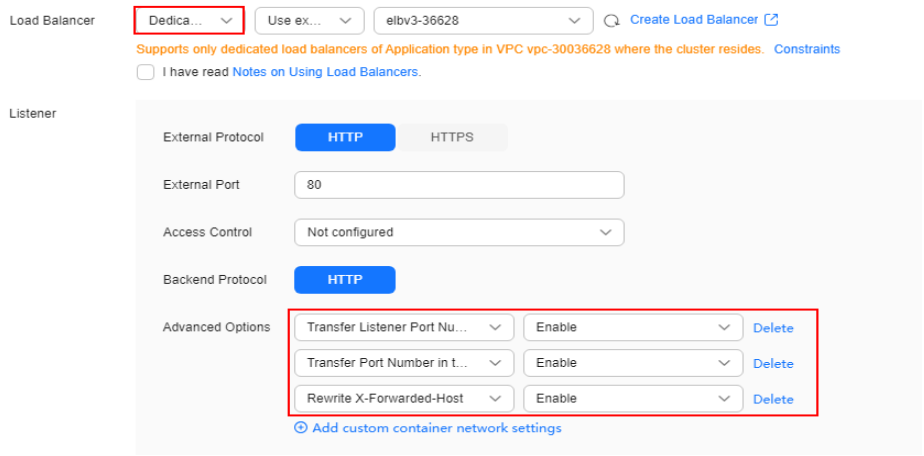
Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **3.7.4.2.1 Creating a LoadBalancer Ingress on the Console**.

- **Name**: can be the same as the workload name.
- **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 **load balancers**.
- **Listener**
  - **External Protocol**: **HTTP** and **HTTPS** are available. This section uses HTTP as an example.
  - **External Port**: port number that is open to the ELB service address. The port number is configurable.
  - **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 body and transferred to backend servers.	This function is available only for dedicated load balancers.

Figure 3-162 Configuring HTTP/HTTPS headers



- **Forwarding Policy**: Specify a domain name matching rule and the target Service to be accessed. 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 will be forwarded to the target Service for processing.

Step 4 Click **OK**.

----End

Using kubectl

The following uses an existing created load balancer as an example. The YAML file is configured 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: 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 3-203 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.

3.7.4.2.14 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.

Constraints

- The following table lists the scenarios where timeout can be configured for a Service.

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	

- If you delete the timeout configuration during an ingress update, the timeout configuration on the existing listeners will be retained.

Using the CCE Console

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

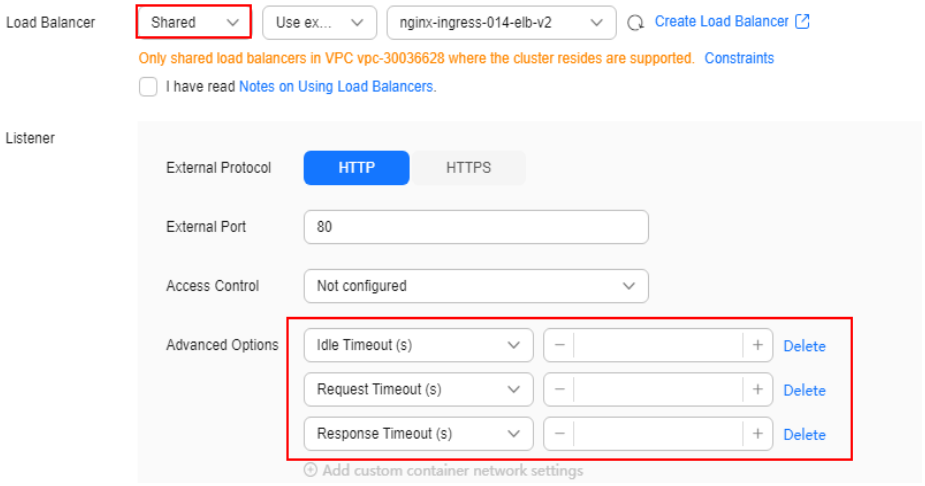
Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **3.7.4.2.1 Creating a LoadBalancer Ingress on the Console**.

- **Name**: can be the same as the workload name.
- **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 **load balancers**.
- **Listener**
  - **External Protocol**: **HTTP** and **HTTPS** are available. This section uses HTTP as an example.
  - **External Port**: port number that is open to the ELB service address. The port number is configurable.
  - **Advanced Options**

Configuration	Description
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.
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.
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.

Figure 3-163 Configuring timeout



- **Forwarding Policy**: Specify a domain name matching rule and the target Service to be accessed. 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 will be forwarded to the target Service for processing.

Step 4 Click **OK**.

----End

Using kubectl

An ingress configuration example 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 3-204 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**.
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**.

3.7.4.2.15 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.

Constraints

- This feature is available in the following versions:
  - 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
  - Versions later than v1.28
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 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.

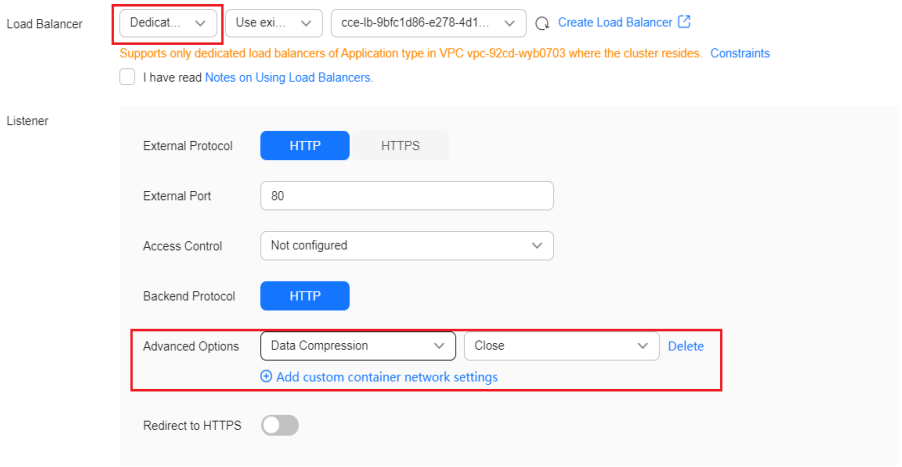
Step 2 In the navigation pane, choose **Services & Ingresses**. Click the **Ingresses** tab and click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters. In this example, only mandatory parameters are listed. For details about how to configure other parameters, see **3.7.4.2.1 Creating a LoadBalancer Ingress on the Console**.

- **Name**: can be the same as the workload name.
- **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 **load balancers**.
- **Listener**
  - **External Protocol**: **HTTP** and **HTTPS** are available. This section uses HTTP as an example.
  - **External Port**: port number that is open to the ELB service address. The port number is configurable.
  - **Advanced Options**

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 function is available only for dedicated load balancers.

Figure 3-164 Configuring GZIP data compression



- **Forwarding Policy**: Specify a domain name matching rule and the target Service to be accessed. 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 will be forwarded to the target Service for processing.

Step 4 Click **OK**.

----End

Using kubectl

The following uses an existing created load balancer as an example. The YAML file is configured 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 3-205 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.

3.7.4.2.16 Configuring Grayscale Release for a LoadBalancer Ingress

Dedicated LoadBalancer ingresses support grayscale release by:

- Weight
- Header
- Cookie

NOTE

Grayscale release of ingresses depends on ELB. Before using grayscale release, submit a service ticket to request for enabling ELB grayscale release.

Constraints

- This feature takes effect only in the following versions:
  - 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
- After a grayscale ingress is created, do not delete the original ingress.
- If the listeners of a load balancer associate with multiple ingresses configured with multiple grayscale release policies, the header-based policy has the highest priority, cookie has the second highest priority, and weight has the lowest priority.

Parameters

Table 3-206 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.
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: ● An array contains 1 to 10 characters. ● Enter 1 to 128 characters. Spaces and double quotation marks are not supported. The following wildcards are supported: * (matching zero or more characters) and ? (matching 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 (!%'"()*+,./:=?@^\\-_`~).
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: ● An array contains 1 to 10 characters. ● 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

Deploying Grayscale Release

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: nginx1.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. 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: nginx1.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. 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:

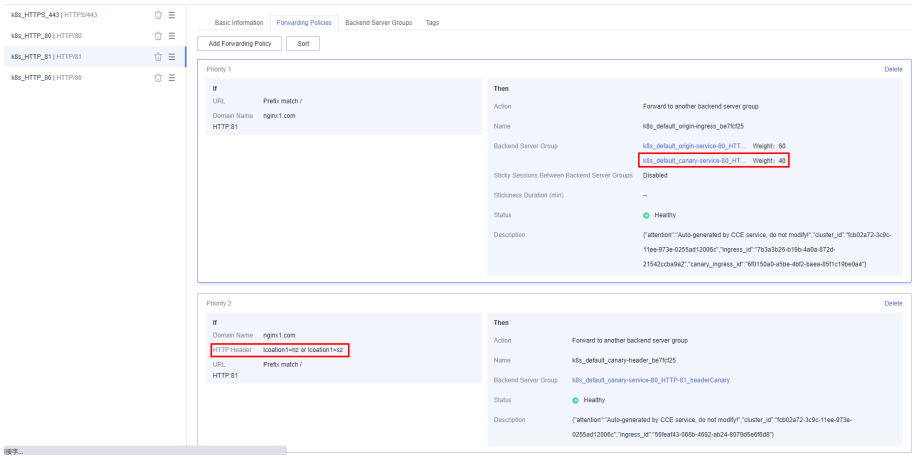
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: canary-header-ingress
  namespace: default
  annotations:
    kubernetes.io/elb.canary: 'true'
    kubernetes.io/elb.canary-by-header: 'location'                          // Set the header key to location.
    kubernetes.io/elb.canary-by-header-value: '{"values":["hz","sz","sh"]}'  // Set the header values to
hz, sz, and sh.
    kubernetes.io/elb.class: performance
    kubernetes.io/elb.id: e491f4e7-2351-4984-ad0a-8569e5e964a3
    kubernetes.io/elb.port: '81'
spec:
  ingressClassName: cce
  rules:
    - host: nginx1.com
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: canary-service
                port:
                  number: 80
            property:
              ingress.beta.kubernetes.io/url-match-mode: STARTS_WITH
```

5. 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-header-ingress   cce     nginx1.com   88.88.88.165   80      46s
canary-weight-ingress   cce     nginx1.com   88.88.88.165   80      117s
origin-ingress          cce     nginx1.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: nginx1.com" -H "location: hz" http://88.88.88.165:81/
```

ii. Expected result:

```
$ new
```

Attempt the access for multiple times. All the returned results are **new**.

iii. Access the Service without using a request header.

```
$ curl -H "Host: nginx1.com" http://88.88.88.165:81/
```

iv. 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: nginx1.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: nginx1.com" -H "location: hz" http://88.88.88.165:81/
$ curl -H "Host: nginx1.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

3.7.4.2.17 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 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 **Ingresses** tab, and click **Create Ingress** in the upper right corner.

Step 3 Configure access control parameters for the ingress.

- **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.

For details about how to configure other parameters, see **3.7.4.2.1 Creating a LoadBalancer Ingress on the Console**.

Step 4 Click **OK**.

----End

Using kubectl

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: 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 3-207 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**. **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.

3.7.4.3 Nginx Ingresses

3.7.4.3.1 Creating Nginx Ingresses 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 **3.5.2.1 Creating a Deployment**, **3.5.2.2 Creating a StatefulSet**, or **3.5.2.3 Creating a DaemonSet**.
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see **3.7.3.2 ClusterIP** or **3.7.3.3 NodePort**.
- To add an Nginx ingress, ensure that the NGINX Ingress Controller add-on has been installed in the cluster. For details, see **Installing the Add-on**.

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 **3.14.7 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 **3.11.3 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 **3.14.7 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.
  - **URL 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.
- **URL**: access path to be registered, 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. Services that do not meet search criteria are automatically filtered out.
- **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. You can use **annotations** to query the configurations supported by nginx-ingress.

Step 4 Click **OK**.

After the ingress is created, it is displayed in the ingress list.

----End

3.7.4.3.2 Using kubectl to Create an Nginx Ingress

Scenario

This section uses an **Nginx workload** as an example to describe how to create an Nginx ingress using kubectl.

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 **3.5.2.1 Creating a Deployment**, **3.5.2.2 Creating a StatefulSet**, or **3.5.2.3 Creating a DaemonSet**.

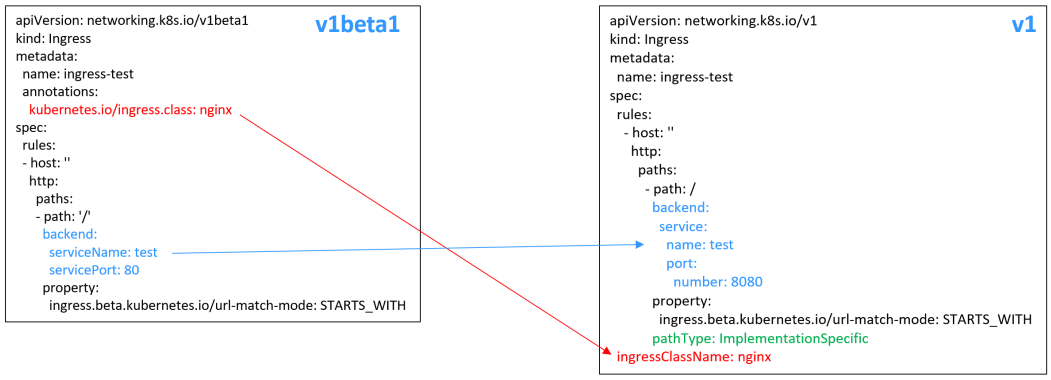
- A ClusterIP or NodePort Service has been configured for the workload. For details about how to configure the Service, see **3.7.3.2 ClusterIP** or **3.7.3.3 NodePort**.

Ingress Description of networking.k8s.io/v1

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 changed from **kubernetes.io/ingress.class** in **annotations** to **spec.ingressClassName**.
- The format of **backend** is 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 (/).



Creating an Nginx Ingress

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 Description of networking.k8s.io/v1**.

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.
```

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 3-208 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 cannot be used if the nginx-ingress add-on is not installed. This parameter is mandatory when an ingress is created by calling the API.
ingressClassName	Yes (only for clusters of v1.23 or later)	String	**nginx**: indicates that Nginx ingress is used. This option cannot be used if the nginx-ingress add-on is not installed. This parameter is mandatory when an ingress is created by calling the API.

Parameter	Mandatory	Type	Description
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 archived. Once a domain name rule is configured, 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 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
```

View the created ingress.

kubectl get ingress

If information similar to the following is displayed, the ingress has been created and the workload is accessible.


```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 4 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

3.7.4.3.3 Configuring Nginx Ingresses Using Annotations

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 ingresses support 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 Body Size**
- **Documentation**

Ingress Type

Table 3-209 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 **3.7.4.3.2 Using kubectl to Create an Nginx Ingress**.	Only clusters of v1.21 or earlier

For details about how to use the preceding annotations, see **3.7.4.3.2 Using kubectl to Create an Nginx Ingress**.

Configuring a Redirection Rule

Table 3-210 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 **3.7.4.3.5 Configuring Redirection Rules for an Nginx Ingress**.

Configuring a URL Rewriting Rule

Table 3-211 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 **3.7.4.3.6 Configuring URL Rewriting Rules for Nginx Ingresses**.

Interconnecting with HTTPS Backend Services

Table 3-212 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 **3.7.4.3.7 Interconnecting Nginx Ingresses with HTTPS Backend Services**.

Creating a Consistent Hashing Rule for Load Balancing

Table 3-213 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 **3.7.4.3.8 Nginx Ingresses Using Consistent Hashing for Load Balancing**.

Customized Timeout Interval

Table 3-214 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 Body Size

Table 3-215 Annotations of customizing 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 is returned to the client. You can use this parameter to adjust the upper limit of the body size. Example: `nginx.ingress.kubernetes.io/proxy-body-size: 8m`

Documentation

For details about annotation parameters supported by Nginx ingresses, see **Annotations**.

3.7.4.3.4 Configuring HTTPS Certificates for Nginx Ingresses

HTTPS certificates can be configured for ingress to provide security services.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 **3.11.3 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
```

View 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
```

View the created ingress.

kubectl get ingress

If information similar to the following is displayed, the ingress has been created and the workload is accessible.

```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 6 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

3.7.4.3.5 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:

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.
```

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:

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.
```


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:

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.
```

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**.

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.
```

3.7.4.3.6 Configuring URL Rewriting Rules for Nginx Ingresses

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 Rewriting Rules

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: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            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: /$2
spec:
  rules:
    - host: 'rewrite.bar.com'
      http:
        paths:
          - path: '/something(/|$)(.*)'
            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 indicates that all characters matched by the second parenthesis (.*) are filled in the **nginx.ingress.kubernetes.io/rewrite-target** annotation.

For example, the preceding ingress definition will result in the following rewrites:

- rewrite.bar.com/something rewrites to rewrite.bar.com/.
- rewrite.bar.com/something/ rewrites to rewrite.bar.com/.
- rewrite.bar.com/something/new rewrites to rewrite.bar.com/new.

In the nginx-ingress-controller container, you can view all ingress configurations in the **nginx.conf** file in the **/etc/nginx** directory. The rewriting rule in the preceding example generates a Rewrite command and writes it to the **location** field in the **nginx.conf** file.


```
## start server rewrite.bar.com
    server {
        server_name rewrite.bar.com ;
        ...
        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 rewrite.bar.com
```

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.

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.

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 two rules, the **/something** path is added to the access URL.

- When a user accesses **rewrite.bar.com/stylesheets/new.css**, it rewrites to **rewrite.bar.com/something/stylesheets/new.css**.
- When a user accesses **rewrite.bar.com/images/new.jpg**, it rewrites to **rewrite.bar.com/something/images/new.jpg**.

3.7.4.3.7 Interconnecting Nginx Ingresses with HTTPS Backend Services

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:

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.
```

3.7.4.3.8 Nginx Ingresses Using Consistent Hashing for Load Balancing

The native Nginx supports multiple load balancing rules, including weighted round robin and IP hash. An 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 the 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

An Nginx ingress can use the **nginx.ingress.kubernetes.io/upstream-hash-by** annotation to configure consistent hashing rules. The following is an example:

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.

Documentation

Custom NGINX upstream hashing

3.7.5 DNS

3.7.5.1 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.

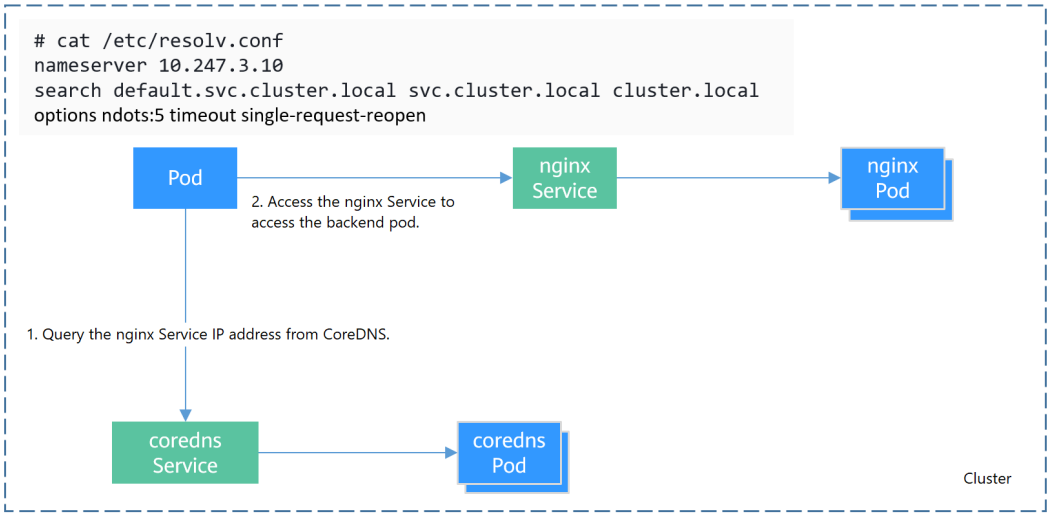
```
$ kubectl get svc -n kube-system
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                  AGE
coredns       ClusterIP   10.247.3.10    <none>        53/UDP,53/TCP,8080/TCP   13d
```

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 3-165 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.

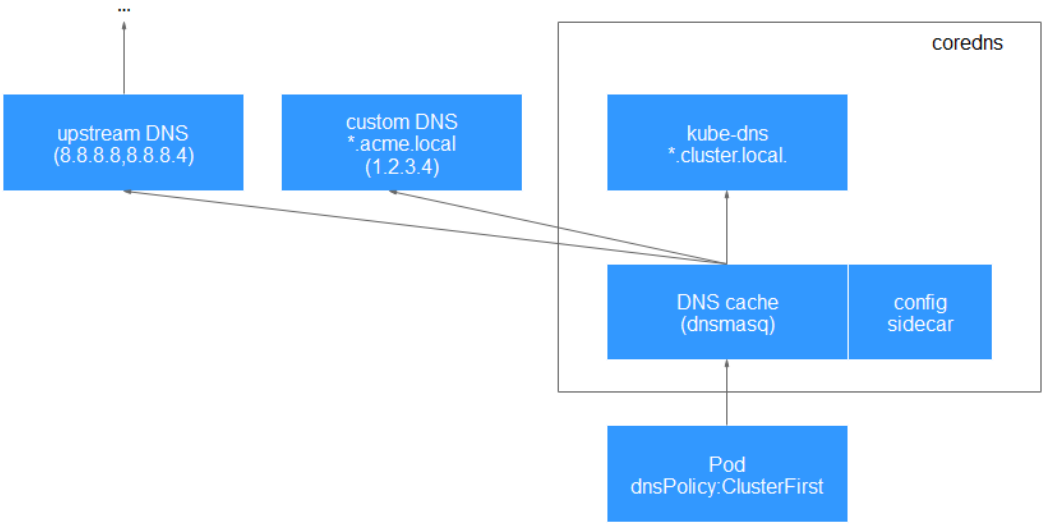
Routing

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 3-166 Routing



Related Operations

You can also configure DNS in a workload. For details, see **3.7.5.2 DNS Configuration**.

You can also use CoreDNS to implement user-defined domain name resolution. For details, see **3.7.5.3 Using CoreDNS for Custom Domain Name Resolution**.

You can also use DNSCache to improve the DNS resolution performance. For details, see **3.7.5.4 Using NodeLocal DNSCache to Improve DNS Performance**.

3.7.5.2 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. There are cases when Kubernetes workloads initiate unnecessary DNS queries. This makes DNS overloaded if there are many concurrent DNS queries. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about DNS, see **3.14.2 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

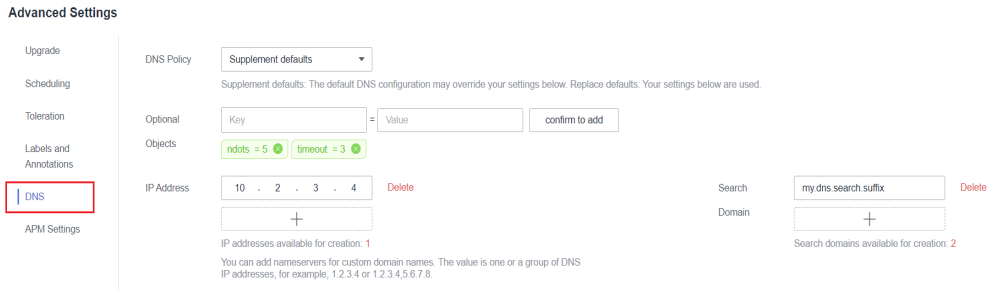
Kubernetes provides DNS-related configuration options for applications. The use of application's DNS configuration can effectively reduce unnecessary DNS queries in certain scenarios and improve service concurrency. The following procedure uses an Nginx application as an example to describe how to add 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 **3.5.2 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 3-216**.
  - **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**: **nameservers** in the **dnsConfig**. You can configure the domain name server for the 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.



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 3-216 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 3-217 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
```

3.7.5.3 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 add it on the console, which is easy to operate.
- **Using the CoreDNS Hosts plug-in to configure resolution for any domain name**: You can add any record set, 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 a service in the cluster**: A nickname is assigned to the Kubernetes Service. You do not need to know the IP address of the resolution record in advance.
- **Using the CoreDNS Forward plug-in to set the self-built DNS as the upstream DNS**: The self-built DNS can manage a large number of resolution records. You do not need to modify 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

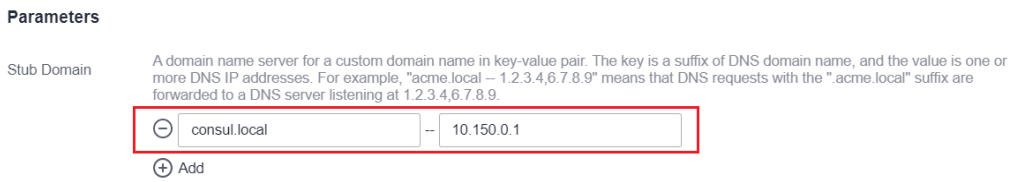
Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works.

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**.



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.

The corresponding Corefile content is as follows:

```
.: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 . /etc/resolv.conf {
        policy random
    }
    reload
    ready {$POD_IP}:8081
}
consul.local:5353 {
    bind {$POD_IP}
    errors
    cache 30
    forward . 10.150.0.1
}
```

----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 the advanced configuration under **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 CoreDNS plug-in. 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/**.

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.

The corresponding Corefile content is as follows:

```
.: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
}
```

----End

Adding the CoreDNS Rewrite Configuration to Point the Domain Name to Services in the Cluster

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 the advanced configuration under **Parameters** and add the following content to the **plugins** field:

```
{
  "name": "rewrite",
  "parameters": "name example.com example.default.svc.cluster.local"
}
```

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.

The corresponding Corefile content is as follows:

```
.: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
}
```

----End

Using CoreDNS to Cascade Self-Built DNS

By default, CoreDNS uses the **/etc/resolv.conf** file of the node for resolution. 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 the advanced configuration under **Parameters** and modify the following content in the **plugins** field:

```
{
    "configBlock": "policy random",
    "name": "forward",
    "parameters": ". 192.168.1.1"
}
```

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.

The corresponding Corefile content is as follows:

```
.: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
}
```

----End

3.7.5.4 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:

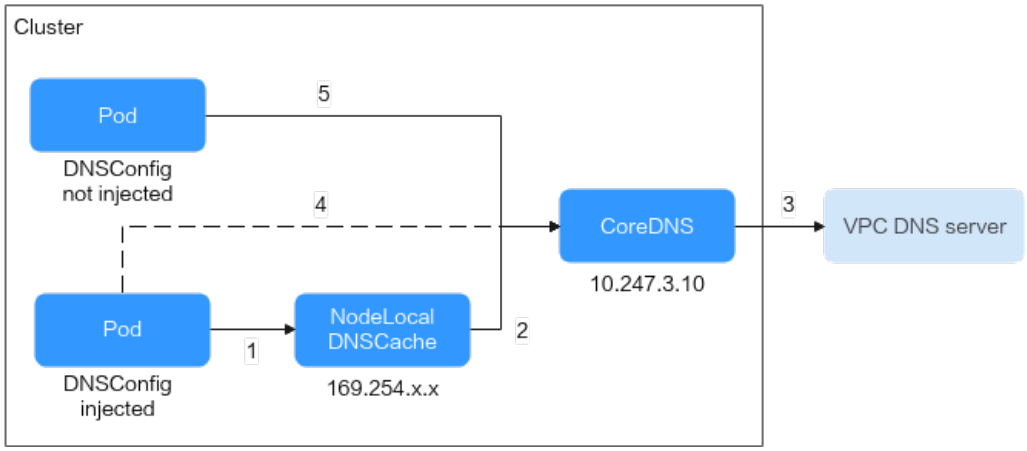
- Increased delay: CoreDNS needs to process more requests, which may slow down the DNS query and affect service performance.
- Increased resource usage: To ensure DNS performance, CoreDNS requires higher specifications.

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 3-167 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.

Constraints

- Only clusters of version 1.19 or later support the **3.14.16 NodeLocal DNSCache** add-on.
- 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 **3.14.16 NodeLocal DNSCache** for you to install 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"
}
```

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.

- **enable_dnsconfig_admission**: After this function is enabled, a DNSConfig dynamic injection controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-localdns-injection=enabled** based on Admission Webhook, and automatically configures **Pod dnsConfig** that uses the DNS cache. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.
- **Target Namespace**: This parameter is available after **DNSConfig Automatic Injection** is enabled. Only NodeLocal DNSCache of v1.3.0 or later supports this function.
  - **All Enabled**: 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 configuration**: 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. (Pods cannot be automatically injected into system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

Auto injection

The following conditions must be met:

- **Automatic DNSConfig injection** has been enabled during the add-on installation.
- The **node-local-dns-injection=enabled** label has been added 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**
- The new pod does not run in system namespaces such as kube-system and kube-public namespace.
- 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**.

After auto injection is enabled, the following **dnsConfig** settings are automatically added to the created pod. In addition to the NodeLocal DNSCache address 169.254.20.10, the CoreDNS 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
...
```

Manual configuration

Manually add the **dnsConfig** settings to the pod.

Create a pod and add the NodeLocal DNSCache IP address 169.254.20.10 to the DNSConfig nameservers configuration.

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

```
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
```

Common Issues

- How Do I Avoid an Automatic DNSConfig Injection?

  Solution:

  To prevent automatic DNSConfig injection for a workload, 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 automatic DNSConfig injection.
      spec:
        containers:
          - name: container-1
            image: nginx:latest
            imagePullPolicy: IfNotPresent
        imagePullSecrets:
          - name: default-secret
  ```

3.7.6 Container Network Settings

3.7.6.1 Host Network

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

If a pod uses the host network, it occupies a host port. The pod IP is the host IP. To use the host network, you must confirm pods do not conflict with each other in terms of the host ports they occupy. Do not use the host network unless you know exactly which host port is used by which pod.

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 pods of the hostNetwork type, 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()
```

3.7.6.2 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 QoS rate limiting for inter-pod access to prevent this problem.

Constraints

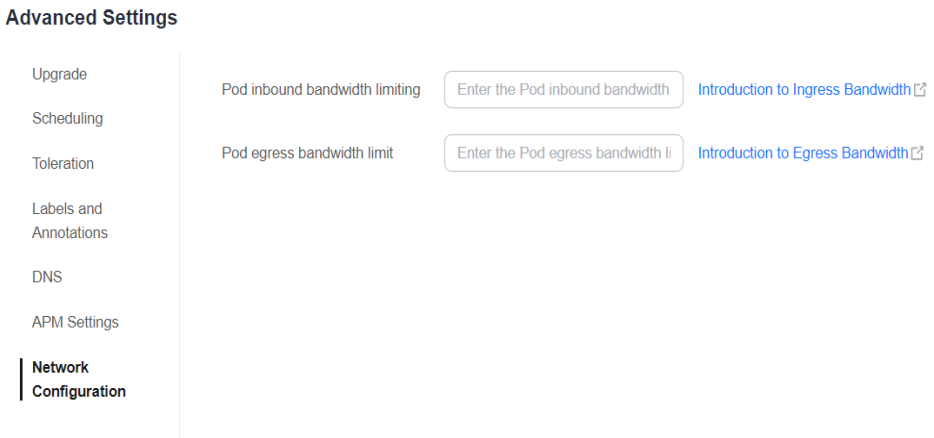
The following shows constraints on setting the rate limiting for inter-pod access:

Constraint Type	Tunnel network model	VPC network model	Cloud Native 2.0 Network Model
Supported versions	All versions	Clusters of v1.19.10 and later	Clusters of v1.19.10 and later
Supported runtime types	Only common containers		
Supported pod types	Only non-HostNetwork pods		
Supported scenarios	Inter-pod access, pods accessing nodes, and pods accessing services		
Constraints	None	None	• Pods access external cloud service CIDR blocks 100.64.0.0/10 and 214.0.0.0/8. • Traffic rate limiting of health check
Value range of rate limit	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.		

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 3-168 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.

3.7.6.3 Container Tunnel Network Settings

3.7.6.3.1 Network Policies

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.

Network policies are classified into the following types:

- **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.

Constraints

- Only clusters that use the tunnel network model support network policies. Network policies are classified into the following types:
  - Ingress: All versions support this type.
  - Egress: This rule type cannot be set currently.
- Network isolation is not supported for IPv6 addresses.

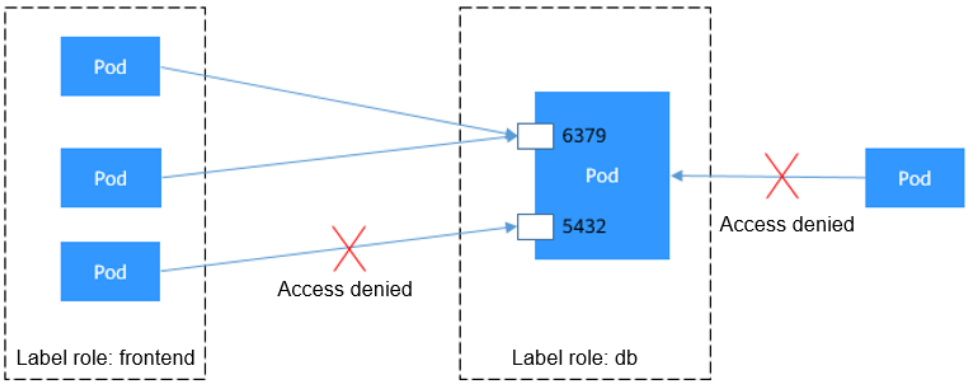
Using Ingress Rules

- **Using podSelector to specify the access scope**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  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 traffic from the pods with the "role=frontend" label is allowed.
        matchLabels:
          role: frontend
    ports:                    # Only TCP can be used to access port 6379.
    - protocol: TCP
      port: 6379
```

The following figure shows how podSelector works.

Figure 3-169 podSelector



- **Using namespaceSelector to specify the access scope**

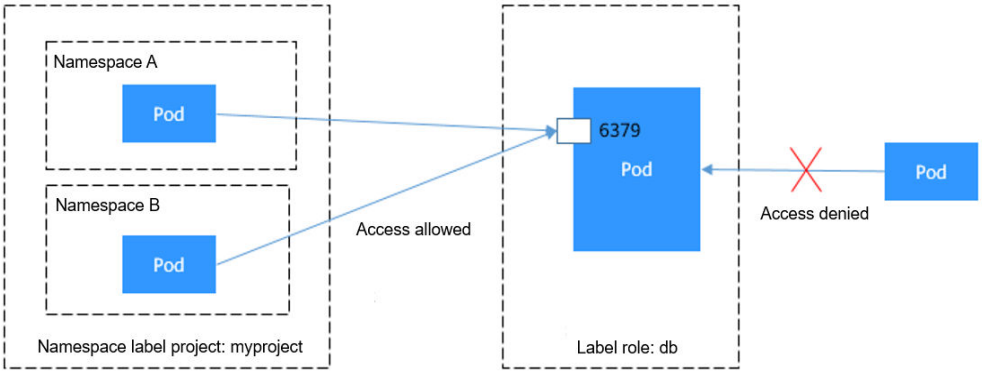
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector:                # The rule takes effect for pods with the role=db label.
    matchLabels:
      role: db
  ingress:                    # This is an ingress rule.
  - from:
    - namespaceSelector:      # Only traffic from the pods in the namespace with the
```



```
"project=myproject" label is allowed.
      matchLabels:
        project: myproject
  ports:                  # Only TCP can be used to access port 6379.
  - protocol: TCP
    port: 6379
```

The following figure shows how namespaceSelector works.

Figure 3-170 namespaceSelector



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 3-218**.

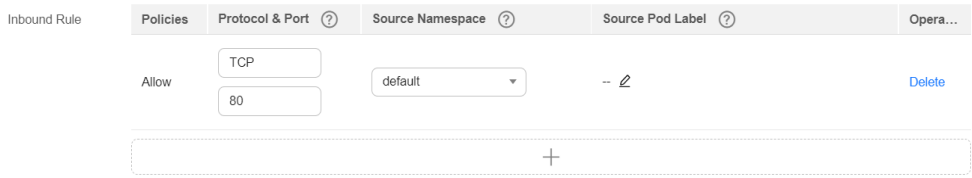


Table 3-218 Adding an inbound rule

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.

Parameter	Description
Source 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

3.7.6.4 Cloud Native Network 2.0 Settings

3.7.6.4.1 Binding a Custom Security Group to a Workload

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**.

Constraints

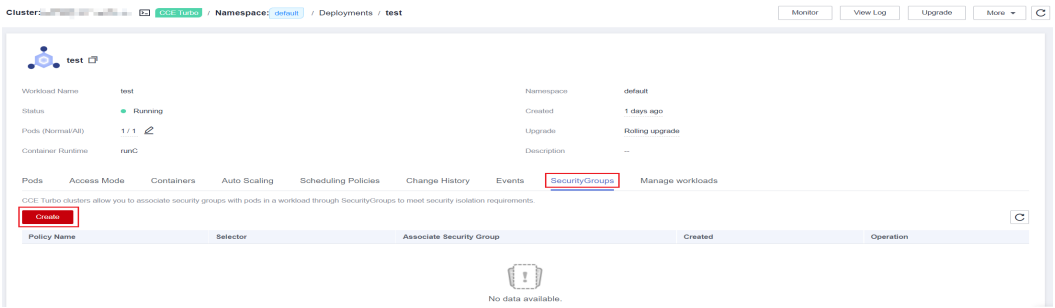
- This function is supported for CCE Turbo clusters of v1.19 and later. Upgrade your CCE Turbo clusters if their versions are earlier than v1.19.
- A workload can be bound to a maximum of five security groups.

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 3-219**.

Table 3-219 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 ⓘ next to the security group name, and you can view 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 connect to the cluster. For details, see **3.2.3.1 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 3-220 describes the parameters in the YAML file.

Table 3-220 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 view 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

3.7.6.4.2 Binding a Subnet and Security Group to a Namespace or Workload

Scenario

In a CCE Turbo cluster, you can configure subnets and security groups for containers by namespace or workload using NetworkAttachmentDefinition **CRDs**.

If you want to configure a specified container subnet and security group for a specified namespace or workload, create a container network configuration and associate it with the target namespace or workload. In this way, service subnets can be planned or services can be securely isolated.

The following table lists the resources that a container network configuration can be associated with.

Table 3-221 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 use the same subnet and security group configurations.	The workloads associated with the same container network configuration 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.
Constraints	The namespaces associated with different container network configurations must be unique.	Only the custom container network configurations that are not associated with any namespace can be specified.

Constraints

- Only the default container network configuration **default-network** supports container ENI prebinding. The speed of creating pods using a custom container network configuration is slower than that of creating pods using **default-network**. Therefore, this function is not suitable for ultra-fast pod scaling.
- **default-network** cannot be deleted.
- 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**.

Using the CCE Console to Create a Container Network Configuration of the Namespace Type

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.

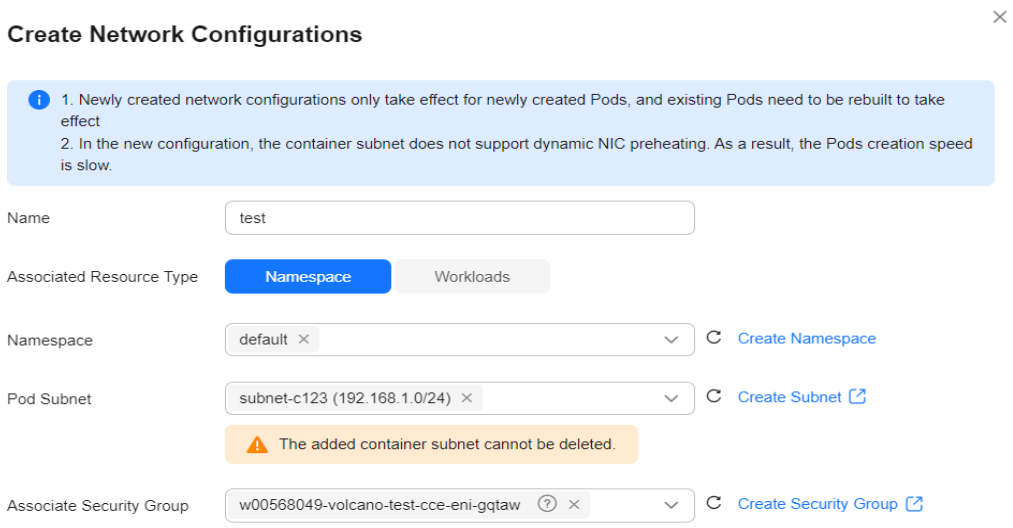
NOTE

If **default-network** is available in the 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**.

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.

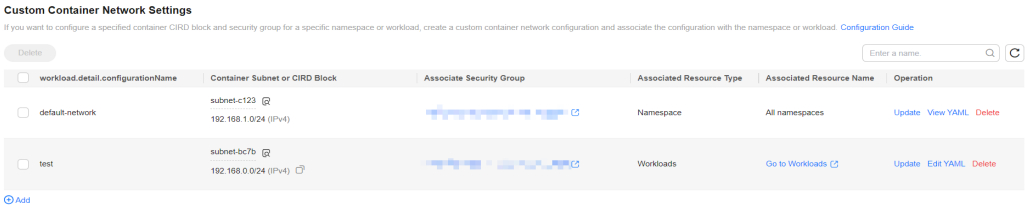
- **Name**: Enter a name that contains a maximum of 253 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.
- **Associated Resource Type**: resource type associated with the custom container network configuration. For details, see **Table 3-221**. To create a container network 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.
- **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 3-171 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 3-172 Container network configuration list



----End

Using the CCE Console to Create a Container Network Configuration of the Workload Type

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.

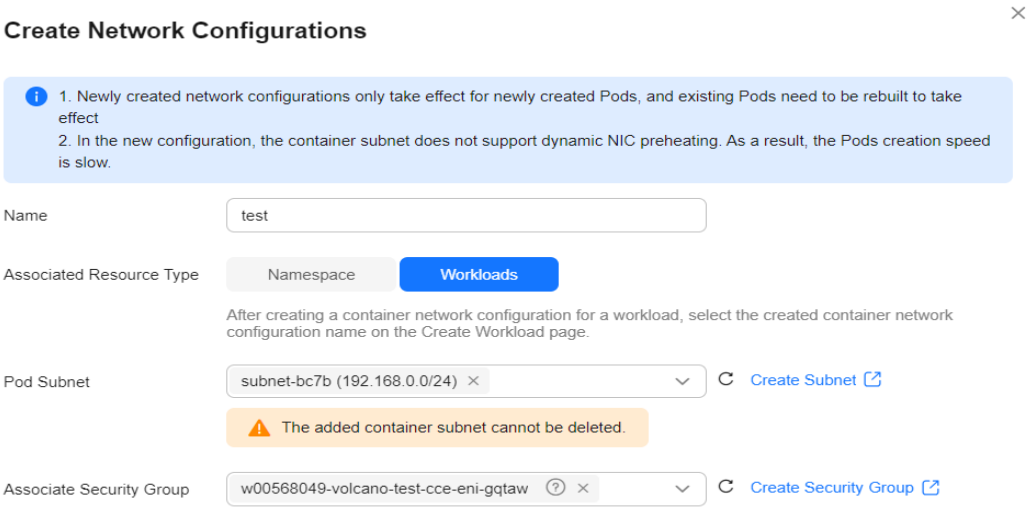
NOTE

If **default-network** is available in the 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**.

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.

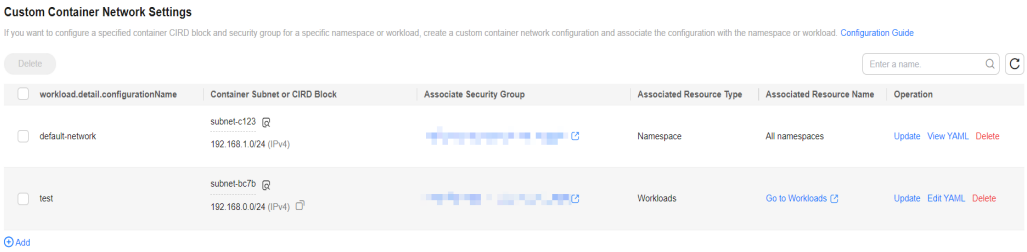
- **Name**: Enter a name that contains a maximum of 253 characters. Do not use **default-network**, **default**, **mgnt0**, or **mgnt1**.
- **Associated Resource Type**: resource type associated with the custom container network configuration. For details, see **Table 3-221**. To create a container network 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. A maximum of 20 subnets can be selected.
- **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 3-173 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.

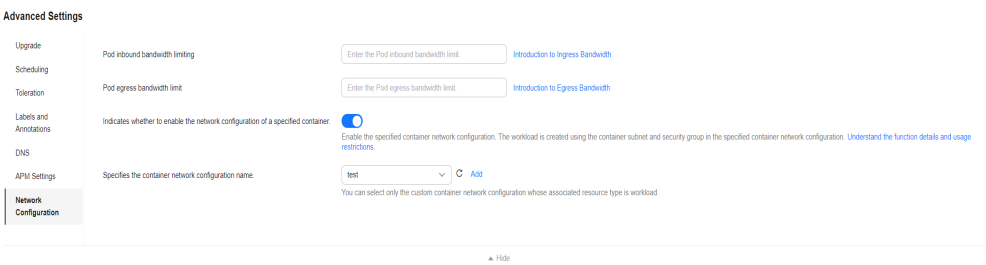
Figure 3-174 Container network configuration list



Step 5 When creating a workload, you can select a custom container network configuration.

1. In the navigation pane, choose **Workloads**. 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.

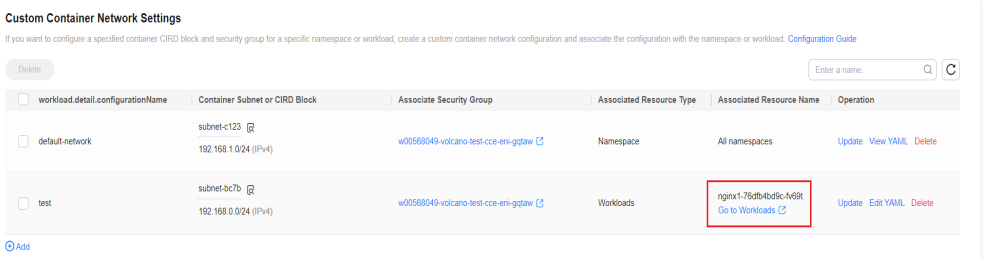
Figure 3-175 Selecting a container network configuration



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 3-176 Resource associated with a container network configuration



----End

Using Kubectl to Create a Container Network Configuration of the Namespace Type

This section describes how to use kubectl to create a container network configuration of the namespace type.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Step 2 Modify the **networkattachment-test.yaml** file.

vi networkattachment-test.yaml

```
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 3-222 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.
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to **kube-system**.
config	Yes	**Table 3-223** object	Configuration content, which is a string in JSON format.

Table 3-223 config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at **eni-neutron**.
args	No	**Table 3-224** object	Configuration parameters.
selector	No	**Table 3-225** object	Namespace on which the configuration takes effect.

Table 3-224 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 3-225 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 a NetworkAttachmentDefinition.

kubectl create -f networkattachment-test.yaml

If information similar to the following is displayed, the NetworkAttachmentDefinition has been created.


```
networkattachmentdefinition.k8s.cni.cncf.io/example created
```

----End

Using Kubectl to Create a Container Network Configuration of the Workload Type

This section describes how to use kubectl to create a container network configuration of the workload type.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Step 2 Modify the **networkattachment-test.yaml** file.

vi networkattachment-test.yaml

```
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**"
        }
      ]
    }'
```

Table 3-226 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.
name	Yes	String	Configuration item name.
namespace	Yes	String	Namespace of the configuration resource. The value is fixed to **kube-system**.
config	Yes	**Table 3-223** object	Configuration content, which is a string in JSON format.

Table 3-227 config parameters

Parameter	Mandatory	Type	Description
type	Yes	String	The value is fixed at **eni-neutron**.
args	No	**Table 3-224** object	Configuration parameters.

Table 3-228 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.

Step 3 Create a NetworkAttachmentDefinition.

kubectl create -f networkattachment-test.yaml

If information similar to the following is displayed, the NetworkAttachmentDefinition 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

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 po -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.

3.7.6.4.3 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.

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 3-229 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**
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**

3.7.6.4.4 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 is automatically allocated and bound to the pod.

Constraints

- You can configure an 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
- 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 EIP name, deleting, unbinding, or binding the EIP, and changing the billing mode of 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

When creating a pod, set the **pod-with-eip** annotation to **true**. An EIP will be automatically allocated and bound to the pod.

The following uses a Deployment named **nginx** as an example. For details about annotations, see **Table 3-231**.

- 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 3-230 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 3-231 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 3-230. • Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	None

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']
...
```

Deleting an EIP with a Pod

When you delete a pod, the EIP automatically allocated to the pod will also be deleted.

3.7.6.4.5 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 **3.7.6.4.4 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 3-232**.

- 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 3-232 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 3-233 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.
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

Annotation	Mandatory	Default Value	Description	Value Range
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

Table 3-234 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 3-230**. • 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.

3.7.7 Cluster Network Settings

3.7.7.1 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.

Constraints

Only CCE standard clusters and CCE Turbo clusters of v1.21 and later are supported.

Planning a Secondary CIDR Block

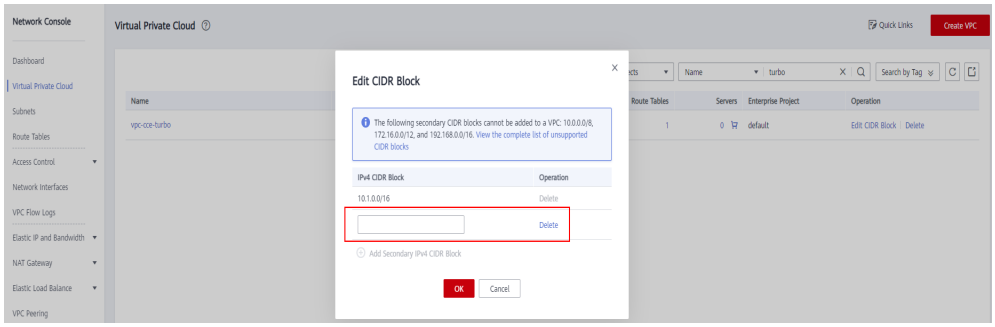
Before adding a secondary CIDR block, plan it properly to prevent CIDR conflicts. Note the following points:

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.

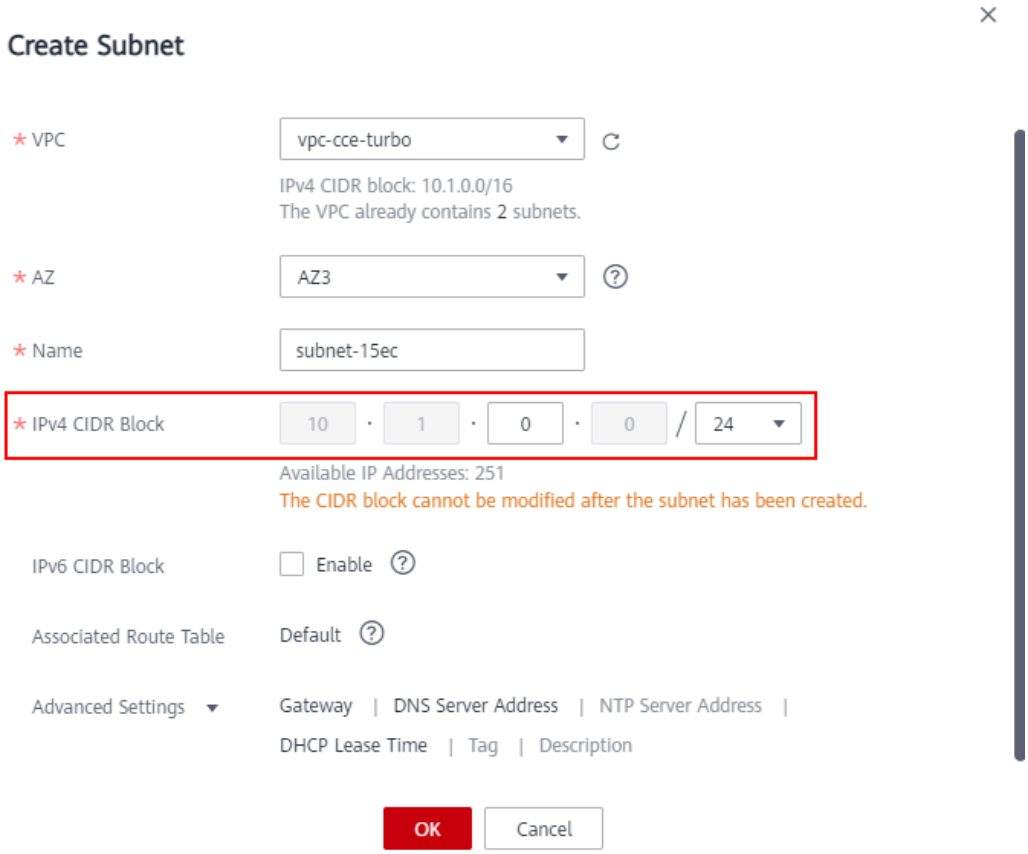
Procedure

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**.

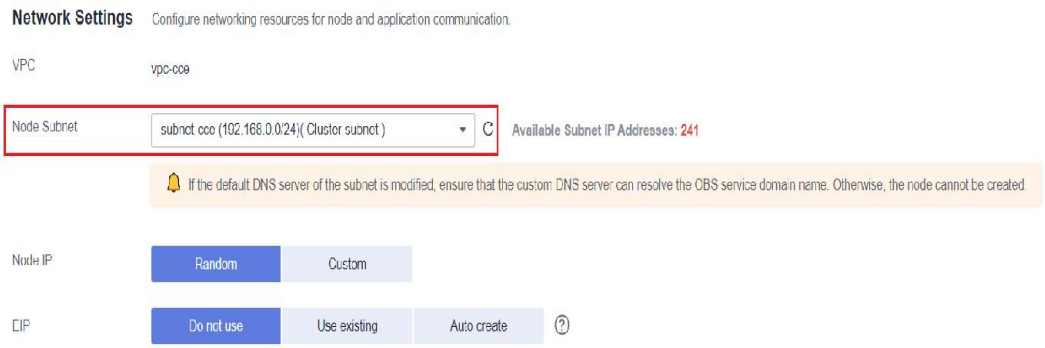
Figure 3-177 Adding a secondary IPv4 CIDR block



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, you can select the subnet when creating a node or node pool in the **Network Settings** page.



----End

3.7.7.2 Switching a Node Subnet

Scenario

This section describes how to switch subnets for nodes in a cluster.

Constraints

- Only subnets in the same VPC as the cluster can be switched. The security group of the node cannot be switched.

- When switching the subnet of a node, comply with the constraints on **Changing a VPC**.

Procedure

Step 1 Log in to the ECS console.

Step 2 Click **More > Manage Network > Change VPC** in the **Operation** column of the target ECS.

Step 3 Set parameters for changing the VPC.

- **VPC**: Select the same VPC as that of the cluster.
- **Subnet**: Select the target subnet to be switched.
- **Private IP Address**: Select **Assign new** or **Use existing** as required.
- **Security Group**: Select the security group of the cluster node. Otherwise, the node is unavailable.

Step 4 Click **OK**.

Step 5 Go to the CCE console and reset the node. You can use the default parameter settings. For details, see **Resetting a Node**.

----End

3.7.7.3 Adding a Container CIDR Block for a Cluster

Scenario

If the container CIDR block (container subnet in a CCE Turbo cluster) set during CCE cluster creation is insufficient, you can add a container CIDR block for the cluster.

Constraints

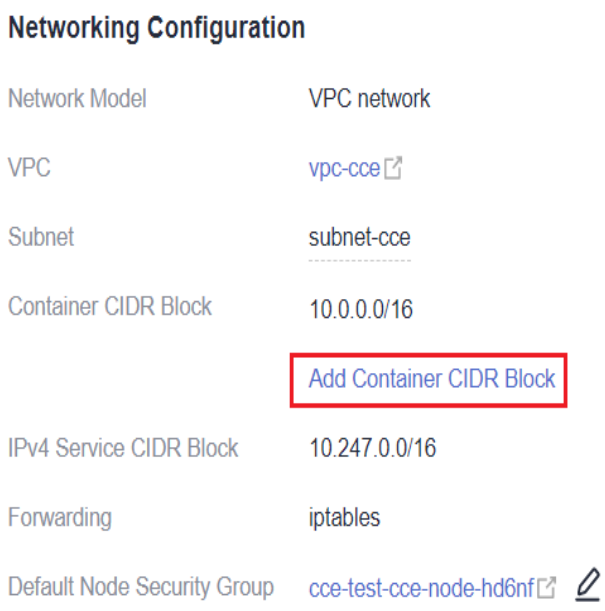
- This function applies to CCE standard clusters and CCE Turbo clusters of v1.19 or later, but not to clusters using container tunnel networking.
- The container CIDR block or container subnet cannot be deleted after being added. Exercise caution when performing this operation.

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 Container CIDR Block**.

Figure 3-178 Adding container CIDR block

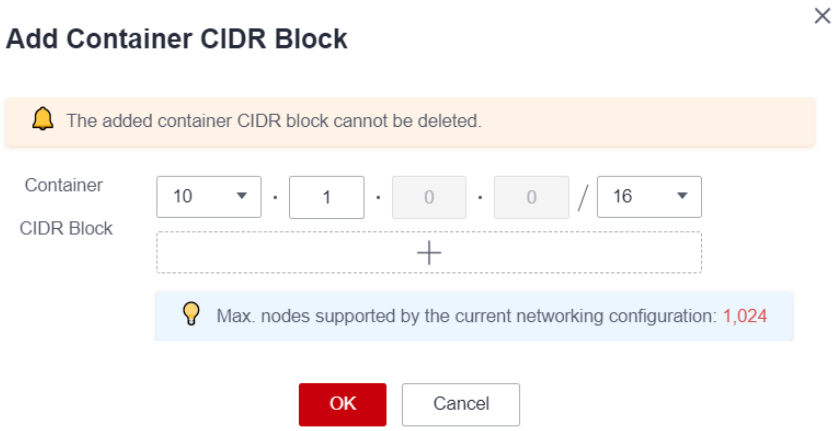


Step 3 Configure the container CIDR block to be added. You can click + to add multiple container CIDR blocks at a time.

NOTE

New container CIDR blocks cannot conflict with service CIDR blocks, VPC CIDR blocks, and existing container CIDR blocks.

Figure 3-179 Configuring the container CIDR block



Step 4 Click **OK**.

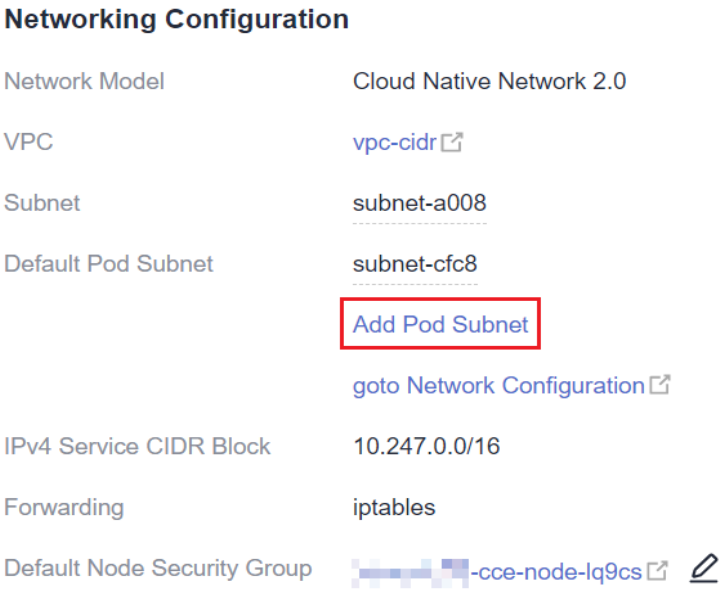
----End

Adding a Container Subnet for a CCE Turbo Cluster

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

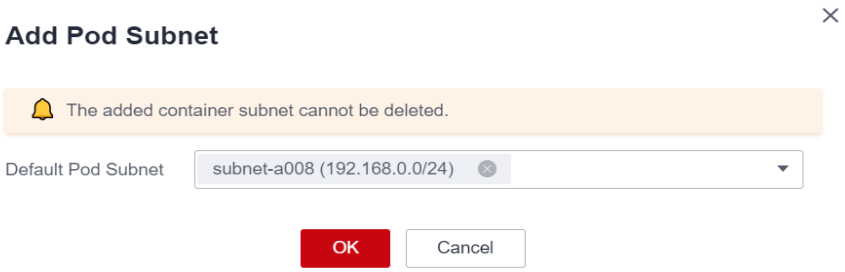
Step 2 On the **Overview** page, locate the **Networking Configuration** area and click **Add Pod Subnet**.

Figure 3-180 Adding a pod subnet



Step 3 Select a container subnet in the same VPC. You can add multiple container subnets at a time. If no other container subnet is available, go to the VPC console to create one.

Figure 3-181 Selecting a pod subnet



Step 4 Click **OK**.

----End

3.7.8 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 the peer resource allows access from the node where the container is located.

- **Cloud Native Network 2.0**

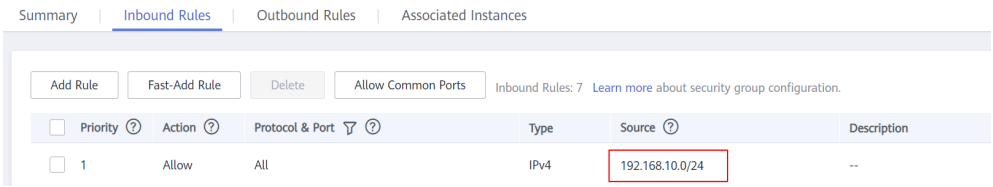
  In the Cloud Native Network 2.0 model, a container is assigned an IP address from the CIDR block of a VPC. The container CIDR block is the subnet of the VPC where the node is located. The container can naturally communicate with other addresses in the VPC. If the access fails, check whether the security 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.

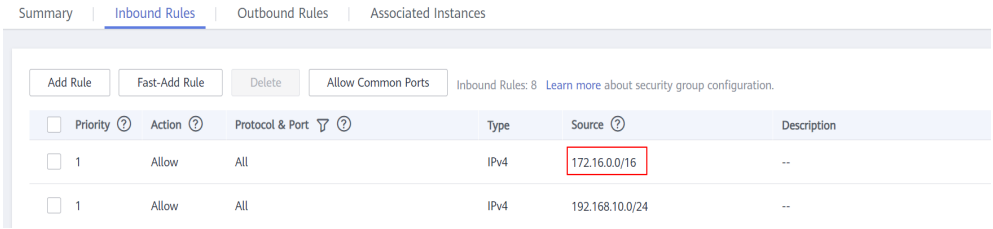
  There is an ECS whose IP address is 192.168.10.52 in the VPC (outside the cluster). The security group of the ECS allows access of only the CIDR block of the cluster node.



In this case, if you ping 192.168.10.52 from the container, the ping operation fails.

```
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.

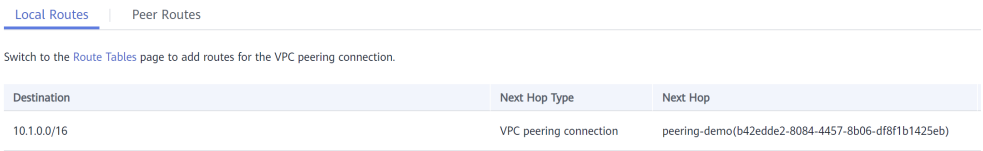
- In the container tunnel network model, a container can access the peer VPC only when the communication is enabled between the node network and the peer VPC.
- Cloud Native Network 2.0 is similar to the container tunnel network. You only need to enable the communication between the subnet where the 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.

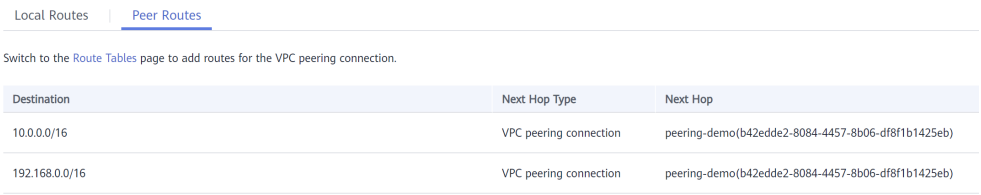
  Assume that there are two VPCs.

  – 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, RabbitMQ, and ModelArts.

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 whitelist. 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 whitelist.

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 whitelist is configured for the peer server. For example, the DCS Redis instance can be accessed only by the IP addresses in its whitelist. Add the container and node CIDR blocks to the whitelist.
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.

3.7.9 Accessing the Internet from a Container

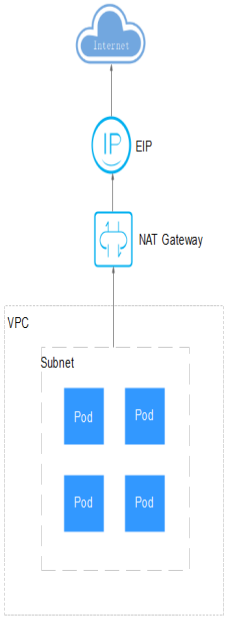
Containers can access the Internet in either of the following ways:

- 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 3-182** 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 3-182 SNAT



To enable a container pod to access the Internet, perform the following steps:

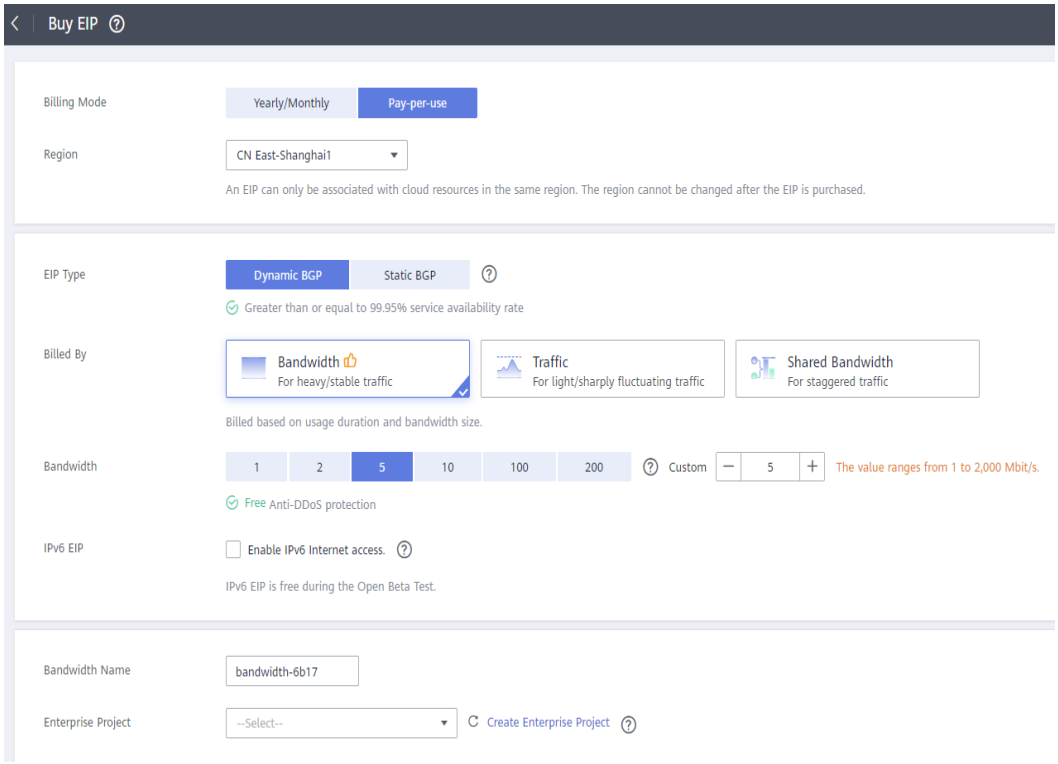
Step 1 Assign an EIP.

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 3-183 Buying an elastic IP address



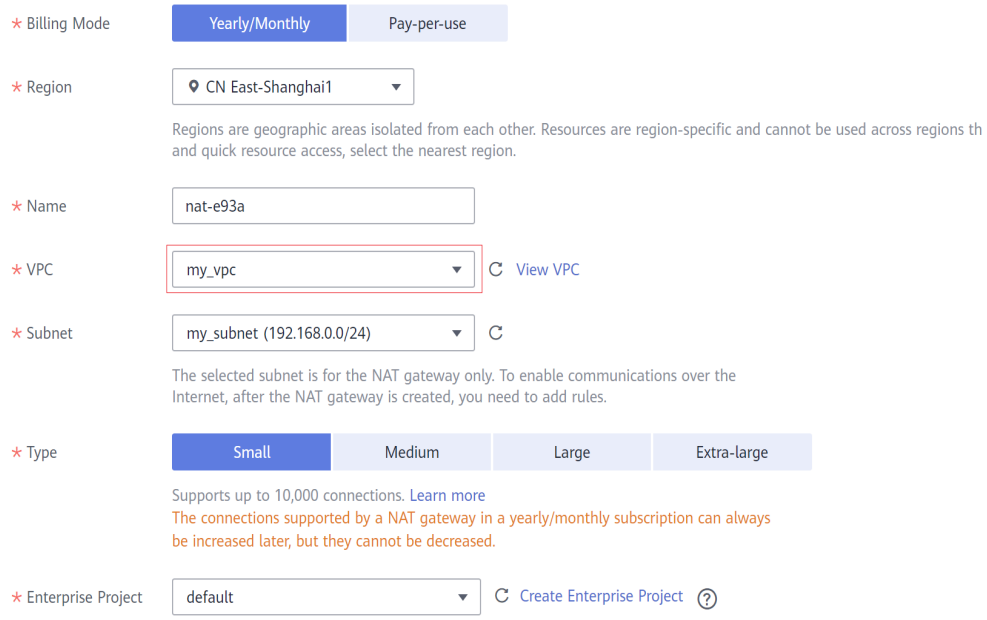
Step 2 Create a NAT gateway. For details, see **Buying a Public NAT Gateway**.

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** > **NAT Gateway** in the expanded list.
4. On the **Public Network Gateways** page, click **Buy Public NAT Gateway** in the upper right corner.
5. Configure parameters as required.

   NOTE

   Select the same VPC.

Figure 3-184 Buying a NAT gateway



Step 3 Configure an SNAT rule and bind the EIP to the subnet. For details, see **Add an SNAT Rule**.

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** > **NAT Gateway** in the expanded list.
4. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
5. On the **SNAT Rules** tab page, click **Add SNAT Rule**.
6. Set parameters as required.

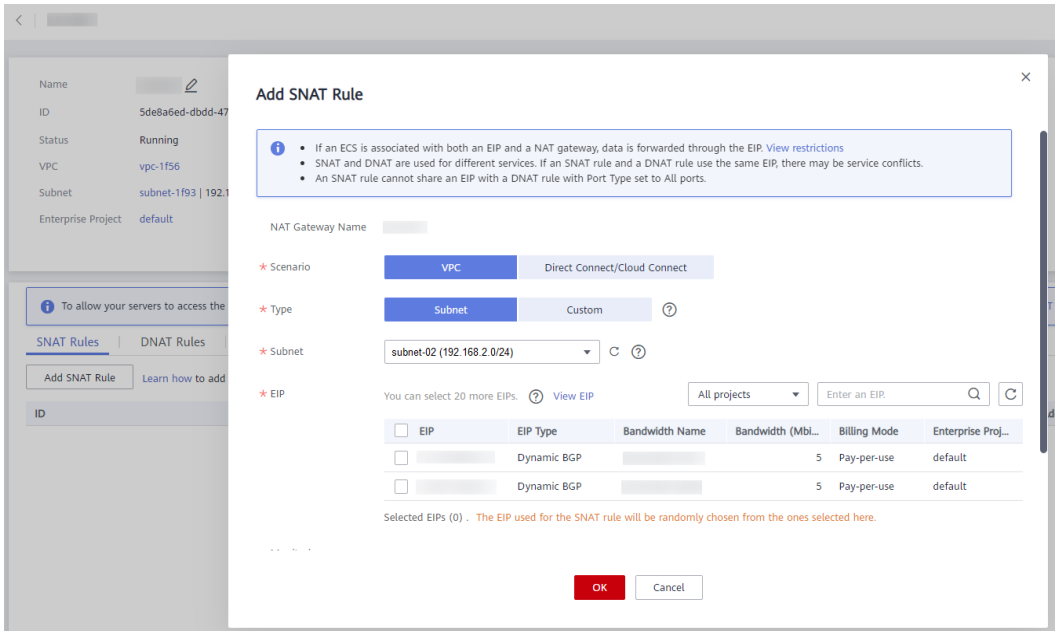
NOTE

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 3-185 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

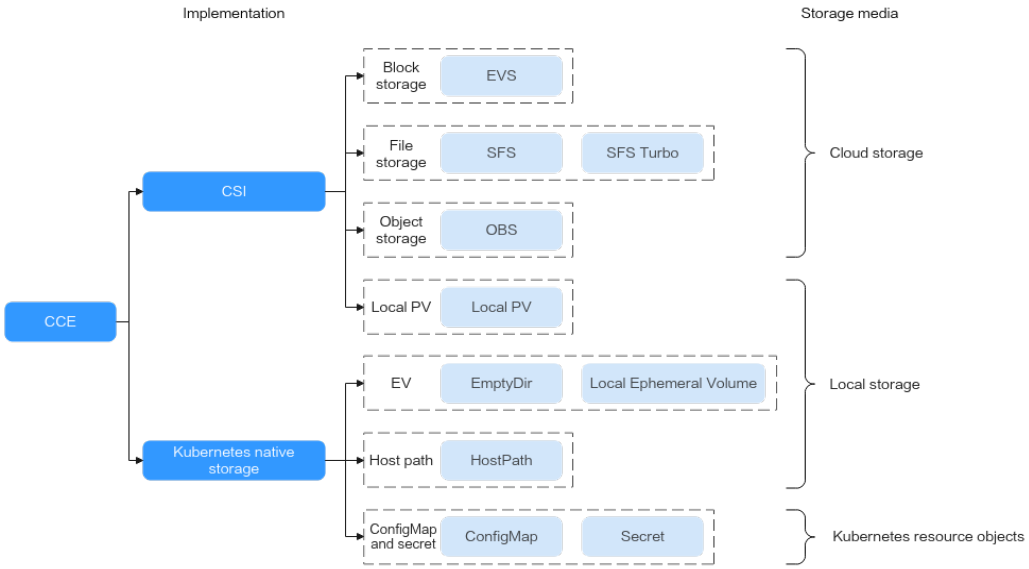
3.8 Storage

3.8.1 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 3-186 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**.

Type	Description	Application Scenario
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
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.

Item	EVS	SFS	SFS Turbo	OBS
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.
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.
Static storage volumes	Supported. For details, see **3.8.3.2 Using an Existing EVS Disk Through a Static PV**.	Supported. For details, see **3.8.4.2 Using an Existing SFS File System Through a Static PV**.	Supported. For details, see **3.8.5.2 Using an Existing SFS Turbo File System Through a Static PV**.	Supported. For details, see **3.8.6.2 Using an Existing OBS Bucket Through a Static PV**.
Dynamic storage volumes	Supported. For details, see **3.8.3.3 Using an EVS Disk Through a Dynamic PV**.	Supported. For details, see **3.8.4.3 Using an SFS File System Through a Dynamic PV**.	Not supported	Supported. For details, see **3.8.6.3 Using an OBS Bucket 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

Item	EVS	SFS	SFS Turbo	OBS
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)
Capacity	TB	SFS 1.0: PB	General-purpose: TB	EB
Latency	1–2 ms	SFS 1.0: 3–20 ms	General-purpose: 1–5 ms	10 ms
Max. IOPS	2200–256000, depending on flavors	SFS 1.0: 2000	General-purpose: up to 100,000	Tens of millions
Bandwidth	MB/s	SFS 1.0: GB/s	General-purpose: up to GB/s	TB/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. **3.8.7.3 Using a Local PV Through a Dynamic PV** is supported.	For details, see **3.8.8.3 Using a Local EV**.	For details, see **3.8.8.4 Using a Temporary Path**.	For details, see **3.8.9 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:

  CCE allows you to specify an enterprise project when creating EVS disks and OBS PVCs. The created storage resources (EVS disks and OBS) belong to the specified enterprise project. **The enterprise project can be the enterprise project to which the cluster belongs or the default enterprise project.**

  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 **Specifying an Enterprise Project for Storage Classes**. 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

- 3.8.2 Storage Basics
- 3.8.3 Elastic Volume Service
- 3.8.4 Scalable File Service
- 3.8.5 SFS Turbo
- 3.8.6 Object Storage Service

3.8.2 Storage Basics

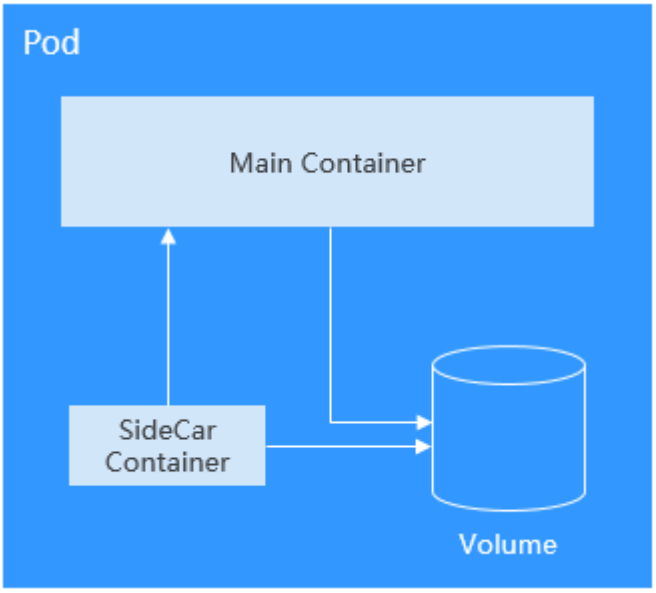
Volumes

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.

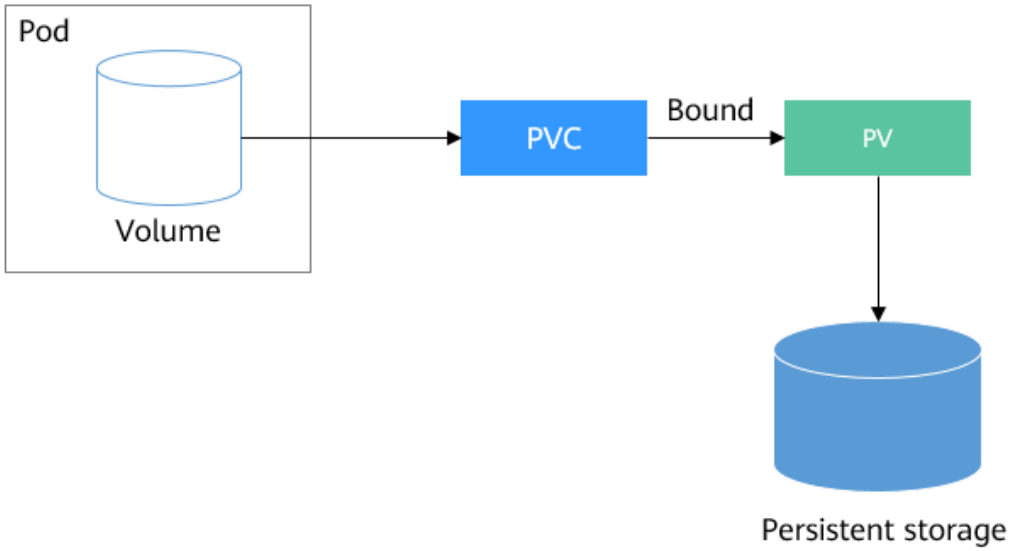
PV and PVC

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 3-187 PVC-to-PV binding



CSI

CSI is a standard for container storage interfaces and a storage plugin implementation solution recommended by the Kubernetes community. **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 3-235 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

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 3-236 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, 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 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 **3.8.1 Overview**.

3.8.3 Elastic Volume Service

3.8.3.1 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 3-237 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

Parameter	Extreme SSD	General Purpose SSD	Ultra-high I/O	High I/O
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:

- **3.8.3.2 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.
- **3.8.3.3 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.
- **3.8.3.4 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**.

3.8.3.2 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 **3.14.3 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 **3.2.3.1 Connecting to a Cluster Using kubectl**.

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 **3.8.3.3 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 3-238**. For details about other parameters, see **3.5 Workloads**.

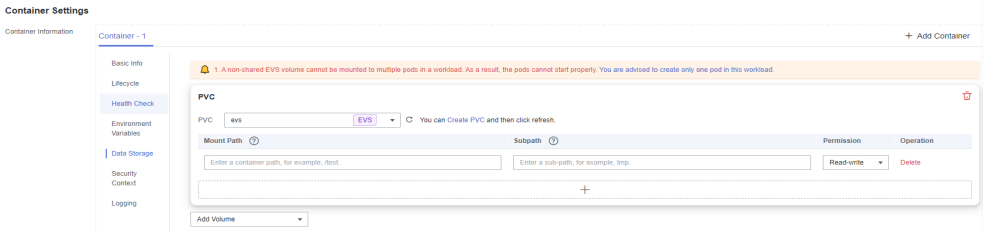
Table 3-238 Mounting a storage volume

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**. 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.

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.

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

  persistentVolumeReclaimPolicy: Delete    # Reclaim policy
  storageClassName: csi-disk            # StorageClass name. The value must be csi-disk for EVS disks.
```


Table 3-239 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 the value of **region**, 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 the value of **zone**, 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 disks ranges from 3000 to 128,000, 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
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	The storage class for EVS disks is **csi-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.
  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   # StorageClass is EVS.
volumeName: pv-evs           # PV name
```

Table 3-240 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of **region**, 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 the value of **zone**, see **Regions and Endpoints**.
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	Storage class name, which must be the same as the storage class of the PV in **1**. The storage class for EVS disks 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 view 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 view 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 3-241**.

Table 3-241 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.

3.8.3.3 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 **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

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.

(Console) Automatically Creating an EVS Disk

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 **3.8.3.2 Using an Existing EVS Disk Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The storage class for EVS disks is **csi-disk**.
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.

Parameter	Description
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.
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. The predefined tags are available to all resources that support tags. You can use predefined 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 3-242**. For details about other parameters, see **3.5 Workloads**.

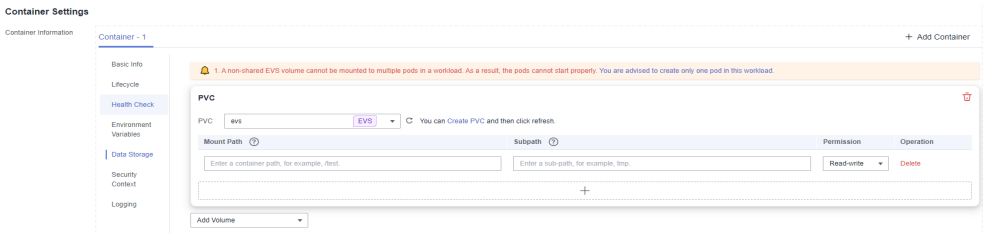
Table 3-242 Mounting a storage volume

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**. 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.

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-iops: '3000'      # (Optional) IOPS of only a GPSSD2 EVS disk
    everest.io/disk-throughput: '125' # (Optional) Throughput of only a GPSSD2 EVS disk
  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    # StorageClass is EVS.
```

Table 3-243 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of **region**, 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 the value of **zone**, see **Regions and Endpoints**.

Parameter	Mandatory	Description
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 disks ranges from 3000 to 128,000, 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	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.

Parameter	Mandatory	Description
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.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	The storage class for EVS disks is **csi-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 view 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 view 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 3-244**.

Table 3-244 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).
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.

3.8.3.4 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.

Prerequisites

- You have created a cluster and installed the **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 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 storage class for EVS disks is **csi-disk**.
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.

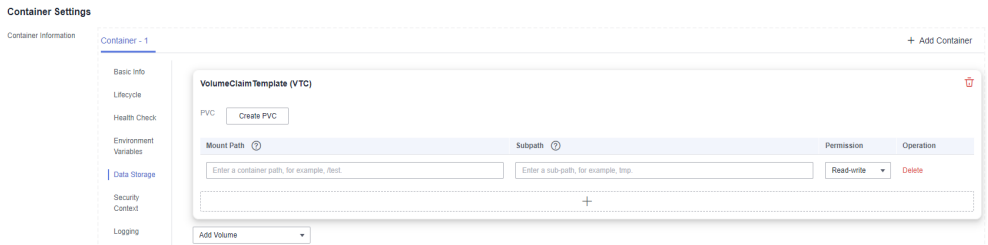
Parameter	Description
Disk Type	Select an EVS disk type. EVS disk types vary depending on regions. Obtain the available EVS types on the console. **NOTE** If the Everest version is 2.4.4 or later, general-purpose SSD V2 EVS disks are supported. General-purpose SSD V2 disks allow you to specify the disk IOPS and throughput. For details, see **EVS performance data**.
Capacity (GiB)	Capacity of the requested storage volume.
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. The predefined tags are available to all resources that support tags. You can use predefined 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 3-245 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**. 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.

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 **3.5.2.2 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-iops: '3000'     # (Optional) IOPS of only a GPSSD2 EVS disk
          everest.io/disk-throughput: '125' # (Optional) Throughput of only a GPSSD2 EVS disk
        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    # 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 3-246 Key parameters

Parameter	Mandatory	Description
failure-domain.beta.kubernetes.io/region	Yes	Region where the cluster is located. For details about the value of **region**, 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 the value of **zone**, 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 disks ranges from 3000 to 128,000, 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	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.
storage	Yes	Requested PVC capacity, in Gi. The value ranges from **1** to **32768**.
storageClassName	Yes	The storage class for EVS disks is **csi-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 view 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 view 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 **web-evs-auto-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 3-247**.

Table 3-247 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).
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.

3.8.3.5 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 (SCSI or VBD), 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 queried or set.

- 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:
  finalizers:
    - snapshot.storage.kubernetes.io/volumesnapshot-as-source-protection
    - snapshot.storage.kubernetes.io/volumesnapshot-bound-protection
  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. The predefined tags are available to all resources that support tags. You can use predefined 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.


  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
```

3.8.4 Scalable File Service

3.8.4.1 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 Capacity-Oriented file systems. For details about file system types, see **File System Type**.

NOTE

- SFS Capacity-Oriented file systems are sold out and cannot be created on the CCE console. You can still create PVs for existing SFS Capacity-Oriented file systems using **kubectl**.

Table 3-248 Performance

Parameter	SFS Capacity-Oriented
Maximum bandwidth	2 GB/s
Maximum IOPS	2000
Latency	3–20 ms
Maximum capacity	4 PB

Application Scenarios

SFS supports the following mounting modes based on application scenarios:

- **3.8.4.2 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.
- **3.8.4.3 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**.

3.8.4.2 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 **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

- You have created an SFS file system that is in the same VPC as the cluster.

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.

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**.
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 **3.8.4.3 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** Currently, only SFS 3.0 Capacity-Oriented is supported.

Parameter	Description
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.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **3.8.4.4 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**.

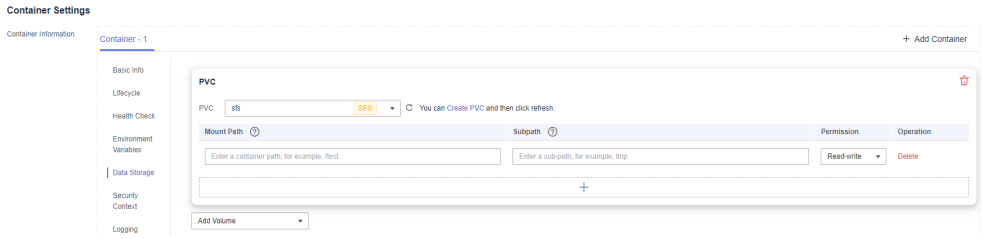
   Mount and use storage volumes, as shown in **Table 3-249**. For details about other parameters, see **3.5 Workloads**.

Table 3-249 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS 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**. 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.

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

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfs.yaml** file.

   SFS Capacity-Oriented:

```
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 3-250 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.
volumeHandle	Yes	– If an SFS Capacity-Oriented volume is used, enter the volume ID. 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**.

Parameter	Mandatory	Description
everest.io/share-export-location	Yes	Shared path of the file system. – For an SFS Capacity-Oriented file system, log in to the CCE console, choose **Service List** > **Storage** > **Scalable File Service**, and obtain the shared path from the **Mount Address** column.
mountOptions	Yes	Mount options. If not specified, the following configurations are used by default. For details, see **3.8.4.4 Configuring SFS Volume 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. 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. **Delete**: – If **everest.io/reclaim-policy** is not specified, both the PV and SFS volume 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 SFS 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	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.

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    # Storage class name, which must be the same as the PV's storage class.
  volumeName: pv-sfs    # PV name
```

Table 3-251 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.
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 view 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 3-252**.

Table 3-252 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**. • 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. • **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **3.8.4.4 Configuring SFS Volume Mount Options**. 2. Click **Create**.
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.

3.8.4.3 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 **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.
- You have created an SFS file system that is in the same VPC as the cluster.

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 **3.8.4.2 Using an Existing SFS File System Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The storage class for SFS volumes is **csi-sfs**.
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 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.

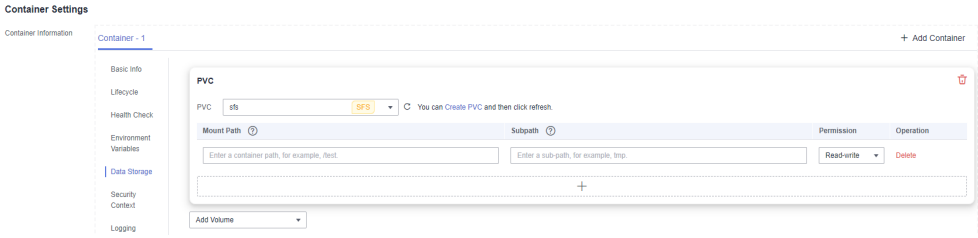
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 3-253**. For details about other parameters, see **3.5 Workloads**.

Table 3-253 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**. 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.

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.
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for SFS.
  resources:
    requests:
      storage: 1Gi              # SFS volume capacity
  storageClassName: csi-nas     # StorageClass is SFS.
```

Table 3-254 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
everest.io/crypt-key-id	No	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.

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 view 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 3-255**.

Table 3-255 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.

3.8.4.4 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 **3.14.3 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.

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 3-256** for mounting SFS volumes.

Table 3-256 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.

Parameter	Value	Description
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 set 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 connect to the cluster. For details, see **3.2.3.1 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 **3.8.4.2 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 connect to the cluster. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Step 2 Create a customized 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 **3.8.4.3 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

3.8.5 SFS Turbo

3.8.5.1 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:

- **3.8.5.2 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.
- **3.8.5.5 Using StorageClass to Dynamically Create a Subdirectory in an SFS Turbo File System**: 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 **Billing**.

3.8.5.2 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 **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 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.

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.
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**.
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**.

Parameter	Description
Reclaim Policy[b]	Only **Retain** is available. This indicates that the PV is not deleted when the PVC is deleted. For details, see **PV Reclaim Policy**.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **3.8.5.3 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**.

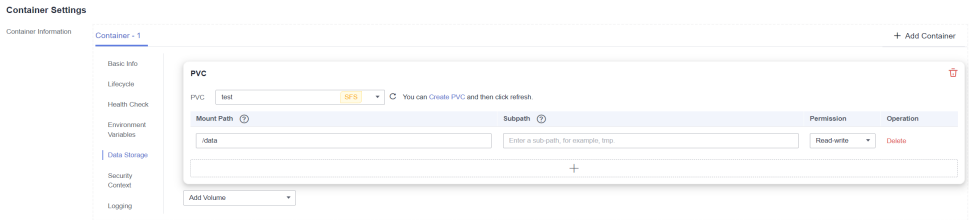
   Mount and use storage volumes, as shown in **Table 3-257**. For details about other parameters, see **3.5 Workloads**.

Table 3-257 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**. 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.

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 File System Through kubectl

Step 1 Use kubectl to access the cluster.

Step 2 Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

```
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          # Storage class name of the SFS Turbo file system
mountOptions: []                        # Mount options
```

Table 3-258 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	SFS Turbo volume 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/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 from the **Mount Address** column.
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click **SFS Turbo** in the left 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 **3.8.5.3 Configuring SFS Turbo 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. Only the **Retain** reclaim policy is supported. 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 storage class 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      # Storage class name of the SFS Turbo file system, which must be the same as that of the PV
  volumeName: pv-sfsturbo    # PV name
```

Table 3-259 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, click **SFS Turbo** in the left 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.

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	Storage class name, which must be the same as the storage class of the PV in **1**. The storage class 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 view 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 3-260**.

Table 3-260 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**. • **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. For details, see **PV Reclaim Policy**. • **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **3.8.5.3 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).

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.

3.8.5.3 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 **3.14.3 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.

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 3-261** for mounting SFS Turbo volumes.

Table 3-261 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.

Parameter	Value	Description
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 set 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 connect to the cluster. For details, see **3.2.3.1 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          # Storage class 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 **3.8.5.2 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

3.8.5.4 Using SFS Turbo Subdirectories Through Dynamic Storage Volumes

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. To properly use the storage capacity, CCE allows you to dynamically create an SFS Turbo subdirectory when creating a PVC so that different workloads can share one SFS Turbo volume.

Prerequisites

- You have created a cluster and installed the **3.14.3 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 **3.2.3.1 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**.
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**.
Subdirectory Reclaim Policy	Whether to retain subdirectories when deleting a PVC. This parameter must be used with **PV Reclaim Policy**. • **Retain**: If the PV reclaim policy associated with a PVC is **Delete**, the PV will be deleted when the PVC is deleted, but the **subdirectories associated with the PV will be retained**. • **Delete**: If the PV reclaim policy associated with a PVC is **Delete**, the **PV and the subdirectories associated with the PV will be deleted** when the PVC is 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      # This parameter is only used for verification for the PVCs of the SFS Turbo
subdirectory type. The value cannot be empty or 0.
  storageClassName: csi-sfsturbo     # Storage class name of the SFS Turbo file system
```

Table 3-262 Key parameters

Parameter	Mandatory	Description
everest.io/volume-as	No	When a dynamically created SFS Turbo subdirectory is used, the value of this parameter is consistently **absolute-path**.
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**.
storage	Yes	Requested capacity in the PVC, in Gi. This parameter is only used for verification for the PVCs of the SFS Turbo subdirectory type. 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

3.8.5.5 Using StorageClass to Dynamically Create a Subdirectory in an SFS Turbo File System

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.

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
mountOptions:
- 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/
  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
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
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: sfsturbo-subpath-sc
  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
  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
      name: container-0
      volumeMounts:
      - mountPath: /tmp
        name: pvc-sfs-turbo-example
    restartPolicy: Always
    imagePullSecrets:
    - name: default-secret
    volumes:
    - name: pvc-sfs-turbo-example
      persistentVolumeClaim:
        claimName: sfs-turbo-test
```

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
  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'
          resources: {}
          volumeMounts:
            - name: sfs-turbo-160024548582479676
              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: sfs-turbo-160024548582479676
      namespace: default
      annotations: {}
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
      storageClassName: sfsturbo-subpath-sc
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

3.8.6 Object Storage Service

3.8.6.1 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.

- Parallel File System (PFS, **recommended**): It is an optimized high-performance file system provided by OBS. It provides millisecond-level access latency, TB/s-level bandwidth, and million-level IOPS, and can quickly process HPC workloads. PFS outperforms OBS buckets. For details, see **About Parallel File System**.
- Object bucket (**not recommended**):
  - 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.

For details about OBS storage classes, see **Storage Classes**.

Application Scenarios

OBS supports the following mounting modes based on application scenarios:

- **3.8.6.2 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.
- **3.8.6.3 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**.

3.8.6.2 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.

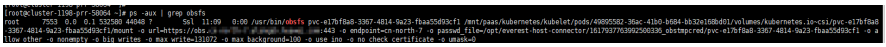
Prerequisites

- You have created a cluster and installed the **3.14.3 CCE Container Storage (Everest)** add-on in the cluster.
- To create a cluster using commands, ensure kubectl is used. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- CCE allows parallel file systems to be mounted using OBS SDKs or PVCs. If PVC mounting is used, the obsfs tool provided by OBS must be used. An obsfs resident process is generated each time an object storage volume generated from the parallel file system is mounted to a node, as shown in the following figure.

  Figure 3-188 obsfs resident process

  Reserve 1 GiB of memory for each obsfs process. For example, for a node with 4 vCPUs and 8 GiB of memory, an obsfs parallel file system should be mounted to **no more than** eight pods.

  NOTE

  - An obsfs resident process runs on a node. If the consumed memory exceeds the upper limit of the node, the node malfunctions. On a node with 4 vCPUs and 8 GiB of memory, if more than 100 pods are mounted to a parallel file system, the node will be unavailable. Control the number of pods mounted to a parallel file system on a single node.
  - When using obsfs, comply with **obsfs Constraints**.
- Kata containers do not support OBS volumes.
- 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 **3.8.6.3 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 **3.8.6.5 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 **3.8.6.4 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**.

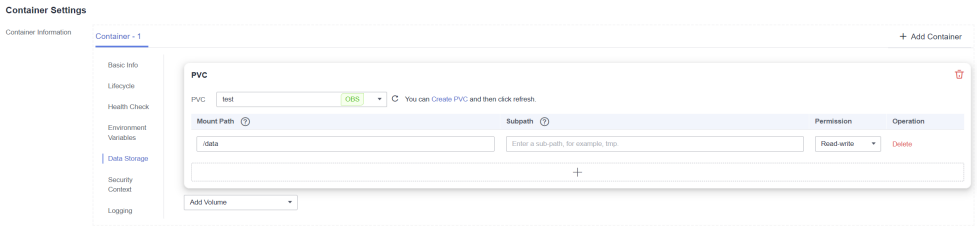
   Mount and use storage volumes, as shown in **Table 3-263**. For details about other parameters, see **3.5 Workloads**.

Table 3-263 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**. 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.

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              # Storage class name
  mountOptions: []                       # Mount options
```

Table 3-264 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**: Parallel file system, which is mounted using obsfs (recommended). – **s3fs**: Object bucket, which is mounted using s3fs.
volumeHandle	Yes	OBS volume name.

Parameter	Mandatory	Description
everest.io/obs-volume-type	Yes	OBS storage class. – If **fsType** is set to **s3fs**, **STANDARD** (standard bucket) and **WARM** (infrequent access bucket) 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 the value of **region**, 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 on the left. 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 **3.8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.
		An example is as follows:
		`nodePublishSecretRef:`
		`  name: secret-demo`
		`  namespace: default`

mountOptions	No	Mount options. For details, see **3.8.6.4 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      # Storage class name, which must be the same as that of the PV.
  volumeName: pv-obs    # PV name
```

Table 3-265 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 on the left. 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	Storage class name, which must be the same as the storage class 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 view 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 3-266**.

Table 3-266 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 storage 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. • **Custom** (Recommended): Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **3.8.6.5 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 **3.8.6.4 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.

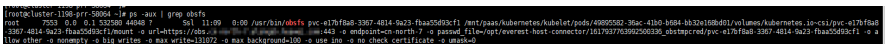
3.8.6.3 Using an OBS Bucket Through a Dynamic PV

This section describes how to automatically create an OBS bucket. It is applicable when no underlying storage volume is available.

Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- CCE allows parallel file systems to be mounted using OBS SDKs or PVCs. If PVC mounting is used, the obsfs tool provided by OBS must be used. An obsfs resident process is generated each time an object storage volume generated from the parallel file system is mounted to a node, as shown in the following figure.

Figure 3-189 obsfs resident process



Reserve 1 GiB of memory for each obsfs process. For example, for a node with 4 vCPUs and 8 GiB of memory, an obsfs parallel file system should be mounted to **no more than** eight pods.

NOTE

- An obsfs resident process runs on a node. If the consumed memory exceeds the upper limit of the node, the node malfunctions. On a node with 4 vCPUs and 8 GiB of memory, if more than 100 pods are mounted to a parallel file system, the node will be unavailable. Control the number of pods mounted to a parallel file system on a single node.
- When using obsfs, comply with **obsfs Constraints**.

- Kata containers do not support OBS volumes.
- 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 **3.8.6.2 Using an Existing OBS Bucket Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	StorageClass name, which is **csi-obs** for an OBS volume.
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. **Parallel file systems are recommended.** – **Object bucket**: a container that stores objects in OBS. All objects in a bucket are at the same logical level.

Parameter	Description
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.
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 **3.8.6.5 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.

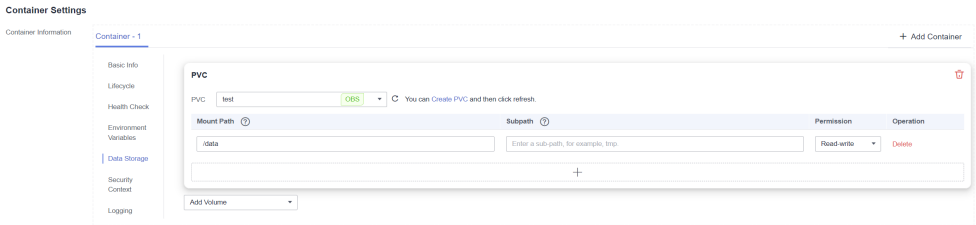
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 3-267**. For details about other parameters, see **3.5 Workloads**.

Table 3-267 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**. 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.

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.

spec:
  accessModes:
    - ReadWriteMany            # The value must be ReadWriteMany for OBS.
  resources:
    requests:
      storage: 1Gi             # OBS volume capacity
  storageClassName: csi-obs    # StorageClass is OBS.
```

Table 3-268 Key parameters

Parameter	Mandatory	Description
everest.io/obs-volume-type	Yes	OBS storage class. – If **fsType** is set to **s3fs**, **STANDARD** (standard bucket) and **WARM** (infrequent access bucket) are supported. – This parameter is invalid when **fsType** is set to **obsfs**.
csi.storage.k8s.io/fstype	Yes	Instance type. The value can be **obsfs** or **s3fs**. – **obsfs**: Parallel file system, which is mounted using obsfs (recommended). – **s3fs**: Object bucket, which is mounted using s3fs.
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 **3.8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.

Parameter	Mandatory	Description
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.
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 view 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 3-269**.

Table 3-269 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).

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.

3.8.6.4 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 **3.14.3 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.

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 3-270** and **Table 3-271** by default. The options in **Table 3-270** are mandatory. You can set other mount options if needed. For details, see **Mounting a Parallel File System**.

Table 3-270 Mandatory mount options configured by default

Parameter	Value	Description
use_ino	Blank	If enabled, obsfs allocates the **inode** number. Enabled by default in read/write mode.
big_writes	Blank	If configured, the maximum size of the cache can be modified.
nonempty	Blank	Allows non-empty mount paths.
allow_other	Blank	Allows other users to access the parallel file system.
no_check_certificate	Blank	Disables server certificate verification.

Parameter	Value	Description
enable_noobj_cache	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	Blank	Specifies the signature version. Used by default in object buckets.
public_bucket	1	If this parameter is set to **1**, public buckets are mounted anonymously. Enabled by default in object bucket read-only mode.

Table 3-271 Optional mount options configured by default

Parameter	Value	Description
max_write	131072	This parameter is valid only when **big_writes** is configured. The recommended value is **128 KB**.
ssl_verify_hostname	0	Disables SSL certificate verification based on the host name.
max_background	100	Allows setting the maximum number of waiting requests in the background. Used by default in parallel file systems.
umask	A three-digit octal number	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 connect to the cluster. For details, see **3.2.3.1 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              # Storage class 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 **3.8.6.2 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 connect to the cluster. For details, see **3.2.3.1 Connecting to a Cluster Using kubectl**.

Step 2 Create a customized 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=0027
```

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 **3.8.6.3 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

3.8.6.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume

Scenario

3.14.3 CCE Container Storage (Everest) of version 1.2.8 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 **How Can I Control Access to OBS?**

Prerequisites

- The **3.14.3 CCE Container Storage (Everest)** version must be 1.2.8 or later.
- The cluster version must be 1.15.11 or later.

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

The key you uploaded is used by default when mounting an OBS volume. That is, all IAM users under your account will use the same key to mount OBS buckets,

and they have the same permissions on buckets. This setting does not allow you to configure differentiated permissions for different IAM users.

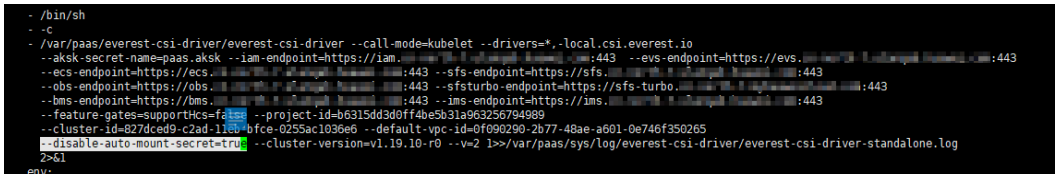
If you have uploaded the AK/SK, disable the automatic mounting of access keys by enabling the **disable_auto_mount_secret** parameter in the Everest add-on to prevent IAM users from performing unauthorized operations. In this way, the access keys uploaded on the console will not be used when creating OBS volumes.

NOTE

- When enabling **disable-auto-mount-secret**, ensure that no OBS volume exists in the cluster. A workload mounted with an OBS volume, when scaled or restarted, will fail to remount the OBS volume because it needs to specify the access key but is prohibited 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, the OBS volume fails to be mounted.

kubectl edit ds everest-csi-driver -nkube-system

Search for **disable-auto-mount-secret** and set it to **true**.



Run **:wq** to save the settings and exit. Wait until the pod is restarted.

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.
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
fsType	File type. The value can be **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. You are advised to set this field to **obsfs**.
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

Parameter	Description
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/

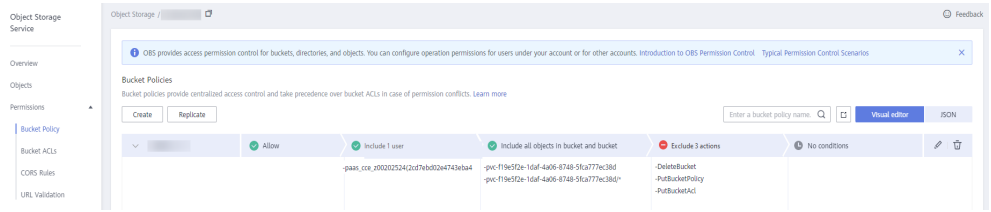
3. Write data into the mount path. In this example, the write operation failed.

   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
   ```

3.8.6.6 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.

Constraints

- The **3.14.3 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 the value of **region**, 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 (recommended), and s3fs indicates an object 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 **3.8.6.5 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                    # OBS bucket type. Currently, standard (STANDARD) and infrequent access (WARM) are supported.
    csi.storage.k8s.io/fstype: s3fs                         # File type. obsfs indicates a parallel file system (recommended), 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

3.8.7 Local PVs

3.8.7.1 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:

- **3.8.7.3 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.
- **3.8.7.4 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.

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.
- **Deleting**, **removing**, **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.

3.8.7.2 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.

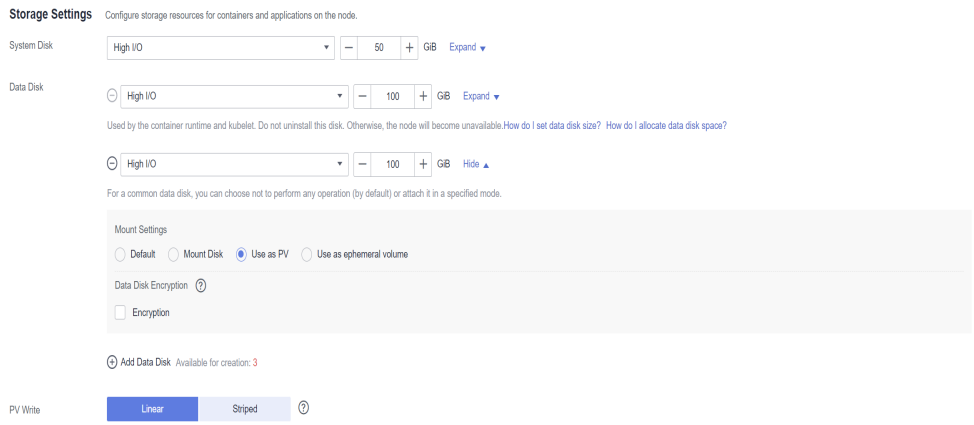
Constraints

- 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 **3.3.4 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.

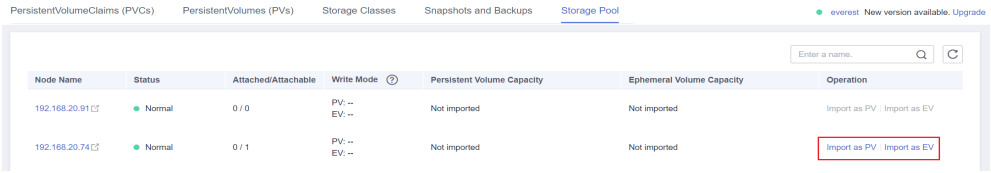
Step 3 Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.

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, allowing data to be concurrently read and written. Select this option only when there are multiple volumes.



----End

3.8.7.3 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 **3.2.3.1 Connecting to a Cluster Using kubectl**.
- You have imported a data disk of a node to the local PV storage pool. For details, see **3.8.7.2 Importing a PV to a Storage Pool**.

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.
- **Deleting**, **removing**, **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 storage class of local PVs is **csi-local-topology**.
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 **3.8.7.2 Importing a PV to a Storage Pool**.
Capacity (GiB)	Capacity of the requested storage volume.

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.

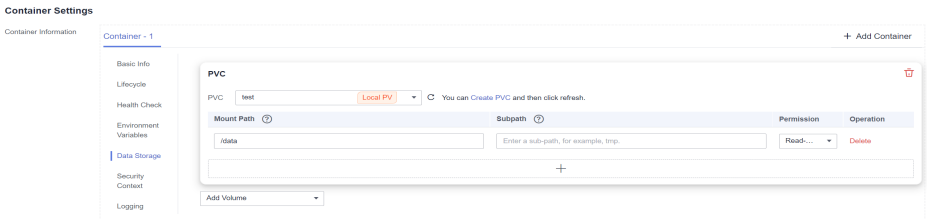
1. Choose **Workloads** in the navigation pane. In the right pane, click the **Deployments** tab.
2. Click **Create Workload** in the upper right corner. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 3-272**. For details about other parameters, see **3.5 Workloads**.

Table 3-272 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**. 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.

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
spec:
  accessModes:
    - ReadWriteOnce             # The value must be ReadWriteOnce for local PVs.
  resources:
    requests:
      storage: 10Gi             # Size of the local PV
  storageClassName: csi-local-topology    # StorageClass is local PV.
```

Table 3-273 Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi.
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 view 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 view 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 3-274**.

Table 3-274 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.

3.8.7.4 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.

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 **3.2.3.1 Connecting to a Cluster Using kubectl**.
- You have imported a data disk of a node to the local PV storage pool. For details, see **3.8.7.2 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**.

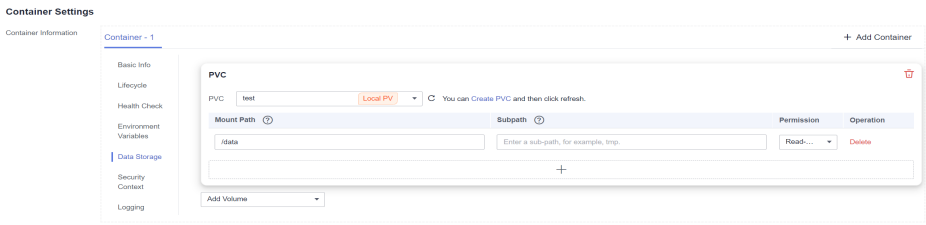
Parameter	Description
PVC Type	In this section, select **Local PV**.
PVC Name	Enter the name of the PVC. After a PVC is created, a suffix is automatically added based on the number of pods. The format is <*Custom PVC name*>-<*Serial number*>, for example, example-0.
Creation Method	You can only select **Dynamically provision** to create a PVC, PV, and underlying storage on the console in cascading mode.
Storage Classes	The storage class of local PVs is **csi-local-topology**.
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 **3.8.7.2 Importing a PV to a Storage Pool**.
Capacity (GiB)	Capacity of the requested storage volume.

Step 5 Enter the path to which the volume is mounted.

Table 3-275 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**. 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.

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 **3.5.2.2 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
      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 3-276 Key parameters

Parameter	Mandatory	Description
storage	Yes	Requested capacity in the PVC, in Gi.
storageClassName	Yes	The storage class 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 view 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 view 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 3-277**.

Table 3-277 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.

3.8.8 Ephemeral Volumes

3.8.8.1 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:

- **3.8.8.4 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.
- **3.8.8.3 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.

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.

3.8.8.2 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.

Constraints

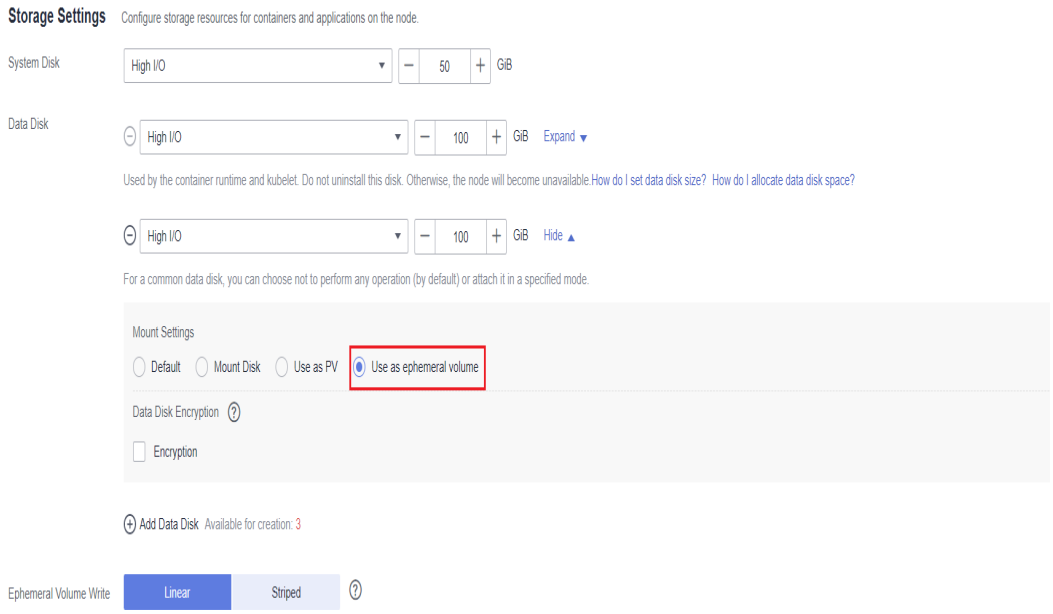
- Local EVs are supported only when the cluster version is v1.21.2-r0 or later and the Everest add-on version is 1.2.29 or later.
- The first data disk (used by container runtime and the kubelet component) on a node cannot be imported as a storage pool.
- Storage pools in striped mode do not support 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 **3.3.4 Creating a Node**.

Figure 3-190 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.

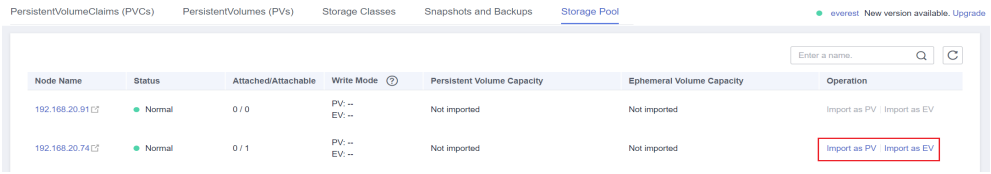
Step 3 Choose **Storage** in the navigation pane. In the right pane, click the **Storage Pool** tab.

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, allowing data to be concurrently read and written. Select this option only when there are multiple volumes.



----End

3.8.8.3 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 **3.2.3.1 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 **3.8.8.2 Importing an EV to a Storage Pool**.

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 3-278**. For details about other parameters, see **3.5 Workloads**.

Table 3-278 Mounting a local EV

Parameter	Description
Capacity	Capacity of the requested 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**. 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

Mounting a Local EV Through kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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.

3.8.8.4 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 3-279**. For details about other parameters, see **3.5 Workloads**.

Table 3-279 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 connect to the cluster. For details, see **3.2.3.1 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

3.8.9 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 3-280**.

Table 3-280 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

3.8.10 StorageClass

Introduction

StorageClass describes the classification of storage types in a cluster and can be represented as a configuration template for creating PVs. When creating a PVC or PV, specify StorageClass.

As a user, you only need to specify **storageClassName** when defining a PVC to automatically create a PV and underlying storage, significantly reducing the workload of creating and maintaining a PV.

In addition to the **default storage classes** provided by CCE, you can also customize storage classes.

- **Application Scenarios of Custom Storage**
- **Custom Storage Class**
- **Specifying a Default Storage Class**
- **Specifying an Enterprise Project for Storage Classes**

CCE Default Storage Classes

As of now, CCE provides storage classes 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 storage classes that CCE supports. Use the CSI add-on provided by CCE to create a storage class.

```
# 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-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 storage class contains the default parameters used for dynamically creating a PV. The following is an example of storage class 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 3-281 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 3-282**.
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**: indicates that a dynamically created PV will be automatically deleted when the PVC is deleted. • **Retain**: indicates that a dynamically created PV will be retained when the PVC is deleted.
allowVolumeExpansion	Specifies whether the PV of this storage class 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**: PV binding and dynamic creation are completed when a PVC is created. • **WaitForFirstConsumer**: PV binding and creation are delayed. The PV creation and binding processes are executed only when the PVC is used in the workload.
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 3-282 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**.

Volume Type	Parameter	Mandatory	Description
	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.
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. You do not need to configure this parameter when SFS 3.0 is used.
	everest.io/sfs-version	No	This parameter is mandatory only when SFS 3.0 is used. The 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**.

Volume Type	Parameter	Mandatory	Description
	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**.
	everest.io/share-volume-type	No	SFS Turbo storage class. 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**: Parallel file system, which is mounted using obsfs (recommended). • **s3fs**: Object bucket, which is mounted using s3fs.
	everest.io/obs-volume-type	Yes	OBS storage class. • If **fsType** is set to **s3fs**, **STANDARD** (standard bucket) and **WARM** (infrequent access bucket) 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
```

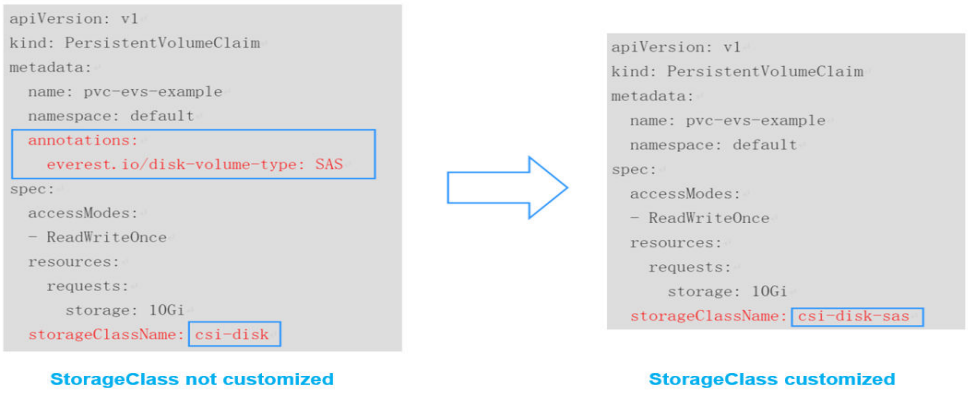
To specify the EVS disk type on CCE, use the **everest.io/disk-volume-type** field. SAS indicates the EVS disk type.

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 storage class, respectively. If a storage class named **csi-disk-sas** is defined, it is used to create SAS storage.	**Custom Storage Class**
When a user migrates services from a self-built Kubernetes cluster or other Kubernetes services to CCE, the storage class 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 storage class with the same name as that in the original application YAML file in the CCE centralization. After the migration, you do not need to modify the **storageClassName** in the application YAML file. For example, the EVS disk storage class used before the migration is **disk-standard**. After migrating services to a CCE cluster, you can copy the YAML file of the **csi-disk** storage class in the CCE cluster, change its name to **disk-standard**, and create another storage class.	
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.	**Specifying a Default Storage Class**

Custom Storage Class

This section uses the custom storage class of EVS disks as an example to describe how to define SAS EVS disk and SSD EVS disk as a storage class, respectively. For example, if you define a storage class named **csi-disk-sas**, which is used to create SAS storage, the differences are shown in the following figure. When compiling a YAML file, you only need to specify **storageClassName**.



- You can customize a high I/O storage class 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 storage class, 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 storage class, 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 storage class, 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
```

Specifying a Default Storage Class

You can specify a storage class as the default class. In this way, if you do not specify **storageClassName** when creating a PVC, the PVC is created using the default storage class.

For example, to specify **csi-disk-ssd** as the default storage class, 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"   # Specifies the default storage class in a cluster. A
cluster can have only one default storage class.
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
```

Delete the created csi-disk-ssd disk, run the **kubectl create** command to create a csi-disk-ssd disk again, and then query the storage class. The following information is displayed.

```
# kubectl delete sc csi-disk-ssd
storageclass.storage.k8s.io "csi-disk-ssd" deleted
# 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
```


Specifying an Enterprise Project for Storage Classes

CCE allows you to specify an enterprise project when creating EVS disks and OBS PVCs. The created storage resources (EVS disks and OBS) belong to the specified enterprise project. **The enterprise project can be the enterprise project to which the cluster belongs or the default enterprise project.**

If you do not specify any enterprise project, the enterprise project in StorageClass is used by default. The created storage resources by using the csi-disk and csi-obs storage classes of CCE belong to the default enterprise project.

If you want the storage resources created from the storage classes to be in the same enterprise project as the cluster, you can customize a storage class and specify the enterprise project ID, as shown below.

NOTE

To use this function, the everest add-on must be upgraded to 1.2.33 or later.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-disk-epid      #Customize a storage class 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
```

Verification

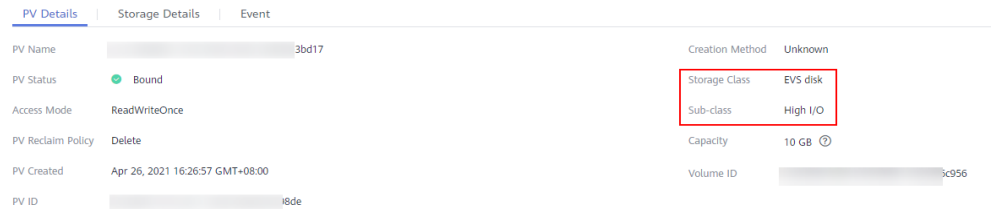
- Use **csi-disk-sas** to create a PVC.

  ```
  apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name:  sas-disk
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
    storageClassName: csi-disk-sas
  ```

  Create a storage class and view its details. As shown below, the object can be created and the value of **STORAGECLASS** is **csi-disk-sas**.

  ```
  # kubectl create -f sas-disk.yaml
  persistentvolumeclaim/sas-disk created
  # kubectl get pvc
  NAME       STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
  sas-disk   Bound    pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO            csi-disk-sas   24s
  # kubectl get pv
  NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
  pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO            Delete           Bound    default/sas-disk      csi-disk-sas            30s
  ```


View the PVC details on the CCE console. On the PV details page, you can see that the disk type is high I/O.



- If **storageClassName** is not specified, the default configuration is used, as shown below.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name:  ssd-disk
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Create and view the storage resource. You can see that the storage class of PVC ssd-disk is csi-disk-ssd, indicating that csi-disk-ssd is used by default.

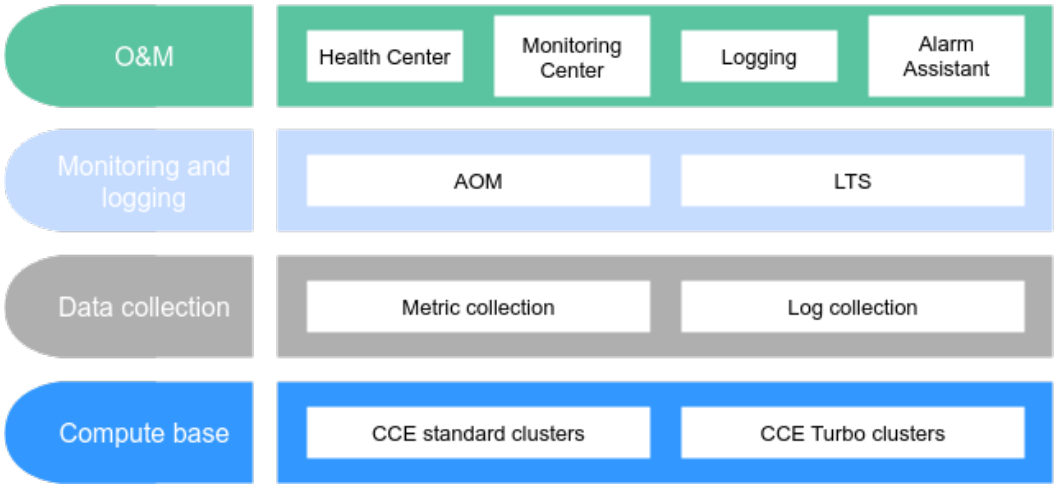
```
# kubectl create -f ssd-disk.yaml
persistentvolumeclaim/ssd-disk created
# kubectl get pvc
NAME       STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
sas-disk   Bound    pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO           csi-disk-sas
16m
ssd-disk   Bound    pvc-4d2b059c-0d6c-44af-9994-f74d01c78731   10Gi       RWO           csi-disk-ssd
10s
# kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
CLAIM                  STORAGECLASS   REASON   AGE
pvc-4d2b059c-0d6c-44af-9994-f74d01c78731   10Gi       RWO            Delete           Bound
default/ssd-disk       csi-disk-ssd            15s
pvc-6e2f37f9-7346-4419-82f7-b42e79f7964c   10Gi       RWO            Delete           Bound       default/
sas-disk       csi-disk-sas            17m
```

View the PVC details on the CCE console. On the PV details page, you can see that the disk type is ultra-high I/O.

3.9 Observability

3.9.1 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.



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 **3.14.17 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 **3.14.18 Cloud Native Logging**.

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.

Cloud Native Observability

CCE provides Health Center, Monitoring Center, Logging, and Alarm Assistant for cloud native observability.

- **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 container insights, health diagnosis, and dashboard. Container insights provides monitoring views from dimensions such as cluster, node, workload, and pod. It supports multi-level drill-down and association analysis. Dashboard gives you monitoring graphs for items such as kube-apiserver, 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 (standard output logs, text logs, and node logs).

- **Alarm Assistant**

  Alarm Assistant works with AOM 2.0 to allow you to create alarm rules and view alarms of clusters and containers.

Resource Permissions

Health Center, Monitoring Center, Logging, and Alarm Assistant work closely with cloud services for cluster monitoring, alarm reporting, and notification. When you access Health Center, Monitoring Center, Logging, or Alarm Assistant for the first time, the system will request permissions to access the cloud services in the region where you run your applications.

The following table lists the permissions.

Assigned To	Permission	Description
CCE	IAM ReadOnlyAccess	IAM users need to access Monitoring Center and Alarm Assistant.
CCE	Tenant Guest	Monitoring Center and Alarm Assistant check the configurations of global resources associated with clusters such as OBS and DNS resources to identify incorrect configurations.
CCE	CCE Administrator	Monitoring Center and Alarm Assistant need to access CCE to obtain information about clusters, nodes, and workloads so that they can help ensure resource health.
CCE	SWR Administrator	Monitoring Center and Alarm Assistant need to access SWR to obtain image information.
CCE	SMN Administrator	Monitoring Center and Alarm Assistant need to access SMN to obtain contact group information.

Assigned To	Permission	Description
CCE	AOM Administrator	Monitoring Center and Alarm Assistant need to access AOM to obtain metric data.
CCE	LTS Administrator	Monitoring Center and Alarm Assistant need to access LTS to obtain logs.
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 metric data from CCE.
AOM	RMS ReadOnlyAccess	AOM CMDB manages cloud service instance data.
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 metric data from CCI.

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 has the Tenant Guest and IAM ReadOnlyAccess permissions in global projects as well as the Tenant Guest, CCE Administrator, and SWR Administrator permissions in regional projects. These permissions are required by Health Center, Monitoring Center, Logging, or Alarm Assistant to access other cloud services.

  To use Health Center, Monitoring Center, Logging, or Alarm Assistant 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 the **aom_admin_trust** agency, see **AOM Cloud Service Authorization**.

NOTE

Health Center, Monitoring Center, Logging, or Alarm Assistant may fail to run as expected if the required permissions are not assigned. When using Health Center, Monitoring Center, Logging, or Alarm Assistant, do not delete or modify the **cia_admin_trust** and **aom_admin_trust** agencies.

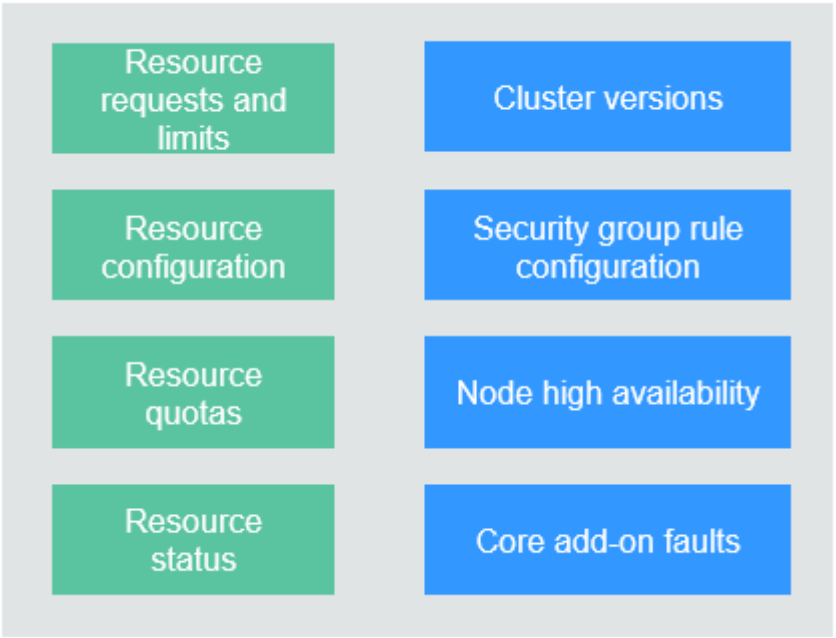
3.9.2 Health Center

3.9.2.1 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.



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 **3.9.2.3 Diagnosis Items and Rectification Solutions**.

3.9.2.2 Using Health Center

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 later than v1.17.
- 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 details page.

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 **3.9.3.2 Enabling Cluster Monitoring**.

----End

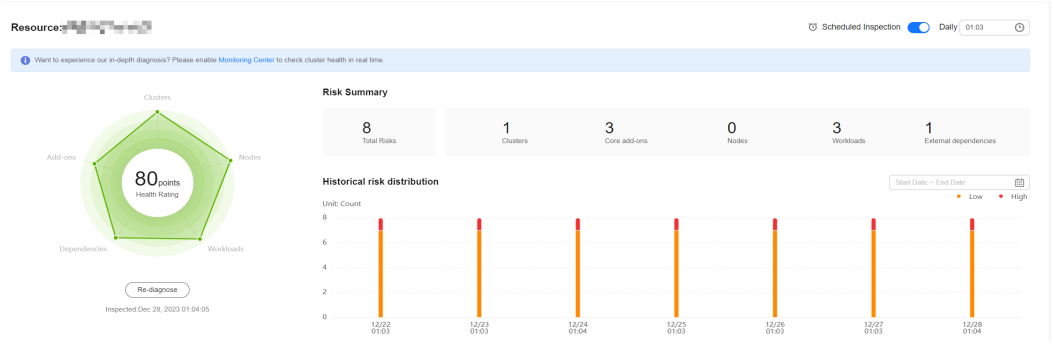
Scheduled Inspection

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 scheduled to be inspected only once every day.

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 3-191 Diagnosis overview

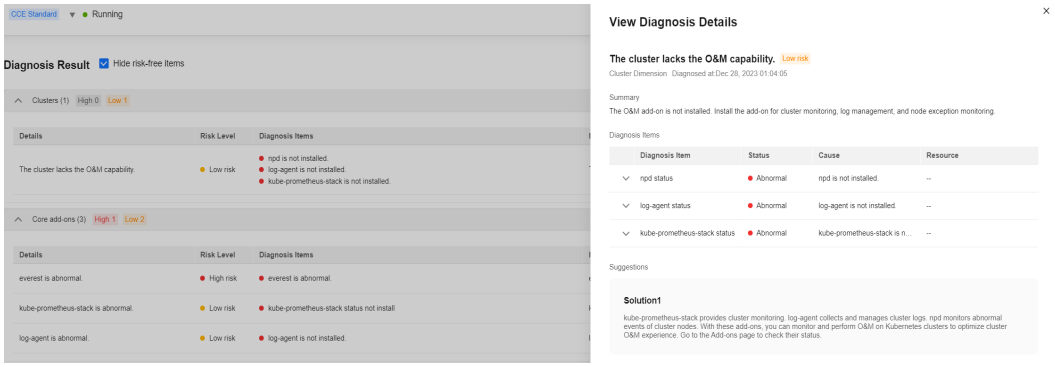


Diagnosis Result

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 3-192 Diagnosis result



3.9.2.3 Diagnosis Items and Rectification Solutions

Clusters

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
Cluster resource planning	Whether HA is enabled for master nodes	Yes	The cluster has only one master node. If the master node is faulty, the cluster is unavailable, affecting service reliability. Use an HA cluster to improve service resilience. When a master node is faulty, the cluster is still available.
	Whether the CPU requests of pods in the cluster have exceeded 80% of the cluster CPU	Yes	A request is the minimum CPU or memory a workload needs. Plan required resources based on service requirements. For details, see **3.5.3.5 Configuring Container Specifications**.
	Whether the memory requests of pods in the cluster have exceeded 80% of the cluster memory	Yes	
	Whether the cluster version has expired	No	After the cluster version has reached the end of service, new clusters cannot be created, and no technical support will be provided, including new feature updates, vulnerability or issue fixes, new patches, work order guidance, and online checks for the cluster version. Such clusters are not covered in the CCE SLA. Go to the **Clusters** page to upgrade the cluster version. For details, see **3.2.4.1 Upgrade Overview**.
Cluster O&M	Whether kube-prometheus-stack is normal	No	kube-prometheus-stack provides one-stop cluster monitoring. Go to the **Add-ons** page to install this add-on and check its status. For details, see **3.14.17 Cloud Native Cluster Monitoring**.

Scenario	Diagnosis Item	Enabling Monitoring Center	Rectification Solution
	Whether log-agent is normal	No	log-agent collects and manages workload logs. Go to the **Add-ons** page to install this add-on and check the add-on status.
	Whether npd is normal	No	node-problem-detector (npd) monitors the nodes. Go to the **Add-ons** page to install this add-on and check the add-on status. For details, see **3.14.4 CCE Node Problem Detector**.
Cluster configuration	Whether security groups are 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, services in 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.
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. Go to Monitoring Center, click the **Container Insights** tab, and click **Workloads** to view the instance metrics. If the metric values reach the thresholds, adjust the specifications. For details, see the everest parameters in "Installing the Add-on" of **everest**.
	Whether the memory usage of everest-controller has exceeded 80% in the last 24 hours	Yes	
kube-prometheus-stack status	Whether kube-prometheus-stack is normal	No	kube-prometheus-stack provides one-stop cluster monitoring. Go to the **Add-ons** page to install this add-on or check the add-on status.
	Whether the CPU usage of the prometheus workload has exceeded 80% in the last 24 hours	Yes	kube-prometheus-stack provides cluster monitoring. If the resource usage is too high, kube-prometheus-stack may be overloaded, and cluster monitoring is affected. Go to Monitoring Center, click the **Container Insights** tab, and click **Workloads** to view the instance metrics. If the metric values reach the thresholds, adjust the specifications. **NOTE** The PVC resource usage is checked when kube-prometheus-stack is deployed in server mode. In server mode, collected metrics data is 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	kube-prometheus-stack 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.
autoscaler status	Whether autoscaler is available when auto scaling is enabled for node pools	No	autoscaler provides auto scaling for clusters. If autoscaler is abnormal, atuo scaling that has been enabled for a node pool becomes unavailable. Check the add-on status on the **Add-ons** page. **NOTE** The autoscaler status is checked only when auto scaling is enabled for node pools.
log-agent status	Whether log-agent is normal	No	log-agent collects and manages workload logs. Go to the **Add-ons** page to install this add-on or check the add-on 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 **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.

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 requests and limits 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, and the node may became unstable. Release unnecessary processes on the node or **modify the PID limit** in a timely manner. 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. Go to Monitoring Center, click the **Container Insights** tab, and click **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	

3.9.3 Monitoring Center

3.9.3.1 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

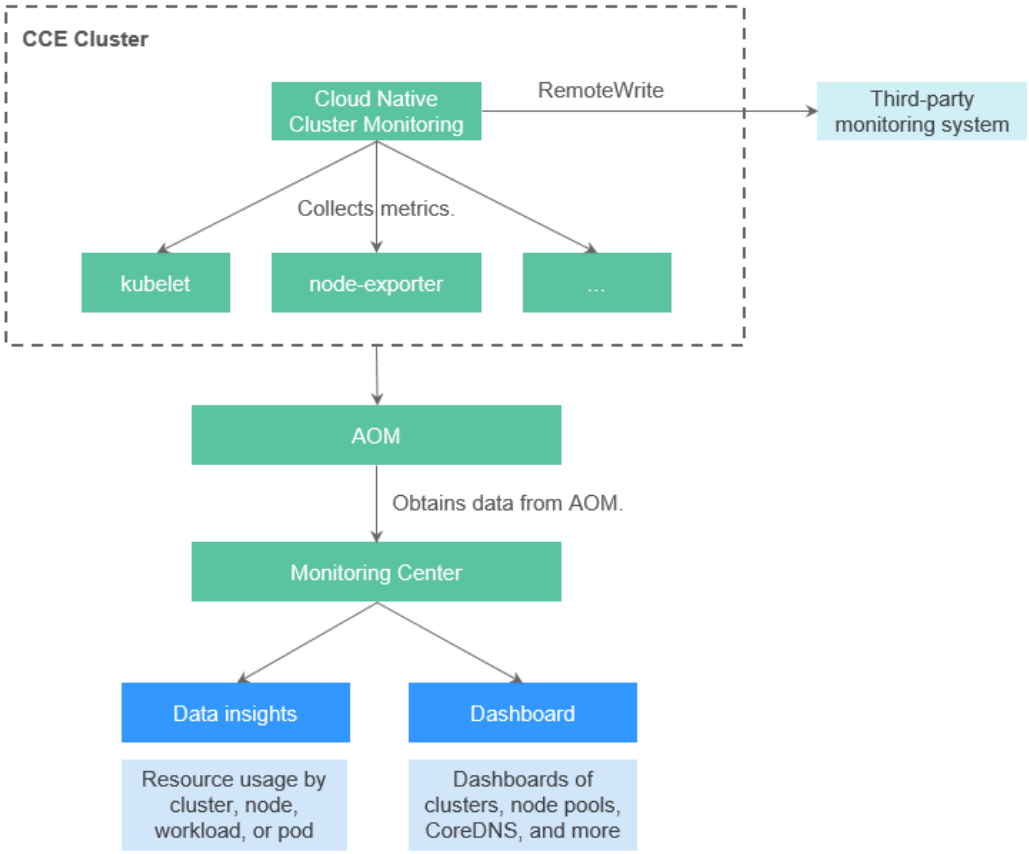
- Multi-dimensional data insights: comprehensively monitor Kubernetes native containers and displays metrics of **clusters**, **nodes**, **workloads**, **pods**, and **events** for the health status and load of clusters.
- **Dashboard**: shows various graphs such as line graphs and digit graphs on the 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 3-193 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 (**3.14.17 Cloud Native Cluster Monitoring**) for monitoring Kubernetes clusters.

The cloud-native monitoring add-on supports Prometheus Server and Prometheus Agent (a lightweight version of Prometheus Server). Prometheus Agent provides a lightweight metric collector for collecting monitoring metrics. Prometheus Server

provides thanos-based HA to ensure that Prometheus can run normally in the case of a single point of failure.

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 from the ECS or BMS console, manually install the ICAgent. For cluster nodes, ICAgent is automatically installed.

3.9.3.2 Enabling Cluster Monitoring

To enable cluster monitoring for a cluster, you need to install the cloud native cluster monitoring add-on that provides metric collection. After cluster monitoring is enabled, Monitoring Center collects cluster metrics and reports them to the AOM instance. This section describes how to enable cluster monitoring.

NOTICE

- After cluster monitoring is enabled, cluster metrics are reported to AOM instances. Basic metrics are free of charge but custom metrics are charged by AOM. For details, see **Pricing Details**.
- Running the cloud native cluster monitoring add-on in a cluster consumes cluster resources. Ensure that there are required cluster resources for installing the add-on. To view resource consumption, go to the cloud native cluster monitoring add-on details page.

Prerequisites

Before enabling cluster monitoring, you need to use 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 **Resource Permissions**.

Constraints

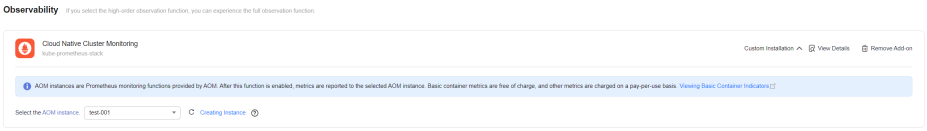
- The cluster version must be v1.17 or later.
- Before using Monitoring Center, you need to use an account in the **admin** user group to delegate CCE and its dependent services. After the authorization is complete, users with the CCE Administrator role or CCE FullAccess permission can perform all operations on Monitoring Center. Users with the CCE ReadOnlyAccess permission can view all resource information but cannot perform any operations.

- On-premises Prometheus or the prometheus add-on (**3.14.23 Prometheus (EOM)**) is not installed in the cluster.

Enabling Cluster Monitoring

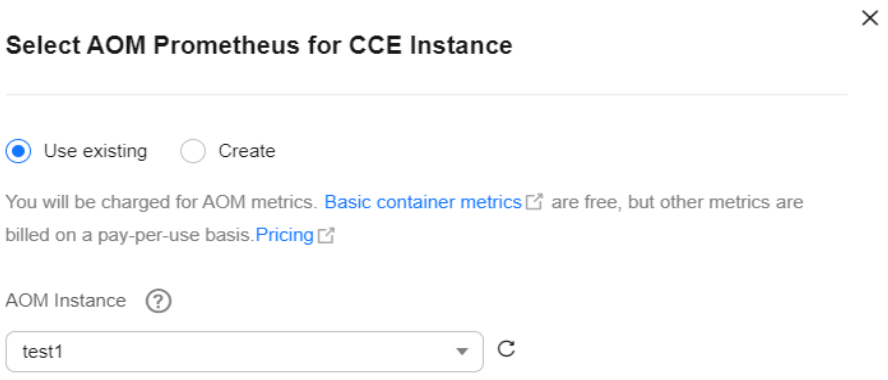
- **Enabling cluster monitoring during cluster purchase**
  a. Log in to the CCE console and purchase a cluster.
  b. On the **Select Add-on** page, select the cloud native cluster monitoring add-on.
  c. On the **Add-on Configuration** page, select the AOM instance to be interconnected with the cloud native cluster monitoring add-on. If there is no access code, create one first.

  Figure 3-194 Enabling cluster monitoring

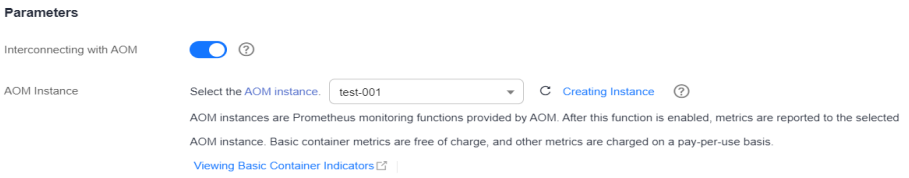
  d. After the cluster is created, create a node on the **Nodes** tab. After the node is created, the cloud native cluster monitoring add-on will be automatically deployed on the node.
- **Enabling cluster monitoring on the Monitoring Center page**
  a. Click the cluster name and choose **Monitoring Center** in the navigation pane.
  b. Click **Enable** and select the AOM instance that metrics are reported to.

  Figure 3-195 Enabling cluster monitoring

  c. Wait for 3 to 5 minutes until the monitoring data is reported to the AOM instance.
     The functions of Monitoring Center are available.
- **Enabling cluster monitoring on the Add-ons page**
  a. Click the cluster name and choose **Add-ons** in the navigation pane.
  b. Select the cloud native cluster monitoring add-on and click **Install**.
  c. Select Agent or Server mode as required and enable interconnection with AOM so that metrics can be reported to AOM instances.

Figure 3-196 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.

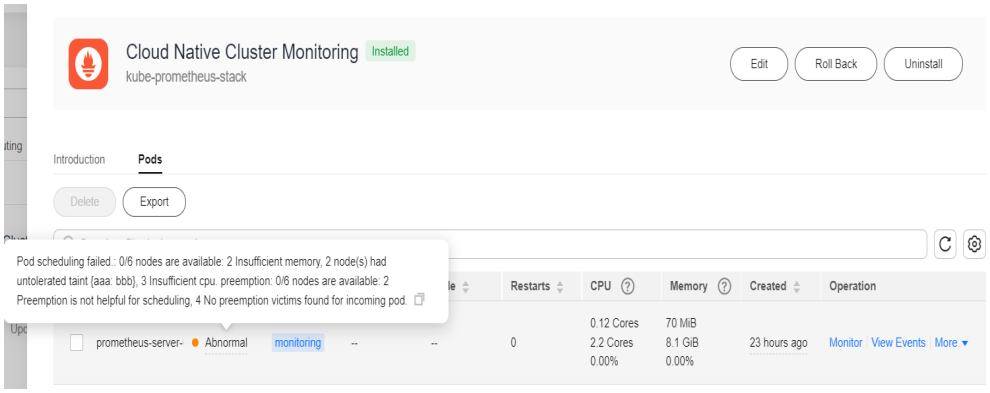
NOTE

To disable Monitoring Center, uninstall the cloud native cluster monitoring add-on on the **Add-ons** page or disable the interconnection with AOM.

FAQ

- Failed to enable cluster monitoring 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.



- 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.

3.9.3.3 Clusters

To observe the resource usage and health of a cluster, click **Container Insights** > **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 details page.

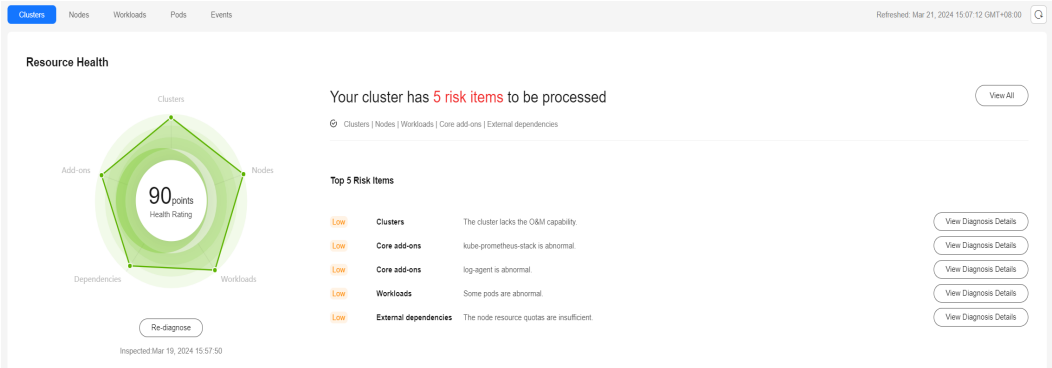
Step 2 Choose **Monitoring Center** in the navigation pane and click **Container Insights** > **Clusters**.

----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 3-197 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

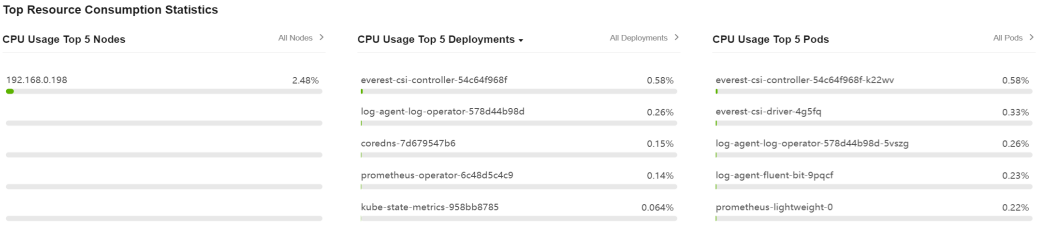
Control Plane Health Overview displays the percentage of exceptions on control plane components and master nodes, total QPS of the API server, and request error rate of the API server. If the API server (the API service provider of the cluster) on the control plane is abnormal, the cluster may fail to 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 3-198 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 3-199 Top Resource Consumption Statistics



Monitoring metrics

- CPU Usage

  Node CPU usage = Average percentage of the non-idle CPU time of the node

  Workload CPU usage = Average CPU usage in each pod of the workload

  Pod CPU usage = The used CPU cores/The sum of all CPU limits of the pods (If not specified, all node CPU cores are used.)

- Memory Usage

  Node memory usage = The used memory of the node/The total memory of the node

  Workload memory usage = Average memory usage in each pod of the workload

  Pod memory usage = The used physical memory/The sum of all memory limits of pods (If not specified, all node memory is used.)

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 **3.9.3.8.1 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.

3.9.3.4 Nodes

To monitor the resource usage of nodes, click **Container Insights** > **Nodes**. This page provides information about all nodes in a cluster and monitoring data of a single node, such as the CPU/memory usage, network inbound/outbound rate, and disk I/O read or write rate.

Navigation Path

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

Step 2 Choose **Monitoring Center** in the navigation pane and click **Container Insights** > **Nodes**.

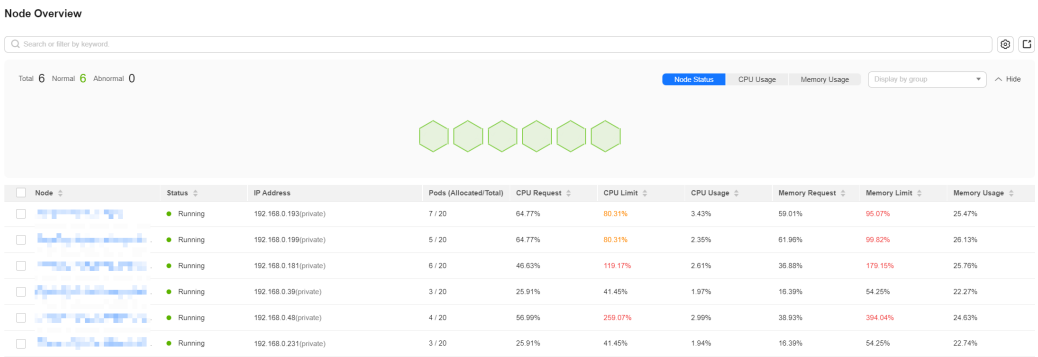
The page displays information about all nodes. To view the monitoring data of a node, click the node name to access its **Overview** page and switch to the **Pods** or **Monitoring** tab.

----End

Nodes

This tab lists the name, status, IP address, number of pods (allocated/total), CPU request/limit/usage, and memory request/limit/usage of each node.

Figure 3-200 Nodes



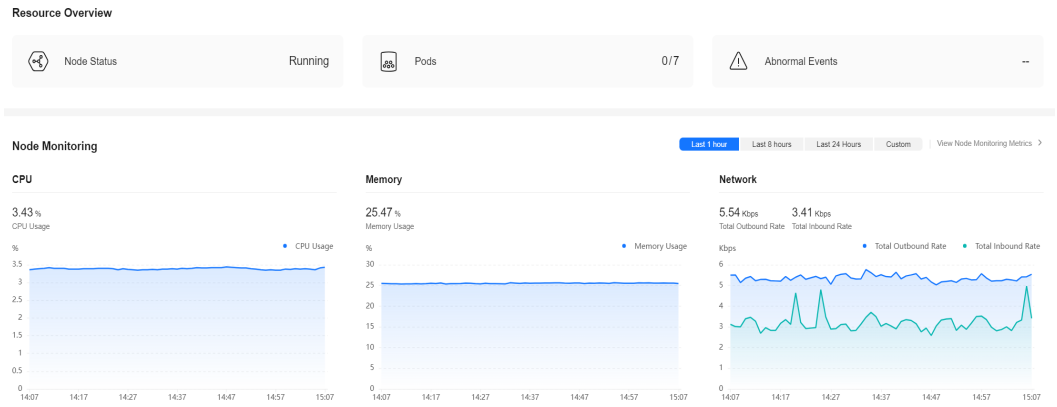
You can search for the desired node by name, status, private IP address, or public IP address. You can click **Export** to export data of all nodes or selected nodes. The exported file is in .xlsx format, and the file name contains the timestamp.

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

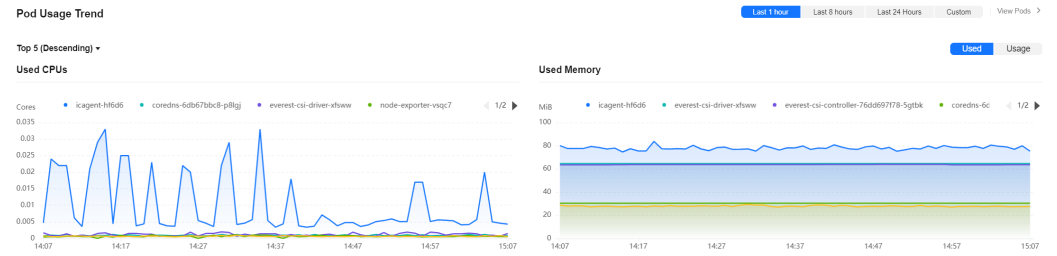
You can click the node name to view the resource health overview, such as the node status, number of pods, and abnormal events. You can also view the monitoring overview of the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 3-201 Resource overview and monitoring overview



The **Overview** tab also shows the pod usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each pod on the node. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left corner to view the top 5 data in descending or ascending order.

Figure 3-202 Pod usage trend

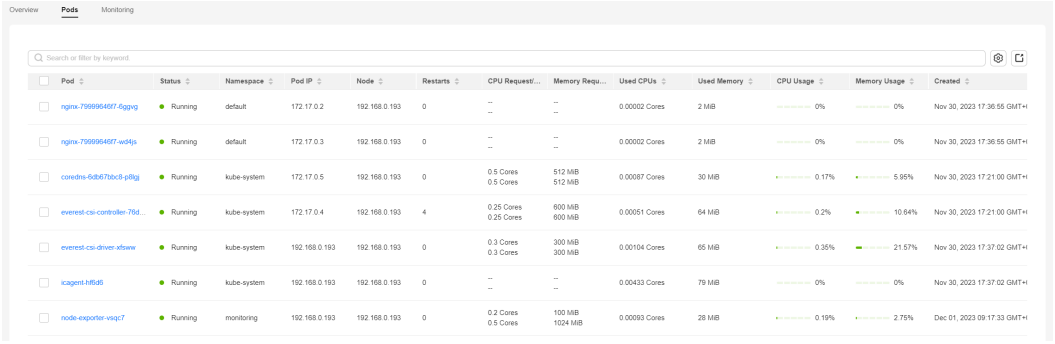


For more metrics, go to the **Monitoring** tab.

Pods

This tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, used CPU cores, CPU usage, used memory, memory usage of each pod.

Figure 3-203 Pods



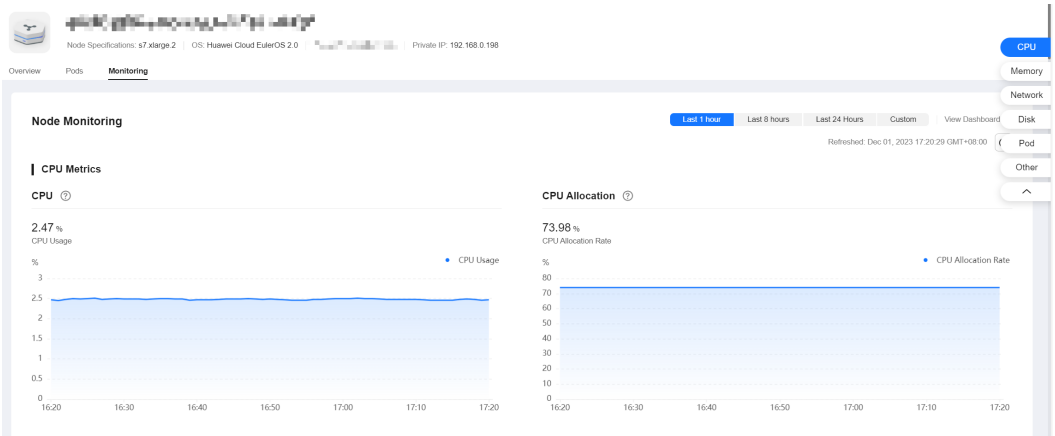
You can search for the desired pod by name, status, namespace, IP address, or node. You can click **Export** to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **3.9.3.6 Pods**.

Monitoring

This tab shows 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 the **Dashboard** page. For details, see **3.9.3.8.1 Using Dashboard**.

Figure 3-204 Node monitoring



- **CPU Metrics**
  - CPU usage: Average percentage of the non-idle CPU time of the node
  - CPU allocation ratio: the percentage of CPU cores requested by all containers on the node to the total CPU cores on the node.
  - Single-core CPU usage: the percentage of the non-idle time of each CPU core on the node.
- **Memory Metrics**
  - Memory usage: the percentage of memory used by the node.

  - Memory allocation ratio: the percentage of memory requested by all containers on the node to the total memory on the node.
- **Networking Metrics**
  - Outbound rate: the number of bytes sent by the NIC on the node per second in different time periods.
  - Inbound rate: the number of bytes received by the NIC on the node per second in different time periods.
  - Packet loss rate (transmit): the percentage of packets not received by the recipient to packets sent from the NIC of the node.
  - Packet loss rate (receive): the percentage of packets not received by the NIC of the node to packets sent to the NIC.
- **Disk Metrics**
  - Disk read rate: the number of bytes read from each file system on the node per second in different time periods.
  - Disk write rate: the number of bytes written to each file system on the node per second in different time periods.
  - Disk usage: the percentage of used disk space of each file system on the node in different time periods.
- **Pod Metrics**
  - Pod CPU usage: the percentage of CPU used by each pod on the node in different time periods to the CPU limit for each pod.
  - Pod memory usage: the percentage of memory used by each pod on the node in different time periods to the memory limit for each pod.
  - Pod status and quantity: the total number of pods in the **Unavailable**, **Unready**, **Running**, **Completed**, or **Other** state on the node in different time periods.
  - Pod quantity trend: the number of pods on the node in 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.

3.9.3.5 Workloads

To monitor the resource usage of workloads, click **Container Insights** > **Workloads**. This page provides information about all workloads in a cluster and monitoring data of a single workload, such as the CPU/memory usage and network inbound/outbound rate.

Navigation Path

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

Step 2 Choose **Monitoring Center** in the navigation pane and click **Container Insights** > **Workloads**.

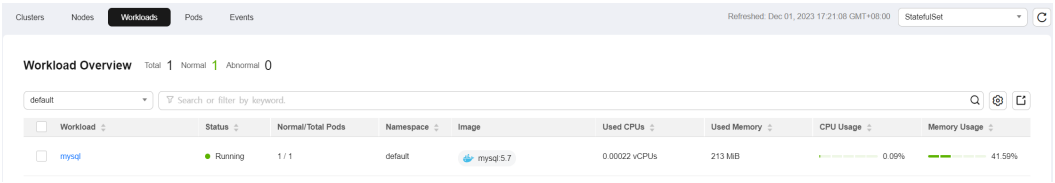
The page displays information about all workloads. To view the monitoring information of a workload, click the workload name to access its **Overview** page and switch to the **Pods** or **Monitoring** tab.

----End

Workloads

This tab lists the name, status, number of pods (normal/all), namespace, image name, CPU usage, and memory usage of each workload.

Figure 3-205 Workloads



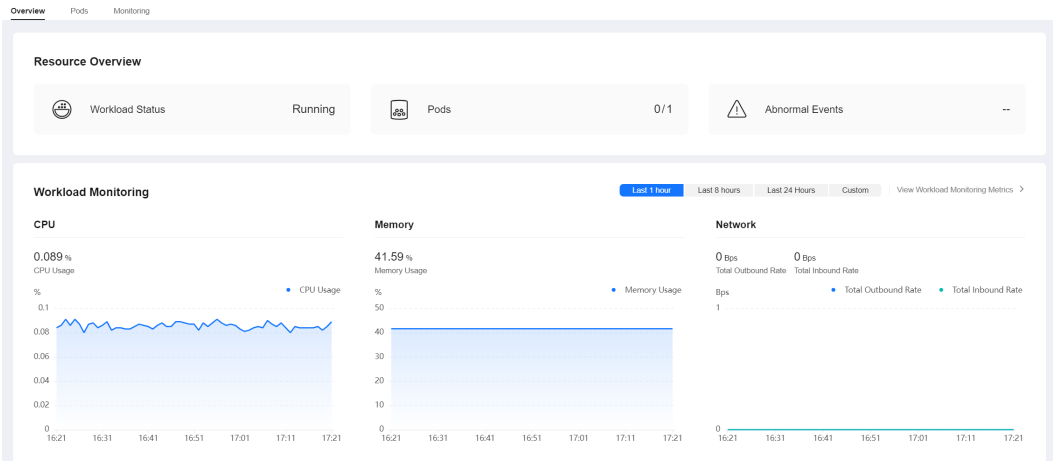
You can search for the desired workload by workload type (in the upper right corner), namespace, and workload name or status.

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.

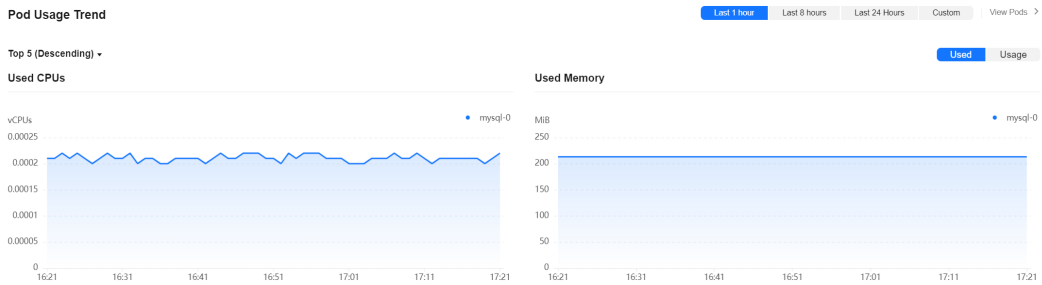
Figure 3-206 Resource overview and monitoring overview



The **Overview** tab also shows the pod usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory

usage, and used memory of each pod of the workload. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left corner to view the top 5 data in descending or ascending order.

Figure 3-207 Pod usage trend

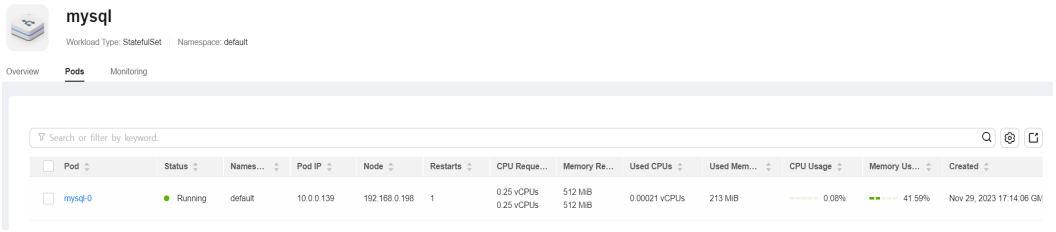


For more metrics, go to the **Monitoring** tab.

Pods

This tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, CPU usage, and memory usage of each pod.

Figure 3-208 Pods



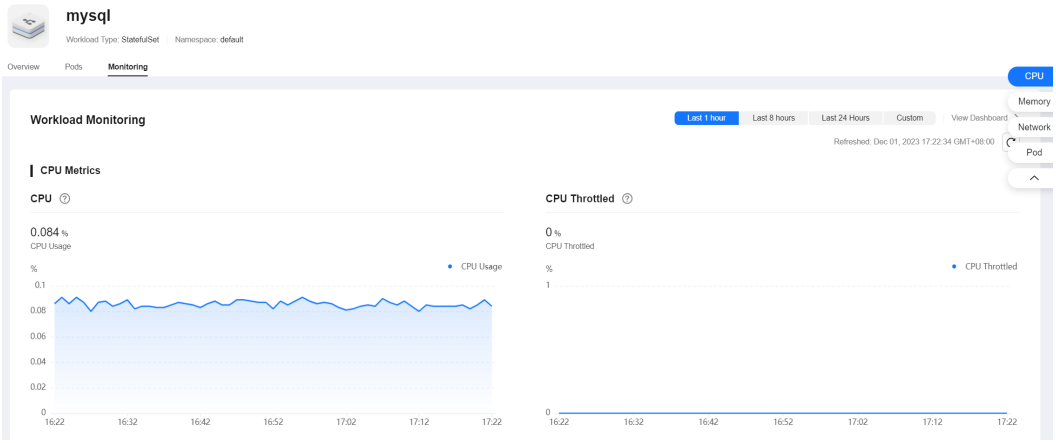
You can search for the desired pod by name, status, namespace, IP address, or node. You can click **Export** to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **3.9.3.6 Pods**.

Monitoring

This tab shows 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 the **Dashboard** page. For details, see **3.9.3.8.1 Using Dashboard**.

Figure 3-209 Workload monitoring



- **CPU Metrics**
  - CPU usage: the percentage of the CPU used by containers in all pods of the workload in different time periods to the total CPU limit for all containers.
  - CPU throttled: the average percentage of time containers have been throttled in all pods of the workload in different time periods.
- **Memory Metrics**
  - Memory usage: the percentage of memory used by containers in all pods of the workload in different time periods to the total memory limit for all containers.
- **Networking Metrics**
  - Total outbound rate: the total number of bytes transmitted by containers in all pods of the workload per second in different time periods.
  - Total inbound rate: the total number of bytes received by containers in all pods of the workload per second in different time periods.
  - Packet loss rate (transmit): the percentage of packets not received by the recipient to packets sent from containers in all pods of the workload in different time periods.
  - Packet loss rate (receive): the percentage of packets not received by containers in all pods of the workload to packets sent to the containers in different time periods.
- **Pod Metrics**
  - Pod CPU usage: the percentage of CPU used by each pod of the workload in different time periods to the CPU limit for each pod.
  - Pod memory usage: the percentage of memory used by each pod of the workload in different time periods to the memory limit for each pod.
  - Pod status and quantity: the total number of pods in the **Unavailable**, **Unready**, **Running**, **Completed**, or **Other** state of the workload in different time periods.
  - Pod quantity trend: the number of pods (replicas) of the workload in different time periods.

3.9.3.6 Pods

To monitor the resource usage of pods, choose **Container Insights** > **Pods**. This page provides information about all pods in a cluster and monitoring data of a single pod, such as the CPU/memory usage and network inbound/outbound rate.

Navigation Path

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

Step 2 Choose **Monitoring Center** in the navigation pane and click **Container Insights** > **Pods**.

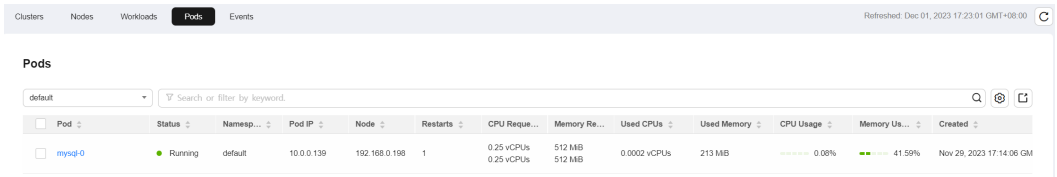
The page displays information about all pods. To view the monitoring information of a pod, click the pod name to access its **Overview** page and switch to the **Containers** or **Monitoring** tab.

----End

Pods

This tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, CPU usage, and memory usage of each pod.

Figure 3-210 Pods



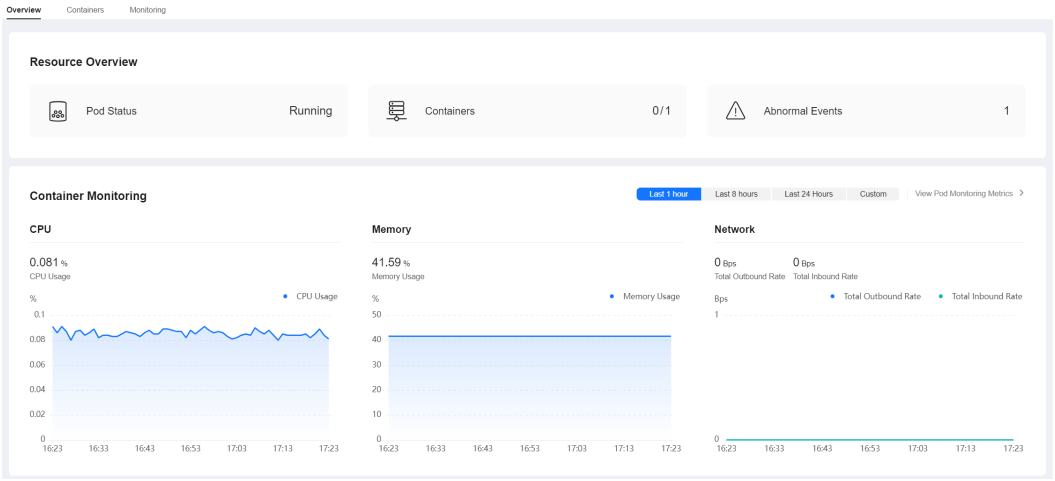
You can search for the desired pod by namespace and pod name, status, pod IP address, or node.

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 node in the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 3-211 Resource overview and monitoring overview



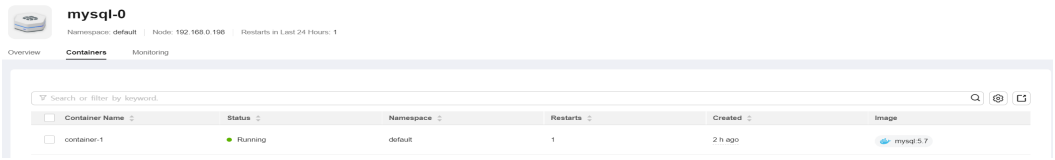
The **Overview** tab also shows the container usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each container in the pod. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left corner to view the top 5 data in descending or ascending order.

For more metrics, go to the **Monitoring** tab.

Containers

This tab contains details such as the name, status, namespace, number of restarts, and image of each container.

Figure 3-212 Containers

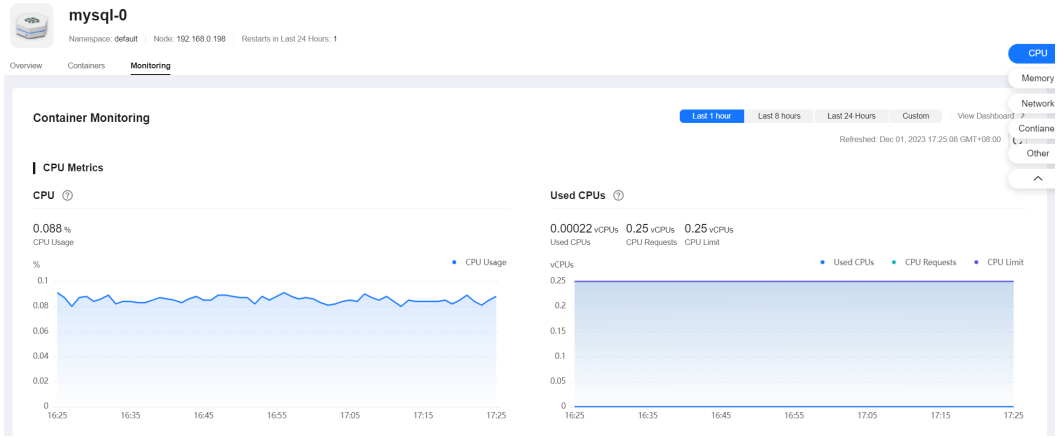


You can search for the desired container by name, status, or namespace. You can click **Export** to export data of all containers or selected containers. The exported file is in .xlsx format, and the file name contains the timestamp.

Monitoring

This tab shows the resource usage of the pod 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 the **Dashboard** page. For details, see **3.9.3.8.1 Using Dashboard**.

Figure 3-213 Pod monitoring



- **CPU Metrics**
  - CPU usage: the percentage of CPU used by all containers in a pod in different time periods to the total CPU limit for all containers.
  - Used CPU: the CPU that the pod is using.
  - CPU request: the CPU requested for the pod.
  - CPU limit: the CPU limit configured for 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 percentage of memory used by all containers in a pod in different time periods to the total memory limit for all containers.
  - Used memory: the memory that the pod is using.
  - Memory request: the memory requested for the pod.
  - Memory limit: the memory limit configured for the pod. When the used memory is close to this limit, OOM will occur.
- **Networking Metrics**
  - Total outbound rate: the total number of bytes transmitted 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 percentage of CPU used by each container in the pod in different time periods to the CPU limit for each container.
  - Container memory usage: the percentage of memory used by each container in the pod in different time periods to the memory limit for each container.
  - Container CPU throttled: the percentage of time duration each container has been throttled in different time periods.
  - Container network packet loss rate: the percentage of packets not received by each container in the pod to packets sent to each container in different time periods.

- **Other Metrics**
  - Historical pod status: the status of the pod in different periods.
  - Historical container status: the status of each container in the pod in different time periods.

3.9.3.7 Events

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 **Container Insights** > **Events** page.

Navigation Path

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

Step 2 Choose **Monitoring Center** in the navigation pane and click **Container Insights** > **Events**.

The **Events** page has two tabs: **Overview** and **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.

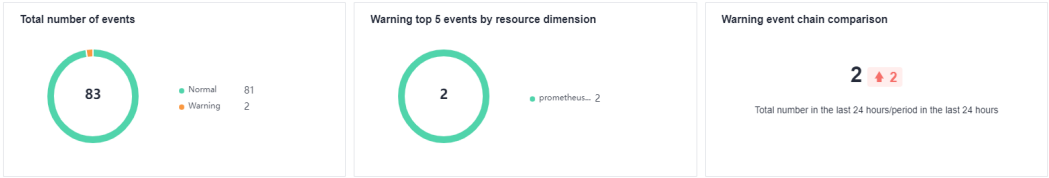
----End

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 3-214 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 3-214 Event statistics



The bar charts in Figure 3-215 display the trend of the number of normal events and warning events in the last 24 hours.

Figure 3-215 Warning/Normal event trend

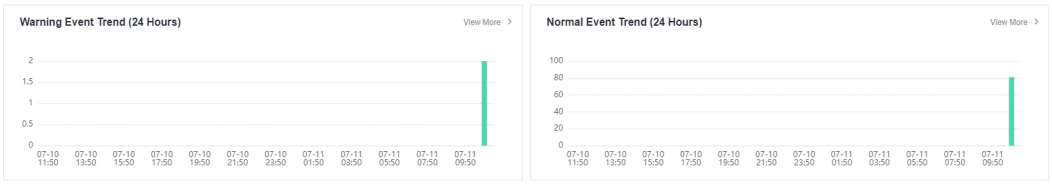
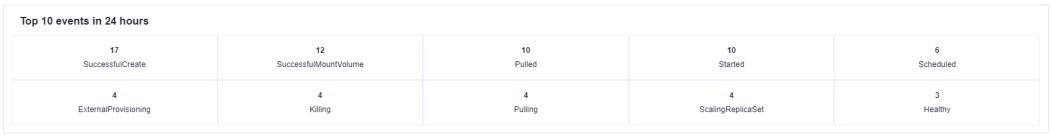


Figure 3-216 displays the names of top 10 events in the last 24 hours.

Figure 3-216 Top 10 events in 24 hours



Events

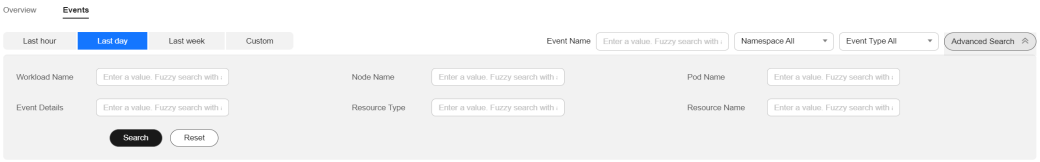
Searching for Events

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.

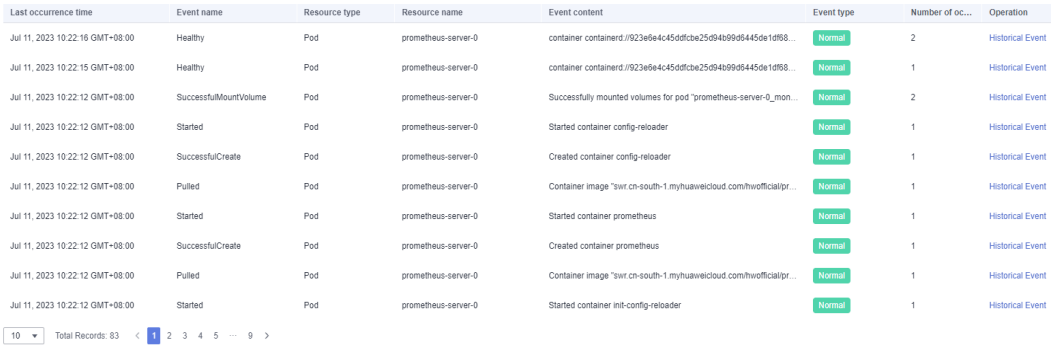
Figure 3-217 Searching for events



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 3-218 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

3.9.3.8 Dashboard

3.9.3.8.1 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 later than v1.17.
- 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 details page.

Step 2 Choose **Monitoring Center** in the navigation pane and 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

3.9.3.8.2 Cluster View

The cluster view is generated based on metrics and Prometheus Query Language (PromQL) and displays information about nodes, CPUs, memory, networks, and disks, which allow you to better monitor the running of clusters. The following

describes cluster resources from two parts: metric description and metric list. In the resource diagrams, larger values in bytes can be converted to ones in MB, KB, or GB.

Metric Description

The following tables list the metrics displayed by the cluster view.

Figure 3-219 Basic resource diagram

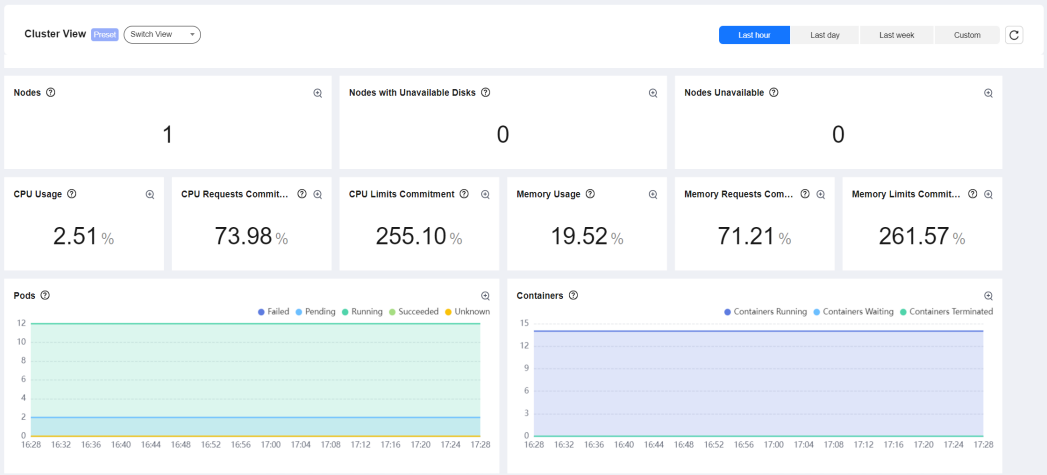


Table 3-283 Basic resource metrics

Metric	Unit	Description
Nodes	Count	Number of nodes in a cluster
Nodes with Unavailable Disks	Count	Number of nodes with unavailable disks in a cluster
Nodes Unavailable	Count	Number of unready nodes in a cluster
CPU Usage	%	Percentage of the used CPU cores of all containers in a cluster to the CPU limits for all containers
CPU Requests Commitment	%	Percentage of the requested CPU cores by all containers in a cluster to the allocatable CPU cores of the cluster
CPU Limits Commitment	%	Percentage of the limits on CPU cores that can be used by all containers in a cluster to the allocatable CPU cores of the cluster
Memory Usage	%	Percentage of the used memory of all containers in a cluster to the memory limits for all containers

Metric	Unit	Description
Memory Requests Commitment	%	Percentage of the requested memory by all containers in a cluster to the allocatable memory of the cluster
Memory Limits Commitment	%	Percentage of the limits on memory that can be used by all containers in a cluster to the allocatable memory of the cluster
Pods	Count	Number of pods in different states (**Failed**, **Pending**, **Running**, **Succeeded**, and **Unknown**) in a cluster
Containers	Count	Number of containers in different states (**Containers Running**, **Containers Waiting**, and **Containers Terminated**) in a cluster
Used CPU	Cores	Total number of CPU cores used by all containers in each namespace
Used Memory	Bytes	Total amount of the memory used by all containers in each namespace

Figure 3-220 Network diagram

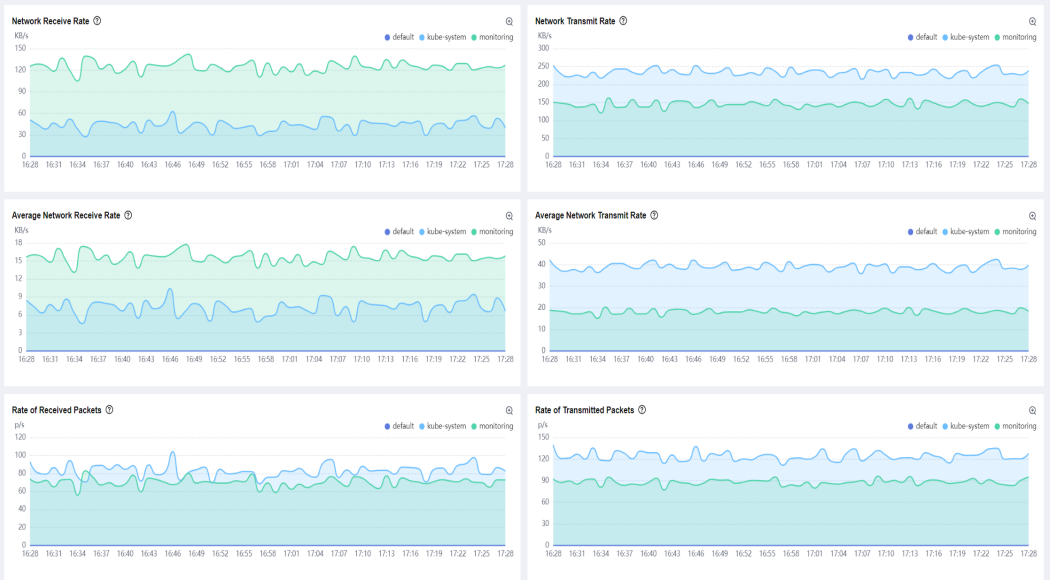


Table 3-284 Network metrics

Metric	Unit	Description
Network Receive Rate	Bytes/s	Total number of bytes received by all containers in each namespace per second

Metric	Unit	Description
Network Transmit Rate	Bytes/s	Total number of bytes sent by all containers in each namespace per second
Average Network Receive Rate	Bytes/s	Average number of bytes received by containers in each namespace per second
Average Network Transmit Rate	Bytes/s	Average number of bytes sent by containers in each namespace per second
Packet Receive Rate	Count/s	Total number of packets received by all containers in each namespace per second
Packet Transmit Rate	Count/s	Total number of packets sent by all containers in each namespace per second
Packet Loss Rate (Receive)	Count/s	Total number of packets not received by all containers in each namespace per second
Packet Loss Rate (Transmit)	Count/s	Total number of packets sent from all containers but not received by the recipient in each namespace per second

Figure 3-221 Disk diagram

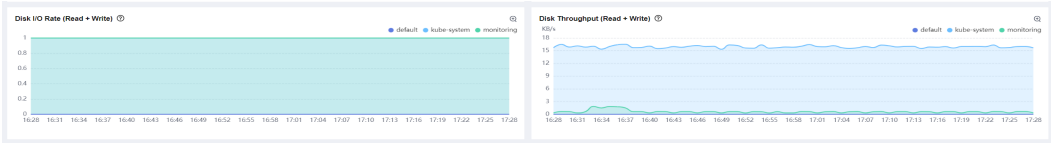


Table 3-285 Metrics

Metric	Unit	Description
Disk I/O Rate (Read + Write)	Count/s	Total number of read and write operations performed on the disk of all containers in each namespace per second
Disk Throughput (Read + Write)	Bytes/s	Total number of bytes read from and written to the disk of all containers in each namespace per second

Metric List

The following table lists the metrics used in the cluster view.

Table 3-286 Cluster metrics

Metric	Type	Description
kube_pod_container_resource_requests	Gauge	The number of requested request resources by a container
kube_pod_container_resource_limits	Gauge	The number of requested limit resources by a container
kube_node_status_allocatable	Gauge	The amount of resources allocatable for pods
kube_pod_status_phase	Gauge	The pods current phase
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
kube_node_info	Gauge	Node information
kube_node_status_condition	Gauge	The condition of a cluster node
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 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
container_network_transmit_bytes_total	Counter	Cumulative count of bytes transmitted
container_network_receive_packets_total	Counter	Cumulative count of packets received
container_network_transmit_packets_total	Counter	Cumulative count of packets transmitted
container_network_receive_packets_dropped_total	Counter	Cumulative count of packets dropped while receiving

Metric	Type	Description
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets dropped while transmitting
container_fs_reads_total	Counter	Cumulative count of reads completed
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read

3.9.3.8.3 API Server View

The API server view displays request, work queue, and resource metrics, which allow you to better monitor the running of the API server.

Metric Description

The following tables list the metrics displayed by the API server view.

Figure 3-222 Request metrics

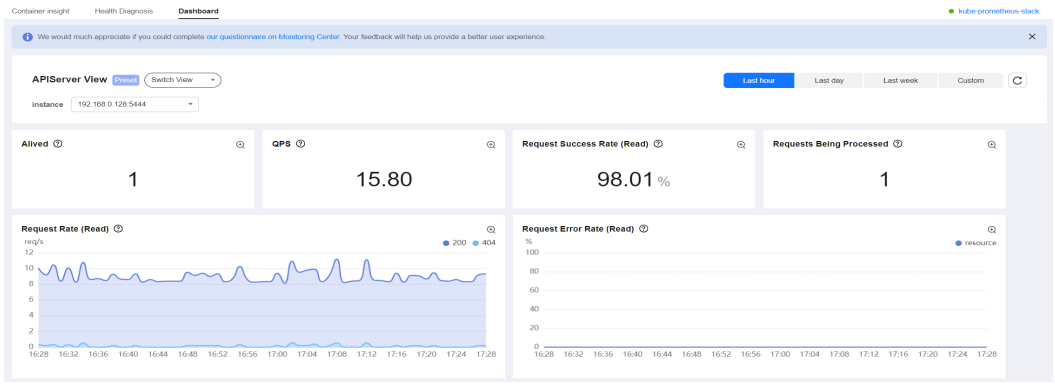


Table 3-287 Request metrics

Metric	Unit	Description
Alived	Count	Number of live component instances
QPS	Requests/s	Number of requests with different response codes per second
Request Success Rate (Read)	%	Percentage of read requests whose response code is 20x to read requests per second
Requests Being Processed	Count	Number of requests being processed by the APIServer
Request Rate (Read)	Requests/s	Number of read requests with different response codes per second

Metric	Unit	Description
Request Error Rate (Read)	%	Percentage of read requests that are rejected per second
P99 Request Latency (Read)	ms	P99 latency of read operations
Request Rate (Write)	Requests/s	Number of write requests with different response codes per second
Request Error Rate (Write)	%	Percentage of write requests that fail per second
P99 Request Latency (Write)	ms	P99 latency of write operations

Figure 3-223 Work queue metrics

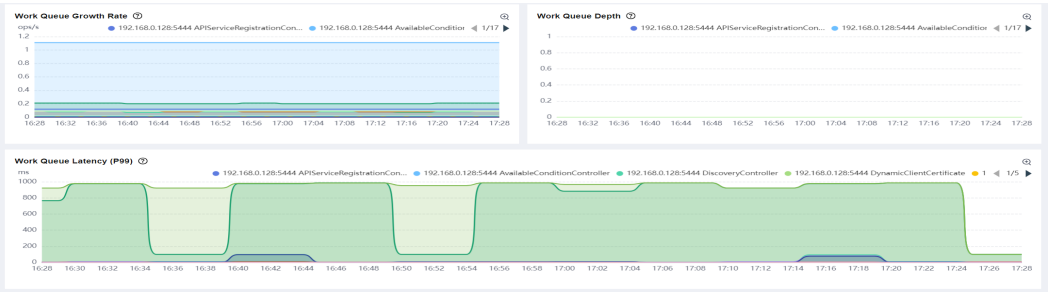


Table 3-288 Work queue metrics

Metric	Unit	Description
Work Queue Growth Rate	Count/s	Number of additions handled by the API server work queue per second
Work Queue Depth	Count	Depth of a work queue
Work Queue Latency (P99)	ms	P99 latency of each API server request in a work queue

Figure 3-224 Resource metrics

Table 3-289 Resource metrics

Metric	Unit	Description
Used Memory	Bytes	Memory used by the API server
Used CPU	Cores	CPU used by the API server
Goroutines	Count	Number of goroutines

Metric List

The following table lists the metrics used in the API server view.

Table 3-290 API server metrics

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	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

3.9.3.8.4 Pod View

The pod view displays pod resource, network, and disk metrics, which allow you to better monitor the running of pods.

Metric Description

The following tables list the metrics displayed by the pod view.

Figure 3-225 Pod resource metrics

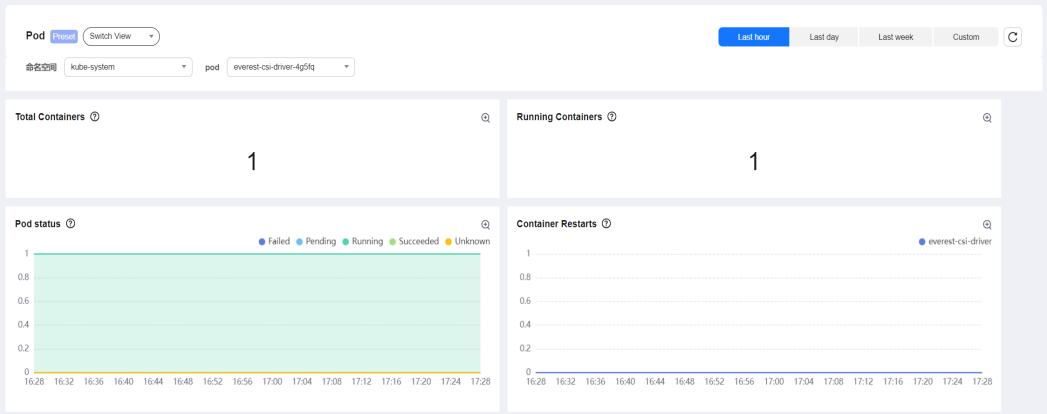


Table 3-291 Pod resource metrics

Metric	Unit	Description
Containers	Count	Number of containers in a pod
Running Containers	Count	Number of running containers in a pod
Pod Status	Count	Number of pods in different states
Container Restarts	Count	Number of container restarts
Used CPU	Cores	CPU used by a pod
CPU Efficiency & Utilization	%	Efficiency: Used CPU/Requested CPU Usage: Used CPU/Requested CPU
Used Memory	Bytes	Used memory
Memory Efficiency & Utilization	%	Efficiency: Used memory/Requested memory Usage: Used memory/Total memory
CPU Throttling	%	CPU throttling period limit rate

Figure 3-226 Pod network metrics

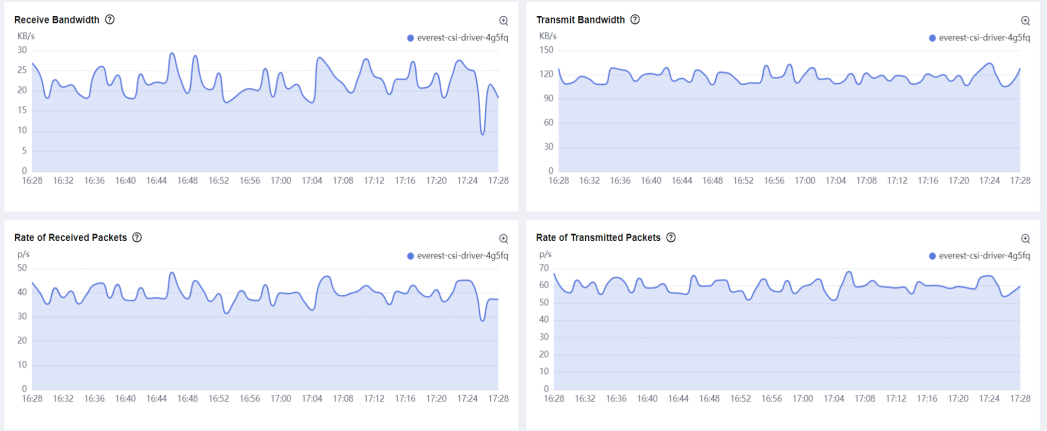


Table 3-292 Pod network metrics

Metric	Unit	Description
Network Receive Rate	Bytes/s	Number of bytes received by the container per second
Network Transmit Rate	Bytes/s	Number of bytes transmitted by the container per second
Rate of Received Packets	Count/s	Number of packets received by the container per second
Rate of Transmitted Packets	Count/s	Number of packets sent by the container per second
Packet Loss Rate (Receive)	Bytes/s	Number of packets not received by the container per second
Packet Loss Rate (Transmit)	Bytes/s	Number of packets sent from the container but not received by the recipient per second

Figure 3-227 Pod disk metrics

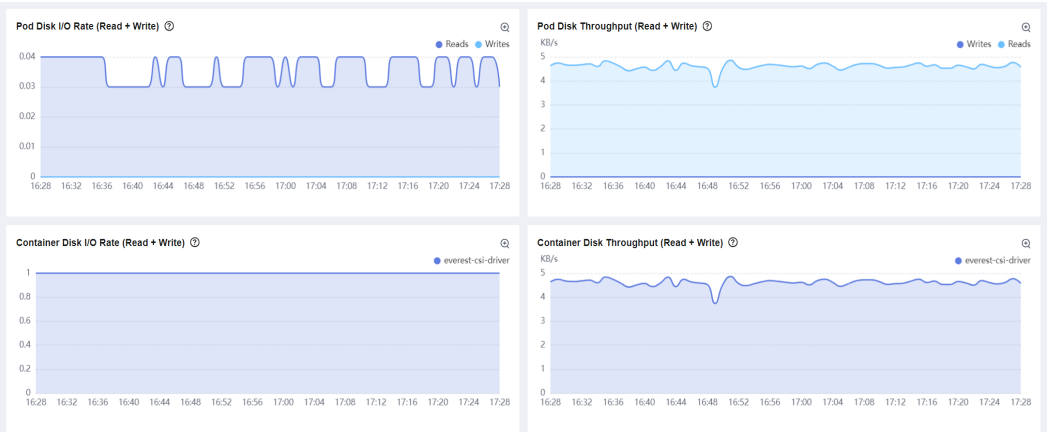


Table 3-293 Pod disk metrics

Metric	Unit	Description
Pod Disk I/O Rate (Read + Write)	Count/s	Number of read/write operations performed by a pod disk per second
Pod Disk Throughput (Read + Write)	Bytes/s	Data volume read/written by a pod disk, measured in bytes per second
Container Disk I/O Rate (Read + Write)	Count/s	Number of read/write operations performed by a pod disk per second
Container Disk Throughput (Read + Write)	Bytes/s	Data volume read/written by a pod disk, measured in bytes per second
File System Usage	%	File system usage
File System Used	Bytes	Number of bytes used by the file system

Metric List

The following table lists the metrics used in the pod view.

Table 3-294 Pod View metrics

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 pods current phase
kube_pod_container_status_restarts_total	Counter	The number of container restarts per container
container_cpu_usage_seconds_total	Counter	Cumulative CPU time consumed in seconds
kube_pod_container_resource_requests	Gauge	The number of requested request resources by a container
container_spec_cpu_quota	Gauge	CPU quota of the container
container_memory_working_set_bytes	Gauge	Current working set in bytes
container_spec_memory_limit_bytes	Gauge	Memory limit for a container

Metric	Type	Description
container_cpu_cfs_throttled_periods_total	Counter	Number of throttled period intervals
container_cpu_cfs_periods_total	Counter	Number of elapsed enforcement period intervals
container_network_receive_bytes_total	Counter	Cumulative count of bytes received
container_network_transmit_bytes_total	Counter	Cumulative count of bytes transmitted
container_network_receive_packets_total	Counter	Cumulative count of packets received
container_network_transmit_packets_total	Counter	Cumulative count of packets transmitted
container_network_receive_packets_dropped_total	Counter	Cumulative count of packets dropped while receiving
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets dropped while transmitting
container_fs_reads_total	Counter	Cumulative count of reads completed
container_fs_writes_total	Counter	Cumulative count of writes completed
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read
container_fs_writes_bytes_total	Counter	Cumulative count of bytes written
container_fs_usage_bytes	Gauge	Number of bytes that are consumed by the container on this filesystem
container_fs_limit_bytes	Gauge	Number of bytes that can be consumed by the container on this filesystem

3.9.3.8.5 Host View

The host view allows you to check the host resource usage and health and view common system device metrics such as disks and file systems to better monitor the running of nodes.

Metric Description

The following table lists the metrics displayed by the host view.

Figure 3-228 Host resource metrics

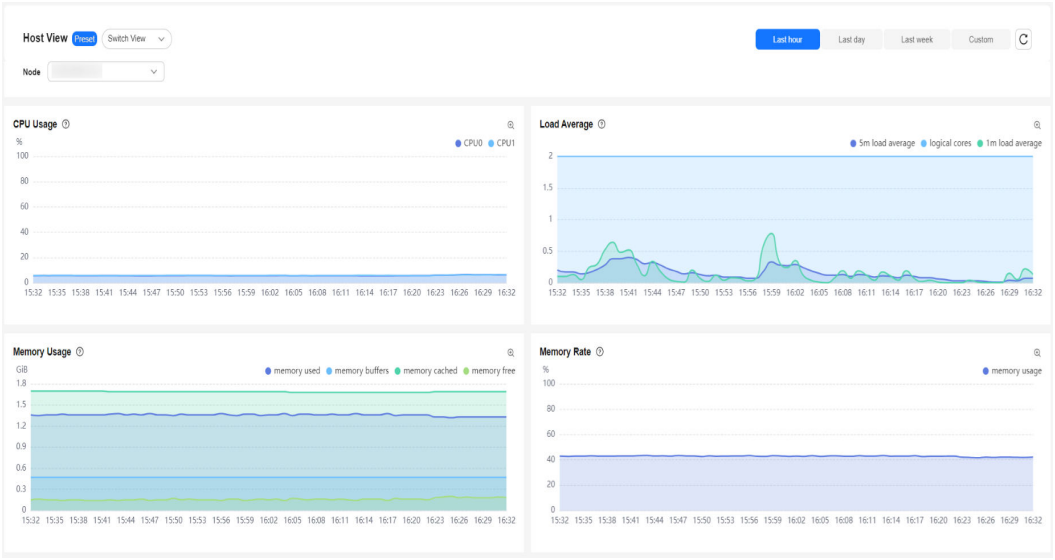


Table 3-295 Metrics

Metric	Unit	Description
CPU Usage	%	Usage of each CPU core
Load Average	/	Competition for 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.
Used Memory	Bytes	The amount of memory used in different modes
Memory Usage	%	Memory usage of the host
Disk Written	Bytes/s	Write rates of different disks
Disk Read	Bytes/s	Read rates of different disks
Used Disk Space	Bytes	Available disk space and used disk space
Disk Space Usage	%	Disk usage of different devices
Disk I/O Latency	Second	Disk I/O latency

Metric	Unit	Description
TCP Connection	Count	Number of TCP sockets • **alloc**: number of TCP sockets that have been allocated (established and applied for sk_buff) • **inuse**: number of TCP sockets in use (listening) • **orphan**: number of TCP sockets that have been allocated (established and applied for sk_buff) • **tw**: number of TCP sockets waiting to be closed
UDP Usage	Count	UDP usage • **liteInuse**: number of UDP-Lite sockets in use • **inuse**: number of UDP sockets in use • **useMemory**: usage of the UPD buffer
File Descriptor	EB: 10 to the power of 18	Maximum number of file descriptors that can be opened by the process
Used File Descriptor	Count	Number of allocated file descriptors
Socket Usage	Count	Socket usage • **socketsUsed**: total number of used protocol sockets • **fragInuse**: number of Frag sockets in use • **fragMemroy**: usage of the Frag buffer • **rawInuse**: number of raw sockets in use
Abnormal File System	/	File system status • **readonly**: read-only file system • **deviceError**: file system error
Disk I/O Rate	Count/s	Number of read/write operations performed by a disk per second
Disk Read/Write Latency	Second	Disk read/write latency
I/O Queues	/	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.
Process State	Count	Number of processes in different states

Metric	Unit	Description
Connection Tracking Table Entries	Count	• **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 table lists the metrics used in the host view.

Table 3-296 Metrics

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
node_filesystem_avail_bytes	Gauge	Available file system size in bytes

Metric	Type	Description
node_disk_io_time_seconds_total	Counter	Total seconds spent doing I/O operations
node_sockstat_TCP_alloc	Gauge	Number of allocated TCP sockets
node_sockstat_UDPLITE_inuse	Gauge	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	Number of used IPv4 sockets
node_filesystem_readonly	Gauge	Read-only status of the file system
node_disk_reads_completed_total	Counter	Number of disk reads completed successfully
node_disk_read_time_seconds_total	Counter	The 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	Number of processes blocked waiting for I/O to complete
node_nf_conntrack_entries	Gauge	Number of currently allocated flow entries for connection tracking

3.9.3.8.6 Node View

The node view displays node resource, network, and disk metrics, which allow you to better monitor the running of nodes.

Metric Description

The following tables list the metrics displayed by the node view.

Figure 3-229 Node resource metrics

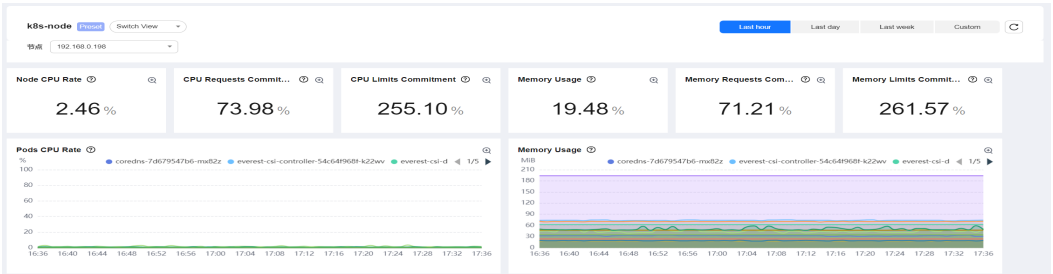


Table 3-297 Node resource metrics

Metric	Unit	Description
Node CPU Usage	%	CPU usage of the node
CPU Requests Commitment	%	Percentage of requested CPU cores by the node
CPU Limits Commitment	%	Percentage of the limits on CPU cores that can be used by the node
Memory Usage	%	Memory usage of the node
Memory Requests Commitment	%	Percentage of the requested memory by the node to the total allocatable memory of the node
Memory Limits Commitment	%	Percentage of the limits on the amount of memory that can be used by the node to the total allocatable memory of the node
Pod CPU Usage	%	CPU usage of the pod on the node
Used Memory	Bytes	Memory usage of the pod on the node

Figure 3-230 Node network metrics

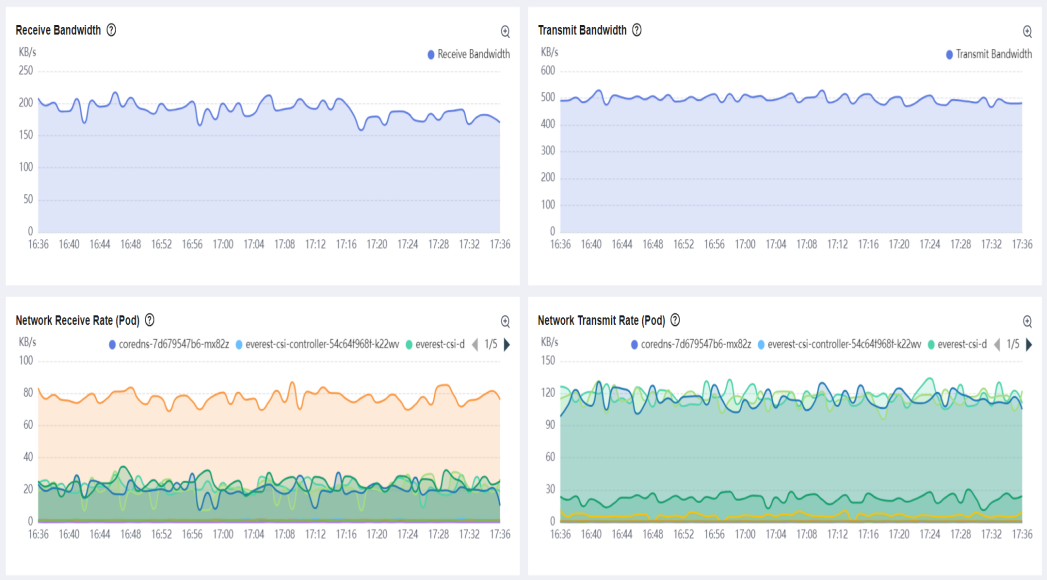


Table 3-298 Node network metrics

Metric	Unit	Description
Network Receive Rate	Bytes/s	Number of bytes received by the node per second

Metric	Unit	Description
Network Transmit Rate	Bytes/s	Number of bytes transmitted by the node per second
Network Receive Rate (Pod)	Bytes/s	Number of bytes received by the pod on the node per second
Network Transmit Rate (Pod)	Bytes/s	Number of bytes sent by the pod on the node per second
Rate of Received Packets	Count/s	Number of packets received by the pod on the node per second
Packet Transmit Rate	Count/s	Number of packets sent by the pod on the node per second
Packet Loss Rate (Receive)	Count/s	Number of packets not received by the pod on the node per second
Packet Loss Rate (Transmit)	Count/s	Number of packets sent from the pod but not received by the recipient on the node per second

Figure 3-231 Node disk metrics

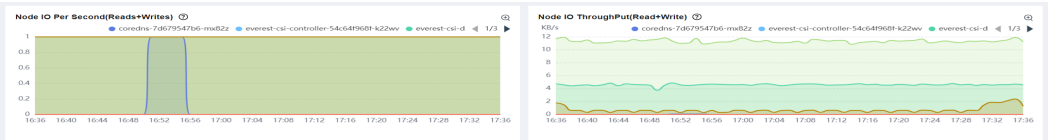


Table 3-299 Node disk metrics

Metric	Unit	Description
Node Disk I/O Rate (Read + Write)	Count/s	Number of read/write operations performed by a node disk per second
Node Disk Throughput (Read + Write)	Bytes/s	Number of bytes read from and written to the disk per second by the pod on the node

Metric List

The following table lists the metrics used in the node view.

Table 3-300 Node metrics

Metric	Type	Description
kube_pod_container_resource_limits	Gauge	The number of requested limit resources by a container
kube_pod_status_phase	Gauge	The pods current phase
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
container_network_transmit_bytes_total	Counter	Cumulative count of bytes transmitted
container_network_receive_packets_total	Counter	Cumulative count of packets received
container_network_transmit_packets_total	Counter	Cumulative count of packets transmitted
container_network_transmit_packets_dropped_total	Counter	Cumulative count of packets dropped while transmitting
container_fs_reads_total	Counter	Cumulative count of reads completed
container_fs_writes_total	Counter	Cumulative count of writes completed
container_fs_reads_bytes_total	Counter	Cumulative count of bytes read
container_fs_writes_bytes_total	Counter	Cumulative count of bytes written

3.9.3.8.7 Node Pool View

The node pool view displays the usage and allocation of node pool resources, helping you monitor the load status of node pools.

Metric Description

The following table lists the metrics displayed by the node pool view.

Figure 3-232 Node pool resource metrics

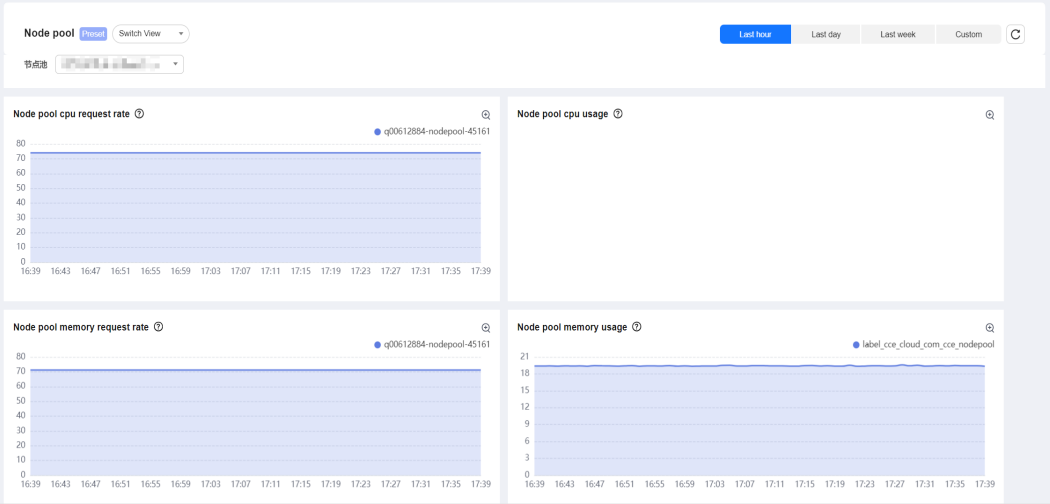


Table 3-301 Metrics

Metric	Unit	Description
Node Pool CPU Allocation Rate	%	Percentage of pod CPU requests of all nodes in a node pool
Node Pool CPU Usage	%	Percentage of used CPU cores of all nodes in a node pool
Node Pool Memory Allocation Rate	%	Percentage of pod memory requests of all nodes in a node pool
Node Pool Memory Usage	%	Percentage of used memory of all nodes in a node pool
Node Count Trend	Count	Number of nodes in a node pool

Metric List

The following table lists the metrics used in the node pool view.

Table 3-302 Metrics

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	Number of resources requested by the pod container

3.9.3.8.8 GPU View

GPU resource metrics are used to measure GPU performance and usage, including the GPU usage, temperature, and GPU memory, helping you monitor the GPU running.

Metric Description

Figure 3-233 GPU resource metrics

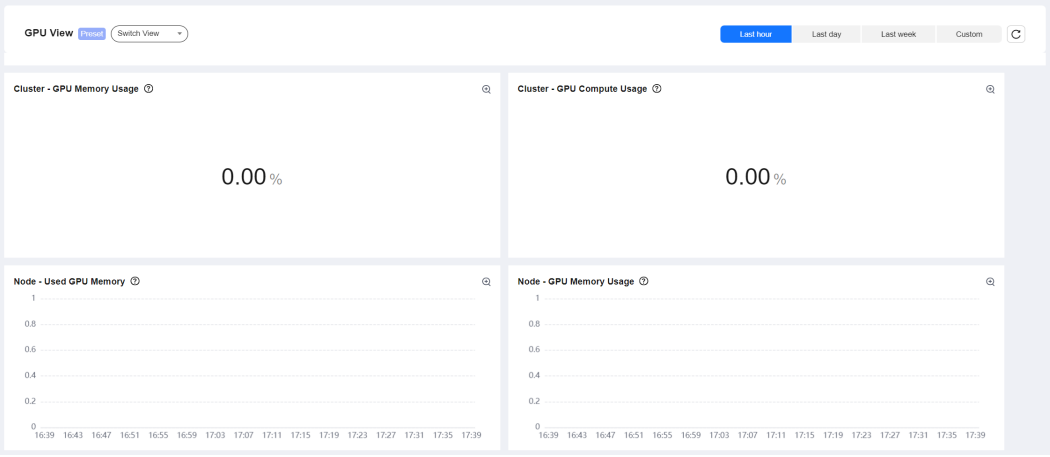


Table 3-303 GPU resource metrics

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: Total compute usage of the cluster/Total compute of the cluster
Node - Used GPU Memory	Bytes	GPU memory used by the node
Node - GPU Compute Usage	%	GPU compute usage of each node Formula: Total compute used by containers on the node/Total compute of containers on the node
Node - GPU Memory Usage	%	GPU memory usage of each node Formula: Total used GPU memory of containers on the node/Total GPU memory of containers on the node
GPU - Used GPU Memory	Bytes	GPU memory usage of each GPU Formula: Total used GPU memory of containers on the GPU/Total GPU memory of the GPU
GPU - GPU Compute Usage	%	Compute usage of each GPU Formula: Total compute used by containers on the GPU/Total compute of the GPU
GPU - Temperature	°C	Temperature of each GPU
GPU - Memory Clock	Hz	Memory clock of each GPU
GPU - PCIe Bandwidth	Bytes/s	PCIe bandwidth of each GPU

Metric List

The following table lists the metrics used in the GPU view.

Table 3-304 GPU metrics

Metric	Type	Description
cce_gpu_gpu_utilization	Gauge	GPU compute usage

Metric	Type	Description
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's 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

3.9.3.8.9 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, helping you monitor the GPU running.

Metric Description

Figure 3-234 xGPU resource metrics

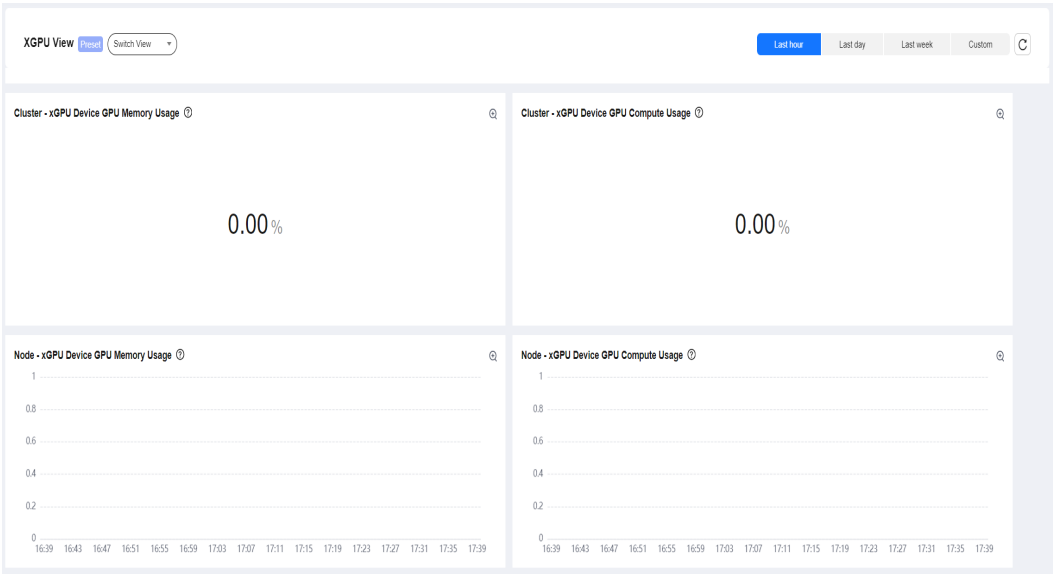


Table 3-305 xGPU resource metrics

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	Count	Number of xGPU devices on the node
Node - Allocated GPU Memory of xGPU Devices	Bytes	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	Bytes	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

Metric	Unit	Description
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	Count	Number of xGPU devices created on the GPU
GPU - Scheduling Policy	Number	• **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	Count	Number of unhealthy xGPU devices on the GPU
Allocated Container GPU Memory	Bytes	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
Used Container GPU Memory	Bytes	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 table lists the metrics used in the xGPU view.

Table 3-306 xGPU metrics

Metric	Type	Description
xgpu_memory_total	Gauge	Total xGPU memory
xgpu_memory_used	Gauge	Used xGPU memory

Metric	Type	Description
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.

3.9.3.8.10 CoreDNS View

The CoreDNS view displays the request, response, and cache status for load domain name resolution.

Metric Description

The following table lists the metrics displayed by the CoreDNS view.

Figure 3-235 CoreDNS view

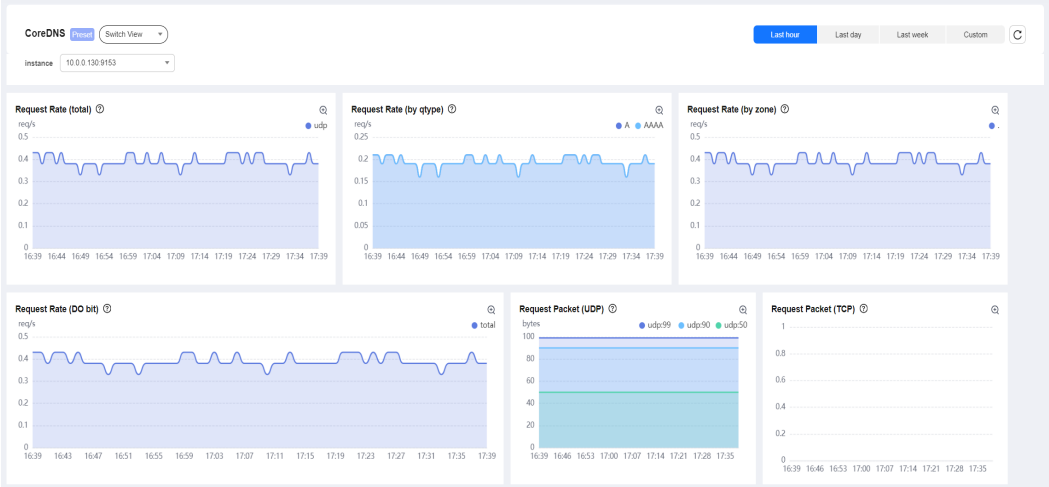


Table 3-307 CoreDNS metrics

Metric	Unit	Description
Request Rate	Requests/s	Number of requests received by CoreDNS per second
Request Rate (by qtype)	Requests/s	Number of requests received by CoreDNS per second for each resolution type
Request Rate (by zone)	Requests/s	Number of requests received by CoreDNS per second for each zone
Request Rate (DO bit)	Requests/s	Number of requests received by CoreDNS per second. Only requests that contain the DO bit are counted.
Request Packet (UDP)	Bytes	Size of each UDP packet received by CoreDNS. Only requests at the 99th percentile, 90th percentile, and 50th percentile are counted.
Request Packet (TCP)	Bytes	Size of each TCP packet received by CoreDNS. Only requests at the 99th percentile, 90th percentile, and 50th percentile are counted.
Response Rate (by rcode)	Responses/s	Number of responses sent by CoreDNS for each response code per second
Response Latency	ms	CoreDNS response time at the 99th percentile, 90th percentile, and 50th percentile
Response Packet (UDP)	Bytes	Size of each UDP packet sent by CoreDNS at the 99th percentile, 90th percentile, and 50th percentile
Response Packet (TCP)	Bytes	Size of each TCP packet sent by CoreDNS at the 99th percentile, 90th percentile, and 50th percentile
Cached DNS Records	Count	Number of DNS records cached by CoreDNS
Cache (hitrate)	Requests/s	Number of hit requests in the CoreDNS cache per second

Metric List

The following table lists the metrics used in CoreDNS View.

Table 3-308 CoreDNS metrics

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, per zone
coredns_dns_request_do_count_total	Counter	Cumulative count of requests with the DO bit set
coredns_dns_do_requests_total	Counter	Number of requests with the DO bit set
coredns_dns_request_size_bytes_bucket	Histogram	Number of CoreDNS request bytes
coredns_dns_response_rcode_count_total	Counter	Cumulative count of response status codes
coredns_dns_responses_total	Counter	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	Number of cache hits

3.9.3.8.11 PVC View

The PVC view displays the PV/PVC status, usage, and usage statistics in a fixed period.

Metric Description

The following table lists the metrics displayed by the PVC view.

Table 3-309 PVC metrics

Metric	Unit	Description
PV Status	/	Current status of the PV
PVC Status	/	Current status of the PVC

Metric	Unit	Description
Used PVC	Bytes	Used and Idle PVCs
PVC Usage	%	Percentage of used PVC space
Used PVC Inodes	Count	Used and Idle PVC inodes
PVC Inodes Usage	%	Percentage of inodes used by PVCs
Hourly PVC Usage	Bytes/s	Average PVC usage per hour
Daily PVC Usage	Bytes/s	Average PVC usage per day
Weekly PVC Usage	Bytes/s	Average PVC usage per week
Volumes Full in Week Based on Daily Use Rate	Bytes	Average PVC usage per week

Metric List

The following table lists the metrics used in the PVC view.

Table 3-310 PVC metrics

Metric	Type	Description
kubelet_volume_stats_inodes_used	Gauge	Number of used inodes in the volume
kubelet_volume_stats_inodes	Gauge	Maximum number of inodes in the volume
kubelet_volume_stats_capacity_bytes	Gauge	Capacity in bytes of the volume
kubelet_volume_stats_available_bytes	Gauge	Number of available bytes in the volume
kubelet_volume_stats_used_bytes	Gauge	Number of used bytes in the volume
kube_persistentvolume_status_phase	Gauge	PV status
kube_persistentvolume-claim_status_phase	Gauge	PVC status

3.9.3.8.12 Kubelet View

The kubelet is the agent that runs on each node in a cluster. The Kubelet view allows you to monitor the running of clusters.

Metric Description

The following table lists the metrics displayed by the Kubelet view.

Table 3-311 Kubelet metrics

Metric	Unit	Description
Running kubelets	Count	Number of running kubelets in the cluster
Running Pods	Count	Number of pods running on the node where Kubelet resides
Running Containers	Count	Number of running containers on the node where Kubelet resides
Actual Volumes	Count	Actual number of volumes on the node where Kubelet resides
Expected Volumes	Count	Expected number of volumes on the node where Kubelet resides
Configuration Errors	Count	Number of incorrect Kubelet configurations on the node where Kubelet resides
Operation Rate	Count/s	Number of operations performed by Kubelet per second
Operation Error Rate	Count/s	Number of failed operations performed by Kubelet per second
Operation Latency	Second	Latency of different Kubelet operations
Pod Startup Rate	Count/s	Number of times that Kubelet executes pod start operations per second
Pod Startup Latency (P99)	Second	Latency of 99% of pod start operations performed by Kubelet
Storage Operation Rate	Count/s	Number of storage-related operations performed by Kubelet per second
Storage Operation Error Rate	Count/s	Number of failed storage-related operations performed by Kubelet per second
Storage Operation Latency (P99)	Second	Latency of 99% of storage-related operations performed by Kubelet
Cgroup Manager Operation Rate	Count/s	Number of destroy or update operations performed by Kubelet per second

Metric	Unit	Description
Cgroup Manager Operation Latency (P99)	Second	Latency of 99% of destroy or update operations performed by Kubelet
PLEG Relist Rate	Count/s	Number of relisting operations in PLEG per second
PLEG Relist Interval (P99)	Second	Interval between 99% of relisting operations in PLEG
PLEG Relist Latency (P99)	Second	Latency of 99% of relisting operations in PLEG
RPC Rate	Count/s	Number of RPC requests with different status codes
Request Latency (P99)	Second	Latency of 99% of requests with different methods
Used Memory	Bytes	Memory used by Kubelet
Used CPU	Bytes	CPU used by Kubelet
Goroutines	Count	Number of goroutines

Metric List

The following table lists the metrics used in Kubelet View.

Table 3-312 Kubelet metrics

Metric	Type	Description
storage_operation_errors_total	Counter	Number of errors in storage operations
storage_operation_duration_seconds_count	Counter	Number of storage operations
storage_operation_duration_seconds_bucket	Histogram	Duration for each storage operation
kubelet_pod_start_duration_seconds_count	Counter	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

Metric	Type	Description
kubelet_runtime_operations_errors_total	Counter	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	Number of volumes in Volume Manager
kubelet_running_containers	Gauge	Number of containers currently running
kubelet_running_pods	Gauge	Number of pods currently running
kubelet_node_name	Gauge	Node name. The value is always **1**.
kubelet_runtime_operations_total	Counter	Number of total runtime operations of each type
kubelet_cgroup_manager_duration_seconds_count	Counter	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	Number of relisting operations in PLEG
kubelet_pleg_relist_interval_seconds_bucket	Histogram	Interval between relisting operations in PLEG
kubelet_pleg_relist_duration_seconds_bucket	Histogram	Duration for relisting pods in PLEG
rest_client_requests_total	Counter	Number of HTTP requests, partitioned by status code, method, and host
rest_client_request_duration_seconds_bucket	Histogram	Number of HTTP requests, partitioned by status code, method, and host
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	Number of goroutines

3.9.3.8.13 Prometheus Server View

Prometheus Server 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 Server view displays some built-in metrics provided by Prometheus, which can be used to monitor and measure system performance and status.

Metric Description

The following table lists the metrics displayed by the Prometheus Server view.

Figure 3-236 Prometheus Server view

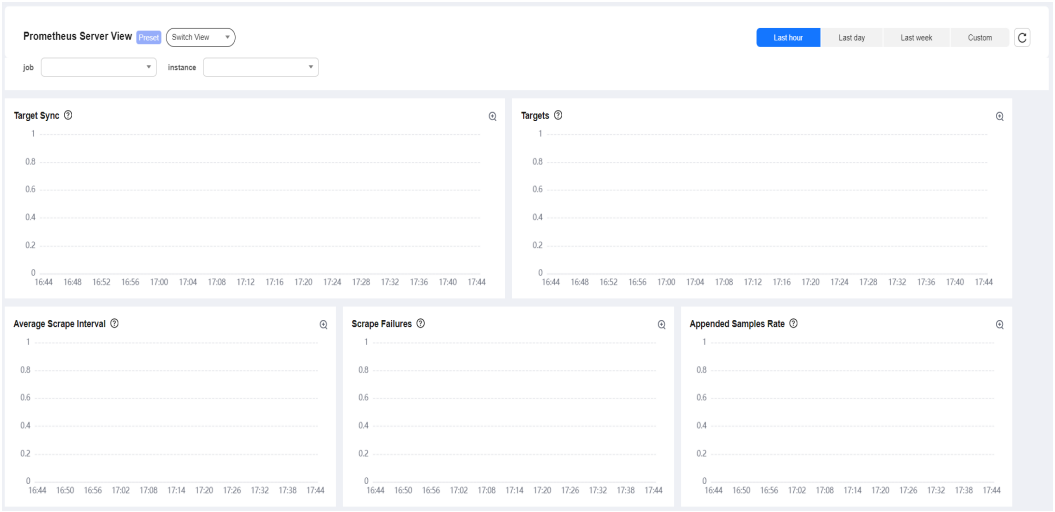


Table 3-313 Prometheus Server metrics

Metric	Unit	Description
Target Sync	Second	Target metric collection latency
Targets	Count	Number of targets collected by Prometheus
Average Scrape Interval	Second	Interval for collecting metrics
Scrape Failures	Count	Number of collection failures
Appended Samples Rate	Count	Rate at which samples are added to the head
Head Series	Count	Number of series in the head
Head Chunks	Count	Number of head blocks
Query Rate	Count/s	Number of Prometheus queries per second

Metric	Unit	Description
P90 Query Duration	Second	Time required 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	Bytes/s	Remote write rate
Current Shards	Count	Number of shards used for parallel sending to the remote storage
Max Shards	Count	Maximum number of shards that the queue is allowed to run
Min Shards	Count	Minimum number of shards that the queue is allowed to run
Desired Shards	Count	Number of shards that the queue wants to run based on the percentage of input samples to output samples
Shard Capacity	Count	Capacity of each shard of the queue used for parallel sending to the remote storage
Pending Samples	Count	Capacity of each shard of the queue used for parallel sending to the remote storage
Current TSDB Segment	Segment	WAL segment index that TSDB is currently writing to
Current Segment of Remote Write	Segment	The current segment from which the WAL watcher is reading records
Sample Discard Rate	Count/s	The rate at which samples are discarded after being read from WAL before being sent via remote write
Sample Failure Rate	Count/s	Rate of samples that failed to be sent to remote storage due to unrecoverable errors
Sample Retry Rate	Count/s	Rate of samples that failed to be sent to remote storage due to recoverable errors and were resent
Retry Rate of Enqueuing	Count/s	Retry rate of enqueuing failed due to full shard queue

Metric List

The following table lists the metrics used in the Prometheus Server view.

Table 3-314 Prometheus Server metrics

Metric	Type	Description
prometheus_target_sync_length_seconds_sum	Summary	Collection latency of different targets
prometheus_sd_discovered_targets	Gauge	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	Number of collection failures
prometheus_tsdb_head_samples_appended_total	Counter	Total number of samples added to the head
prometheus_tsdb_head_series	Gauge	Number of series in the head block
prometheus_tsdb_head_chunks	Gauge	Number of chunks in the head block
prometheus_engine_query_duration_seconds_count	Counter	Number of queries
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	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

Metric	Type	Description
prometheus_remote_storage_shards_desired	Gauge	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	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	The 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
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	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	Number of retries upon enqueuing failed due to full shard queue

3.9.3.8.14 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

The following table lists the metrics displayed by the Prometheus Agent view.

Table 3-315 Prometheus Agent metrics

Metric	Unit	Description
CPU Usage	%	Average CPU usage of pods
Memory Usage	%	Average memory usage of pods
Remote Write Rate	Bytes/s	Number of bytes remotely written per second
Average Remote Write Duration	Second	Average time consumed by remote writes
Bytes Pending for Remote Write	Bytes	Number of pending bytes during a remote write
Packets Discarded for Remote Write per Second	Count	Number of packets discarded per second during a remote write
Failed Remote Write Requests per Second	Count	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	Count	Number of retries of the remote write per second
Scrapers	Count	Number of scrapers
Collections per Second	Count	Collections per second
Average Collection Duration	Second	Average collection duration
Failed Reads During Collection per Second	Count	Number of read errors during scrapes per second
Failed Writes During Collection per Second	Count	Number of write errors during scrapes per second
Collections with Size Exceeding the Threshold per Second	Count	Collections with size exceeding the threshold per second
Read Rate of Sending Queue	Bytes/s	Number of bytes read by a sending queue per second
Write Rate of Sending Queue	Bytes/s	Number of bytes written to a sending queue per second
Pending Size of Sending Queue	Bytes	Number of bytes suspended in a sending queue
Blocks Read per Second	Count	Number of blocks read by a sending queue per second

Metric	Unit	Description
Blocks Written per Second	Count	Number of blocks written to a sending queue per second
Discarded Blocks per Second	Count	Number of dropped blocks in a sending queue per second

Metric List

The following table lists the metrics used in the Prometheus Agent view.

Table 3-316 Prometheus Agent metrics

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	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	Number of collected shards
vm_promscrape_scrapes_total	Counter	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	Number of read errors during scrapes
vm_promscrape_conn_write_errors_total	Counter	Number of write errors during scrapes

Metric	Type	Description
vm_promscrape_max_scrape_size_exceeded_errors_total	Counter	Number of failed scrapes due to the exceeded response size
vm_persistentqueue_bytes_read_total	Counter	Number of bytes read by a send queue
vm_persistentqueue_bytes_written_total	Counter	Number of bytes written to a send queue
vm_persistentqueue_bytes_pending	Gauge	Number of pending bytes in a send queue
vm_persistentqueue_blocks_read_total	Counter	Number of blocks read by a send queue
vm_persistentqueue_blocks_written_total	Counter	Number of blocks written to a send queue
vm_persistentqueue_blocks_dropped_total	Counter	Number of dropped blocks in a send queue

3.9.4 Logging

3.9.4.1 Overview

Kubernetes logs allow you to locate and rectify faults. This section describes how you can manage Kubernetes logs in the following ways:

- Use Cloud Native Logging to collect application logs and report them to LTS, which provides log statistics and analysis. For details, see **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.
- (Not recommended) Connect CCE to AOM. For details, see **3.9.4.2.2 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 **3.9.4.3 Collecting Control Plane Component Logs** and **3.9.4.4 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 **3.9.4.5 Collecting Kubernetes Events**.

Comparison Between ICAgent and Cloud Native Logging

Table 3-317 Comparison between ICAgent and Cloud Native Logging

Collection Tool	ICAgent	Cloud Native Logging

Log Storage Location	LTS	AOM	LTS	AOM
Content to Be Collected	- Container standard output logs - Container file logs - Node logs - Kubernetes events	- Container standard output logs - Container file logs	- Container standard output 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 **3.9.4.2.2 Collecting Container Logs Using ICAgent (Not Recommended)**.	Create a policy on the **Logging** page. For details, see **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.	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 standard output logs in JSON format		Up to 4,096 logs collected based on log policies on each node	-

3.9.4.2 Collecting Container Logs

3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging

An add-on (**3.14.18 Cloud Native Logging**) based on Fluent Bit and OpenTelemetry is provided for log and Kubernetes event collection. It supports CRD-based log collection policies, and collects and forwards standard output logs, container file logs, node file logs, and Kubernetes events of containers in a cluster.

Constraints

- A maximum of 50 log rules can be created for each cluster.
- Cloud Native Logging 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 standard output log cannot be in multiple lines. (This does not apply to Cloud Native Logging v1.3.0 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**.

Log Collection

Step 1 Enable log collection.

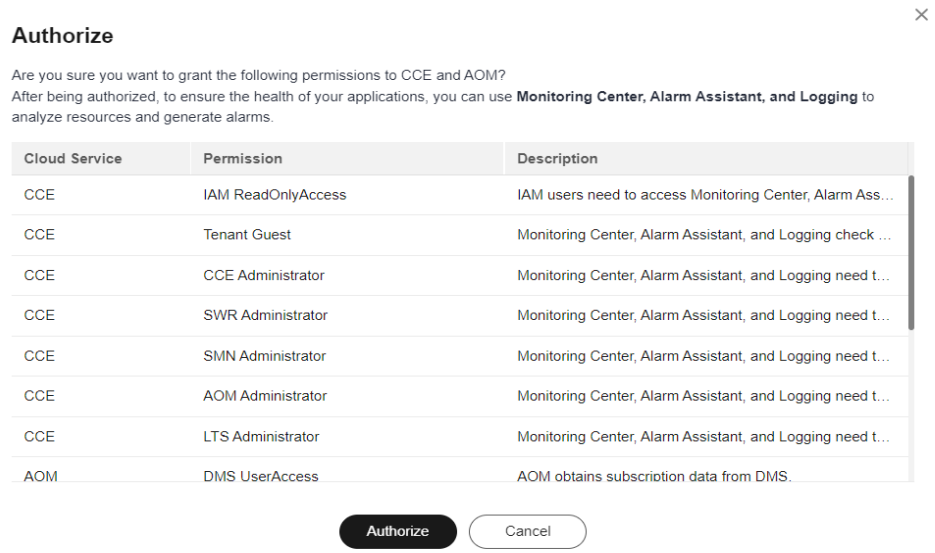
Enabling log collection during cluster creation

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 Logging**.
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, and standard output logs in all namespaces will be reported to LTS.
   - Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.
5. Click **Next: Confirm Configuration** in the lower right corner. On the displayed page, click **Submit**.

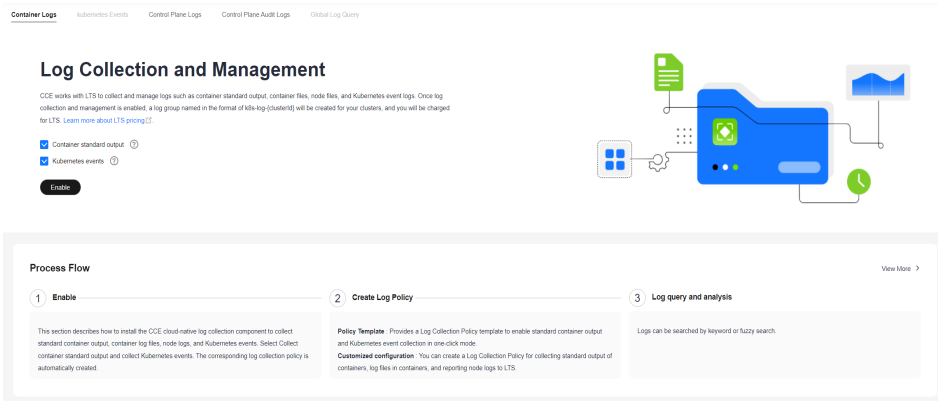
Enabling log collection for an existing cluster

1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. (Optional) If you are not authorized, obtain required permissions first.

   In the displayed dialog box, click **Authorize**.



3. Click **Enable** and wait for about 30 seconds until the log page is automatically displayed.



- Standard output logs: A log collection policy named **default-stdout** will be created, and standard output logs in all namespaces will be reported to LTS.

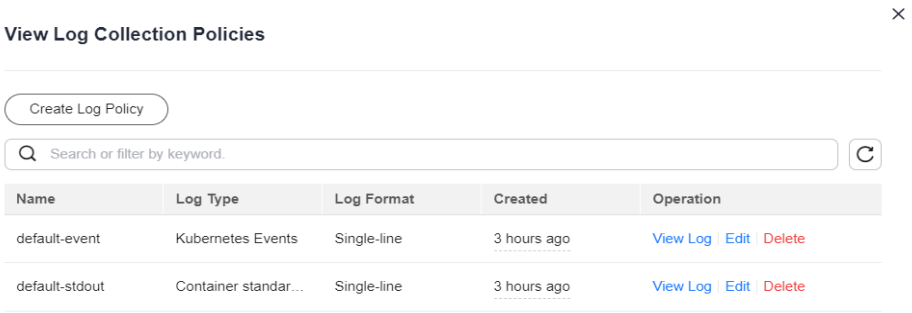
- Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.

Step 2 View and configure log collection policies.

1. On the CCE console, click the cluster name to access the details page. 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 3-237 Viewing log collection policies



   If you select **Container standard output** and **Kubernetes events** while enabling logging, two log collection policies will be created, and the collected logs will be reported to the default log group and log streams.

   – default-stdout

     Standard output logs are collected to the default log stream **stdout-***{Cluster ID}* in the default log group **k8s-logs-***{Cluster ID}*.

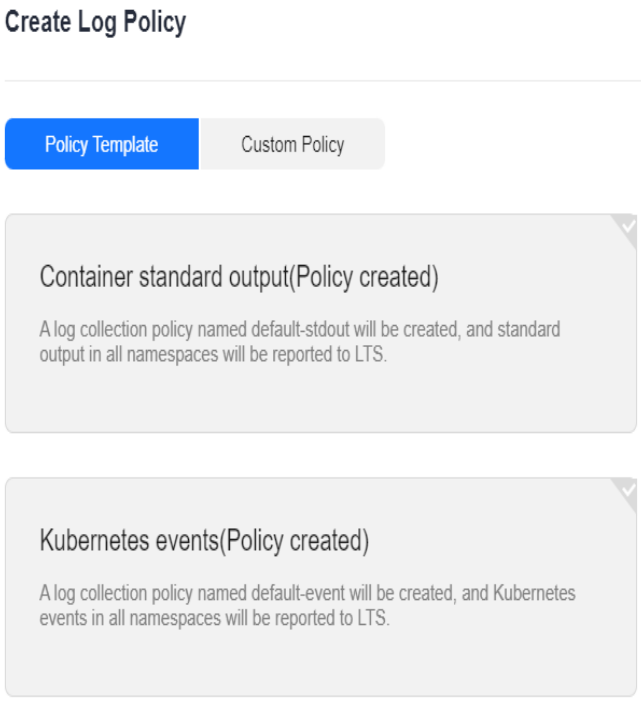
   – default-event

     Kubernetes events are collected to the default log stream **event-***{Cluster ID}* in the default log group **k8s-logs-***{Cluster ID}*.

3. Click **Create Log Policy** and configure parameters as required.

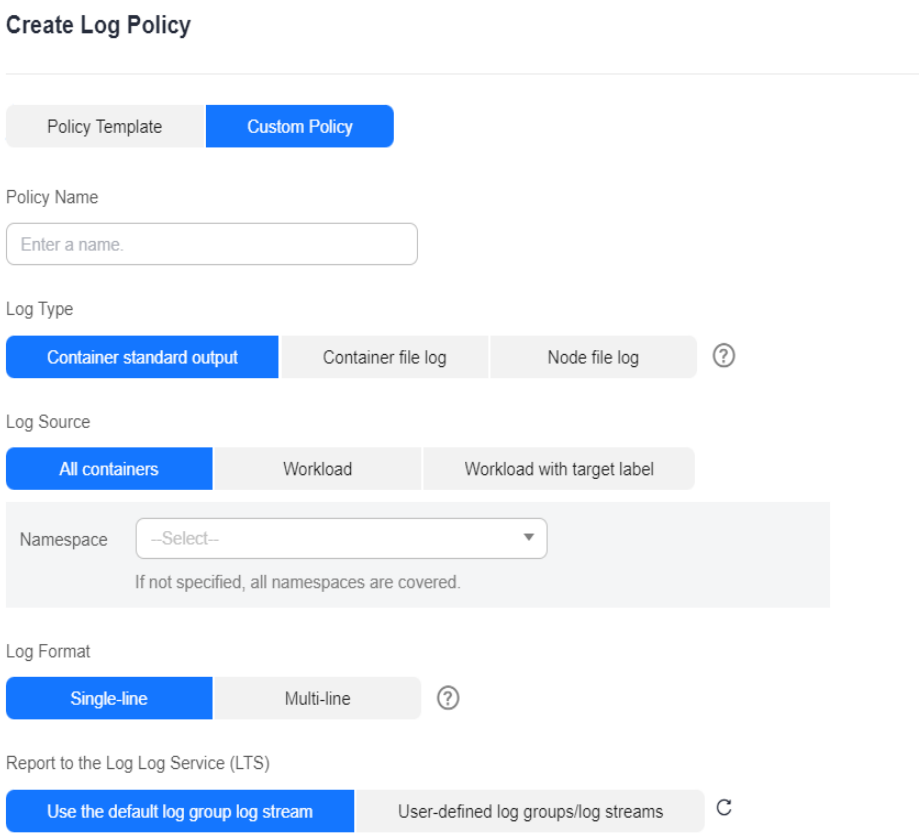
   Policy Template: If you do not select **Container standard output** and **Kubernetes events** while enabling logging or delete their log collection policies, you can use this option to create default log collection policies.

Figure 3-238 Policy template



Custom Policy: You can use this option to create custom log collection policies.

Figure 3-239 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**, any log files with the extension .log in all directories in the **/var/logs** directory will be reported.
  - 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 3-318 Custom policy parameters

Parameter	Description		
Log Type	**Container standard output**: used to collect container standard output 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.
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)`

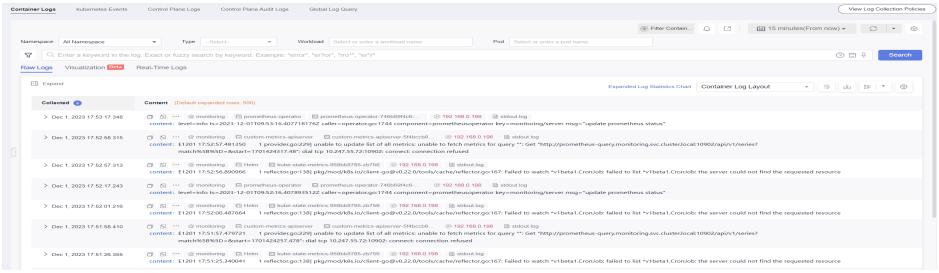
Report to LTS	This parameter is used to configure the log group and log stream for log reporting. – **Default log groups/log streams**: The default log group (**k8s-log-*{Cluster ID}***) and default log stream (**stdout-*{Cluster ID}***) are automatically selected. – **Custom log groups/log streams**: 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 policy based on a 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**

4. Click **Edit** to modify an existing log collection policy.
5. Click **Delete** to delete an existing log collection policy.

Step 3 View the logs.

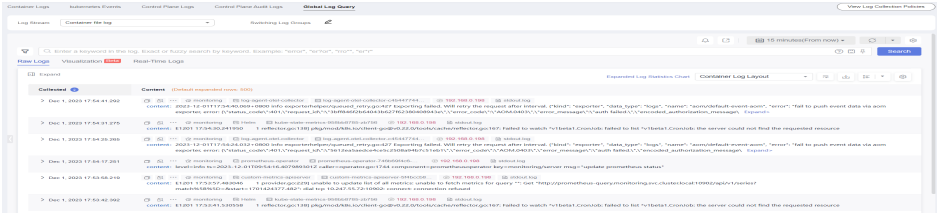
1. On the CCE console, click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. View different types of logs:
   - **Container Logs**: displays all logs in the default log stream **stdout-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*. You can search for logs by workload.

     Figure 3-240 Querying container logs

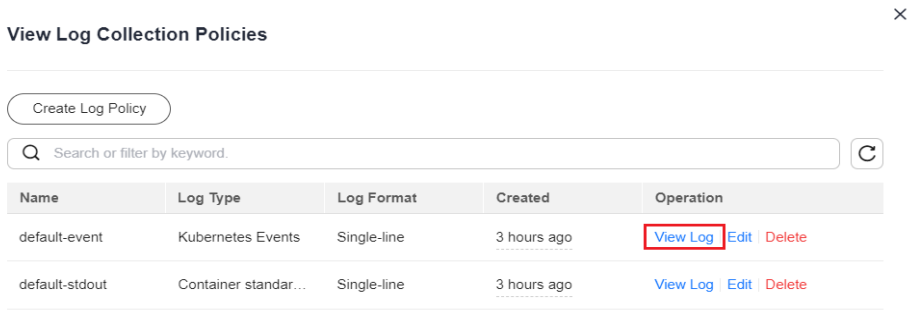
   - **Kubernetes Events**: displays all Kubernetes events in the default log stream **event-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   - **Control Plane Logs**: displays all logs of components on the control plane in the default log stream *{Component name}*-*{Cluster ID}* 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 3-241 Global log query

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 3-242 Viewing logs



----End

3.9.4.2.2 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

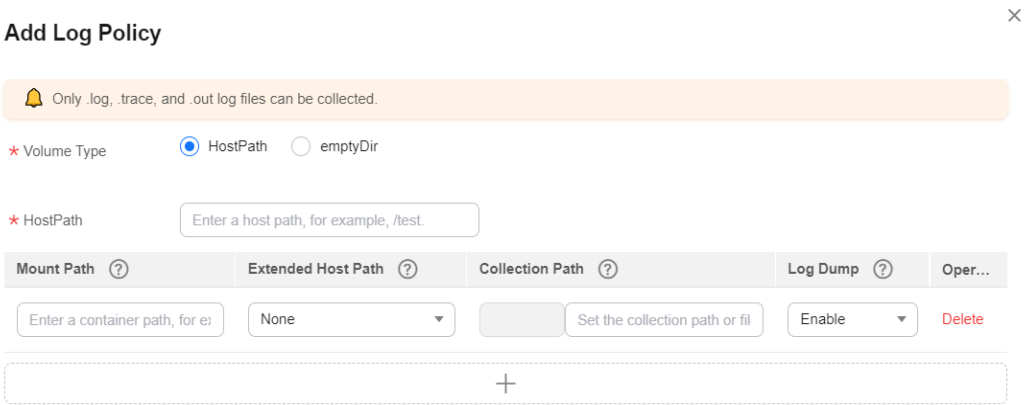
You can add a policy to collect logs using ICAgent for a workload.

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 3-243 Adding a log policy



Step 3 Set **Volume Type** to **hostPath** or **emptyDir**.

Table 3-319 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 v5.12.22 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 3-320 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 v5.12.22 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 standard output of a container.

```
# 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)
```

3.9.4.3 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 3-321 Control plane components

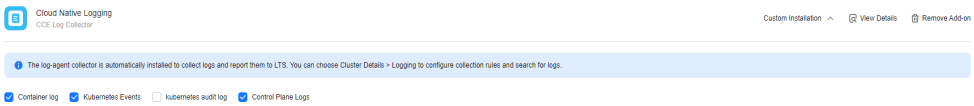
Log Type	Component	Log Stream	Description
Control plane component logs	kube-apiserver	kube-apiserver-{{clusterID}}	It exposes Kubernetes APIs. For more information, see **kube-apiserver**.
	kube-controller-manager	kube-controller-manager-{{clusterID}}	It manages controllers and embeds the core control 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**. On the displayed page,configure the parameters and click **Next: Select Add-on**.
3. On the displayed page, select **Cloud Native Logging** and click **Next: Add-on Configuration**.
4. On the displayed page, select **Control Plane Logs** for **Cloud Native Logging**.

   Figure 3-244 Enabling control plane logging during cluster creation

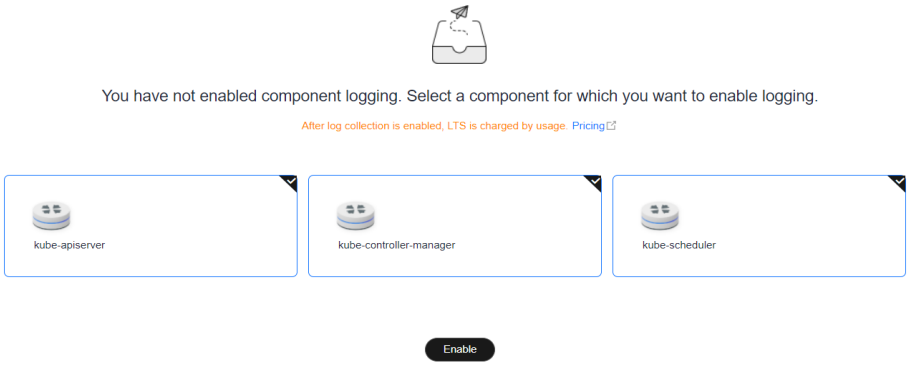
   

5. Click **Next: Confirm configuration**.

Enabling control plane logging for an existing cluster

1. Log in to the CCE console and click the cluster name to access the details page. 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 3-245 Selecting control plane components

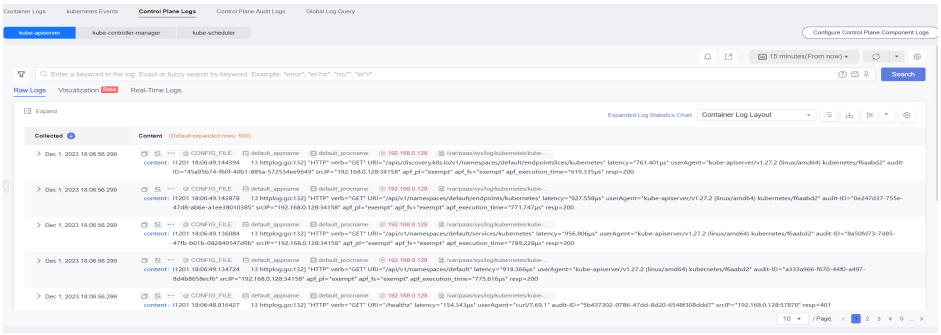
   

Viewing Control Plane Component Logs of the Target Cluster

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 details page. 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 operations, see **LTS User Guide**.

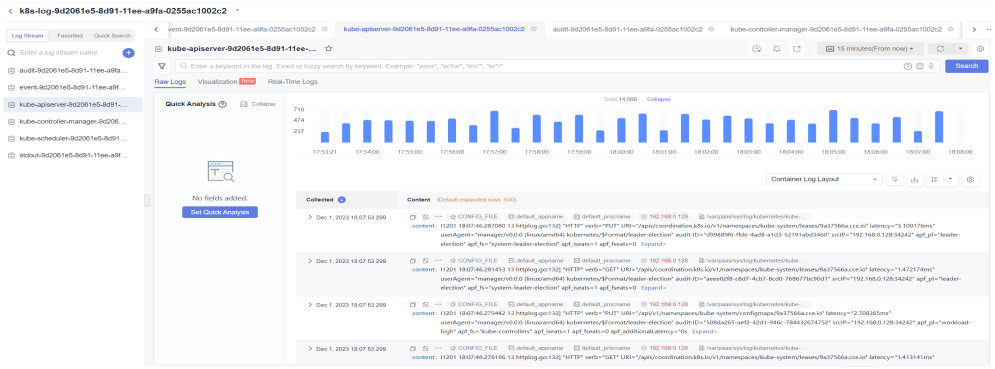
Figure 3-246 Viewing control plane component logs



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**.

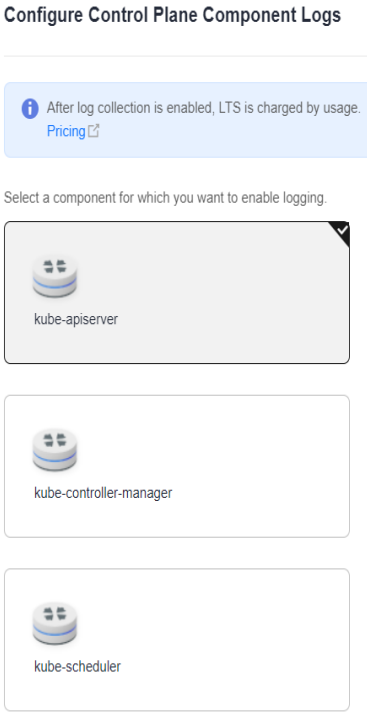
Figure 3-247 Viewing control plane component logs on the LTS console



Disabling Control Plane Logging

1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. On the **Control Plane Logs** tab, click **Configure Control Plane Component Logs** in the upper right corner and modify the log settings.

Figure 3-248 Disabling control plane logging



3. Determine whether to enable logging for each component and 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.

3.9.4.4 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

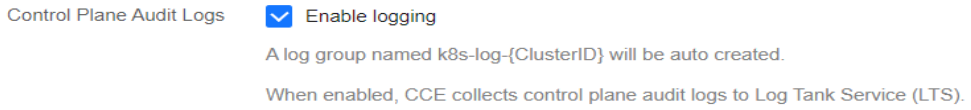
Table 3-322 Kubernetes audit logs

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

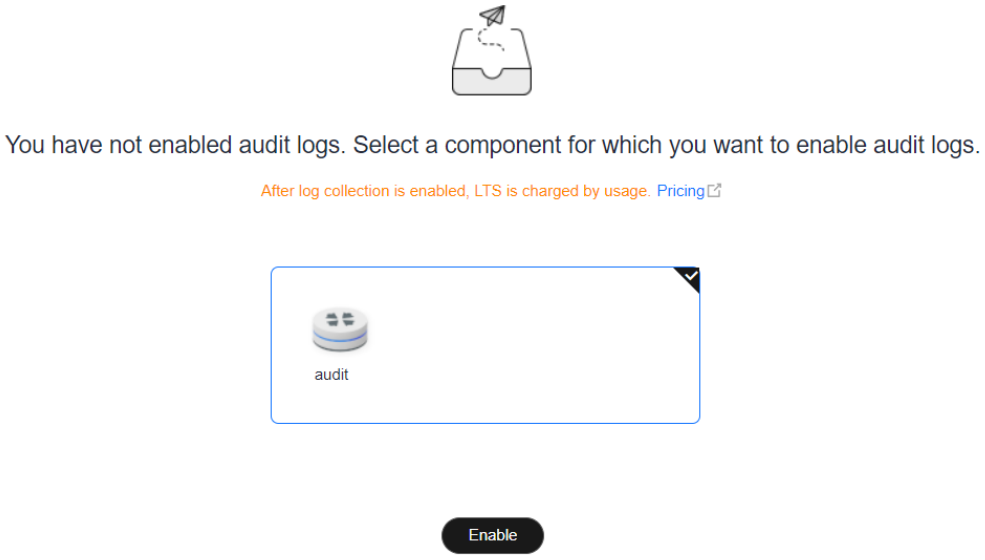
Enabling control plane audit logging during cluster creation

1. Log in to the CCE console.
2. From the top menu, click **Buy Cluster** and select a cluster type.
3. On the **Add-on Configuration** page, check the box of **Enable logging** for **Control Plane Audit Logs**.



Enabling control plane audit logging for an existing cluster

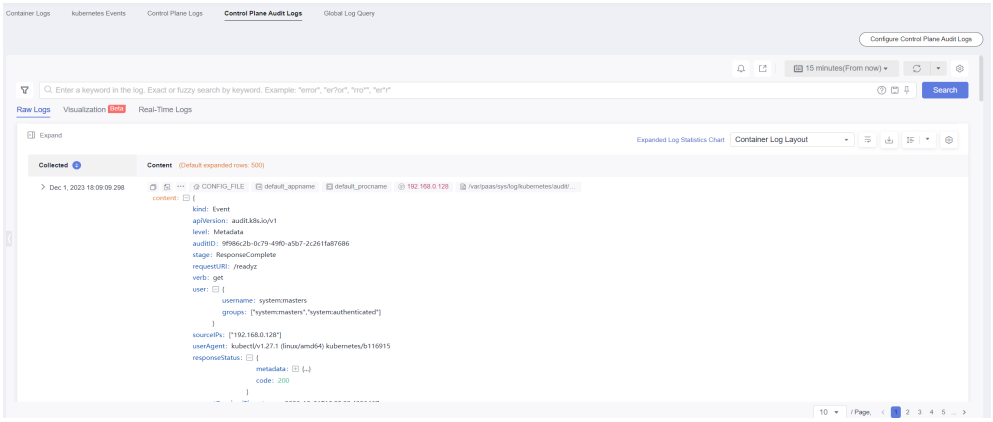
1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab, select the audit component, and click **Enable**.



Viewing Control Plane Audit Logs

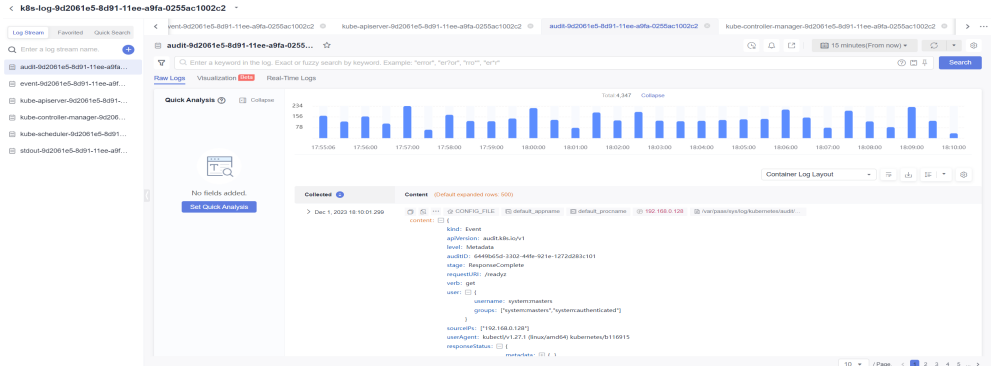
Viewing control plane audit logs on the CCE console

1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. Click the **Control Plane Audit Logs** tab to view audit logs in the cluster. For details about operations, see **LTS User Guide**.



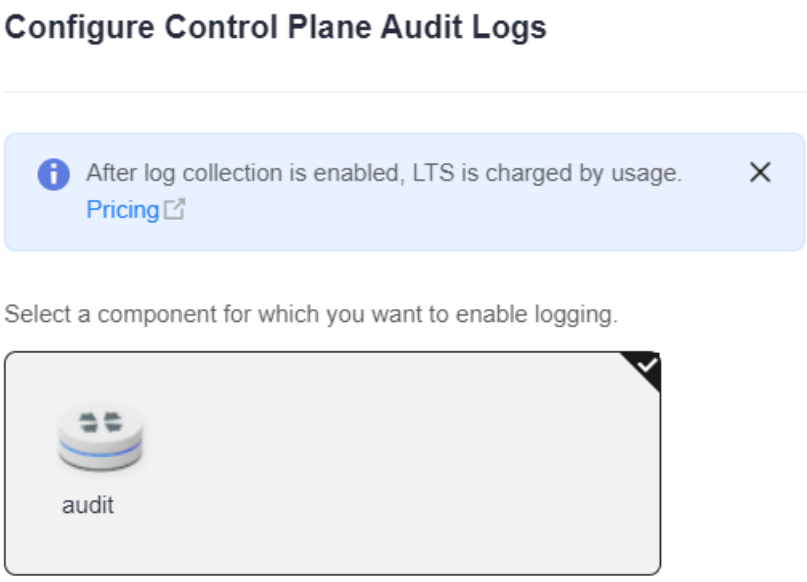
Viewing control plane audit logs on the TLS console

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**.



Disabling Control Plane Audit Logging

1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. On the **Control Plane Audit Logs** tab, click **Configure Control Plane Audit Logs** in the upper right corner and determine whether to enable 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.

3.9.4.5 Collecting Kubernetes Events

Cloud Native Logging works with LTS to collect and store Kubernetes events and works with AOM to generate alarms.

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**.

Reporting Kubernetes Events to LTS

Cloud Native Logging is not installed in a cluster.

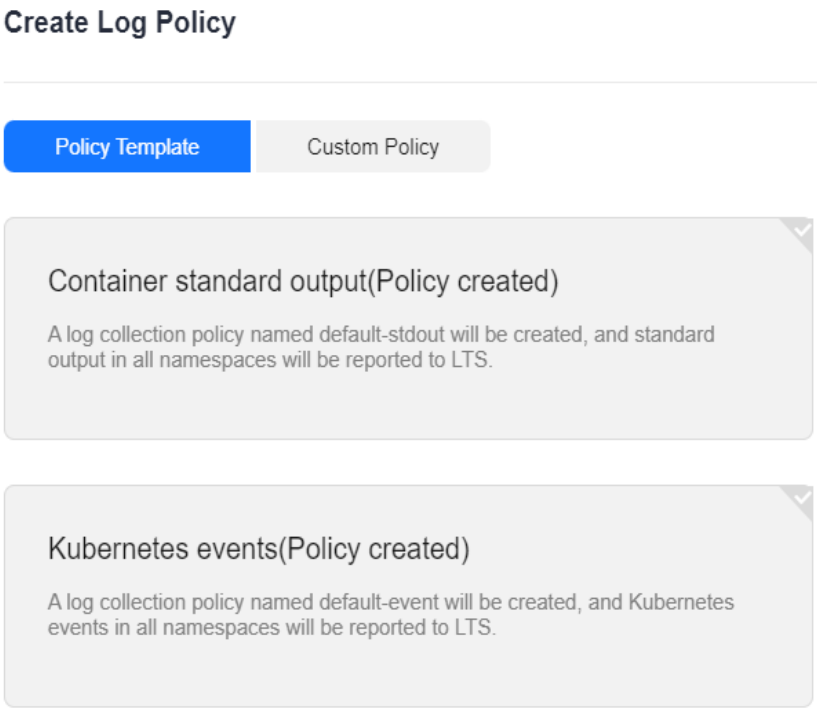
When installing Cloud Native Logging, you can select **Kubernetes events** to create a default log collection policy, so that this add-on collects all events and reports them to LTS. For details about the installation method, see **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.

Cloud Native Logging has been installed in a cluster.

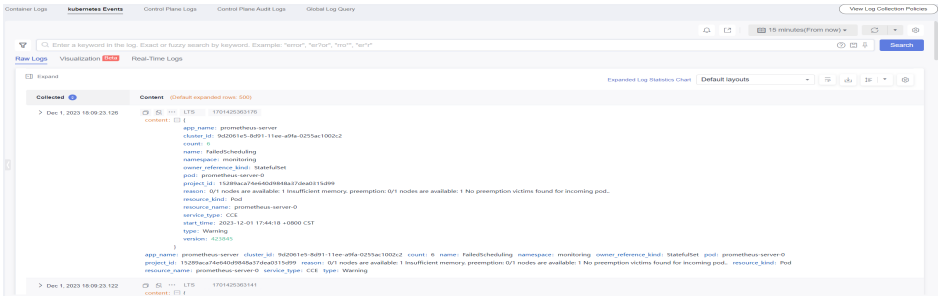
1. Log in to the CCE console and click the cluster name to access the details page. In the navigation pane, choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.
3. Click **Create Log Policy** and configure parameters as required.

Policy Template: If you do not select **Kubernetes events** while installing this add-on or delete the log collection policy, you can use this option to create a default log collection policy.



4. On the **Logging** page, select the log stream configured in the log collection policy to view the events reported to LTS.



Reporting Kubernetes Events to AOM

After Cloud Native Logging 1.3.2 is installed, all warning events and some normal events will be reported 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 Logging is installed, events will be reported to AOM by this add-on instead of the control plane component. After Cloud Native Logging 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 details page. 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 Blacklist** to add events that do not need to be reported to AOM to the blacklist.
- **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 Whitelist** to add the events to the whitelist.

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

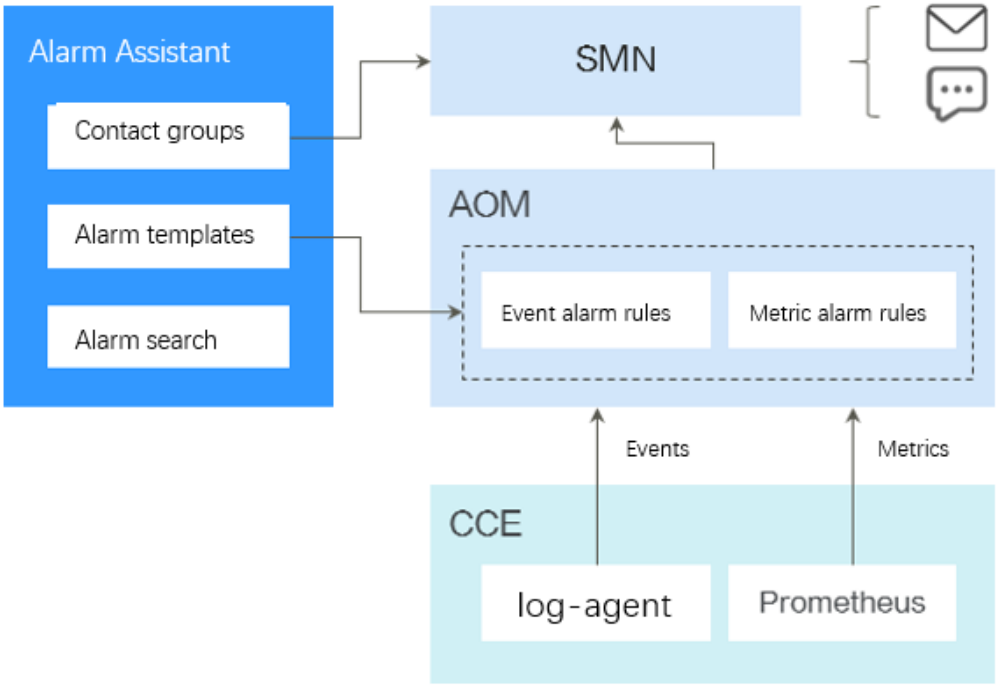
3.9.5 Alarm Assistant

3.9.5.1 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 Assistant that allows you to quickly configure and view common alarms (such as resource usage alarms).

Figure 3-249 Alarm Assistant architecture



- Alarm Assistant

  Based on the alarm capabilities of AOM, Alarm Assistant provides quick alarm search and configuration for clusters. You can use Alarm Assistant 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.

3.9.5.2 Configuring Alarms in Alarm Assistant

By using AOM, Alarm Assistant can promptly detect cluster faults and generate alarms for service stability. Alarm Assistant 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 Assistant. IAM users with the CCE ReadOnlyAccess permission can only view all resources.

Enabling Alarm Assistant

Alarm Assistant can be enabled for CCE standard clusters and CCE Turbo clusters.

Step 1 Click the name of the target cluster and choose **Alarm Assistant** in the navigation pane.

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.

Step 3 Click **OK**.

NOTE

Metric alarm rules can be created in Alarm Assistant 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 **3.9.3.2 Enabling Cluster Monitoring**.

The alarm rules that use the problem_gauge metric in **Table 3-323** depend on the CCE Node Problem Detector add-on (**3.14.4 CCE Node Problem Detector**). To use related alarm rules, ensure that CCE Node Problem Detector has been installed and is running normally.

Event alarms in **Table 3-323** can be reported only when Kubernetes event collection is enabled in Logging. For details, see **3.9.4.5 Collecting Kubernetes Events**.

----End

Configuring Alarm Rules

After Alarm Assistant 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 name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and click the **Alarm Rules** tab. Then configure and manage alarm rules.

By default, Alarm Assistant 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 3-323** lists default alarm rules.

----End

Table 3-323 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)

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	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)
	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

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
	Load balancer update failed	Check whether a load balancer is updated.	Event	Cloud Native Logging	N/A
	Pod OOM	Check whether OOM occurs on the pod.	Event	CCE Node Problem Detector (1.18.41 or later) Cloud Native Logging (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 running normally.	Metric	Cloud Native Cluster Monitoring	kube_persistentvolumeclaim_status_phase{phase=~"Failed|Pending|Lost"} > 0
	Abnormal Kubernetes PV	Check whether the PV is running normally.	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 Logging	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 Logging	TaskHung
	Incorrect node storage pool configuration	Check whether the node's EV and PV storage pools are correctly configured.	Event	Cloud Native Logging	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 Logging	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 sold out	Check whether the node pool resources are sufficient.	Event	Cloud Native Logging	NodePoolSoldOut
	Scale-out timed out	Check whether adding nodes to the node pool timed out.	Event	Cloud Native Logging	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 Logging	FailedToScaleUpGroup
	Node pool scale-in failed	Check whether an error occurred during a node pool scale-in.	Event	Cloud Native Logging	ScaleDownFailed
Cluster status rule set	Unavailable cluster	Check whether the cluster is available.	Event	Cloud Native Logging	N/A

Configuring Alarm Notification Recipients

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 configure contact groups to manage endpoints that have subscribed to alarm messages. After creating a contact group, associate alarm rule set with the group. When an alarm is triggered, the subscription endpoints in the contact group can receive the alarm messages.

Step 1 Log in to the CCE console.

Step 2 On the cluster list page, click the name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and click the **Contact Group** tab.

Step 4 Click **Create Contact Group** and configure parameters.

- **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 you do not specify **Alarm message display name**, the sender will be **username@example.com**. The display name of an alarm message can be changed after the 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. If the endpoint receives the request, confirm the request as prompted. After the confirmation is complete, the subscription endpoint changes to **Confirmed**.

Step 7 Click  to enable the contact group so that the contact group is bound to the alarm rule set.

NOTE

An alarm rule set can be bound to a maximum of five contact groups.

----End

Viewing Alarms

You can view the latest historical alarms on the **Alarm list** tab.

Step 1 Log in to the CCE console.

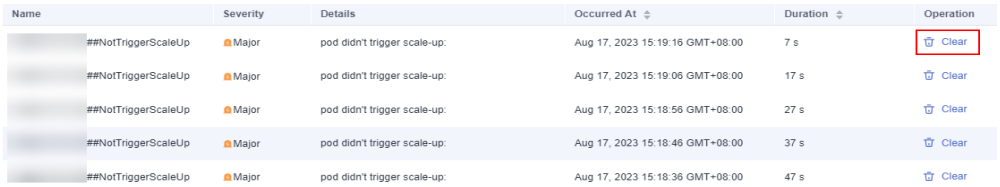
Step 2 On the cluster list page, click the name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and 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 you confirm that an alarm has been handled, click **Clear** in the **Operation** column. After the alarm is cleared, you can view it in the historical alarm list.

Figure 3-250 Querying alarms



----End

3.9.5.3 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 **3.9.3.2 Enabling Cluster Monitoring**.
- Some metric templates are created based on the problems reported by CCE Node Problem Detector (**3.14.4 CCE Node Problem Detector**). For details about these metrics, see **Table 3-323**. 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 details page.

Step 2 Choose **Alarm Assistant** in the navigation pane, click **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 a contact group. For details about the parameters, see **Configuring Alarm Notification Recipients**.	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 **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.
- Some metric templates are created based on the problems reported by CCE Node Problem Detector (**3.14.4 CCE Node Problem Detector**). For details about these metrics, see **Table 3-323**. 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 details page.

Step 2 Choose **Alarm Assistant** in the navigation pane, click **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 a contact group. For details about the parameters, see **Configuring Alarm Notification Recipients**.	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

3.9.5.4 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 Policy**
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 Event Alarms**.
   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 Metric Alarms**.

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.

NOTE

After subscribing to a topic, confirm the subscription in the email or SMS message for the notification to take effect.

Creating an Action Policy

AOM allows you to customize alarm action policies. You can create an alarm action policy to associate an SMN topic and a message template. You can also customize notification content by using a message template.

For details, see **Creating Alarm Action Policies**. When creating an action policy, select the topic that is created and subscribed to in **Creating a Topic on SMN**.

Adding Event Alarms

The following uses the **NodeNotReady** alarm as an example to describe how to add an event alarm.

This function is provided by AOM. For details about parameters, see **Creating an Event Alarm Rule**.

Table 3-324 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.

Event Name	Source	Description	Solution
NodePool SoldOut	CCE	An alarm is triggered immediately when node pool resources are sold out.	Set auto node pool switchover or change the node pool specifications.
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 console.

Step 2 In the navigation pane, choose **Alarm Center** > **Alarm Rules** and click **Add Alarm**.

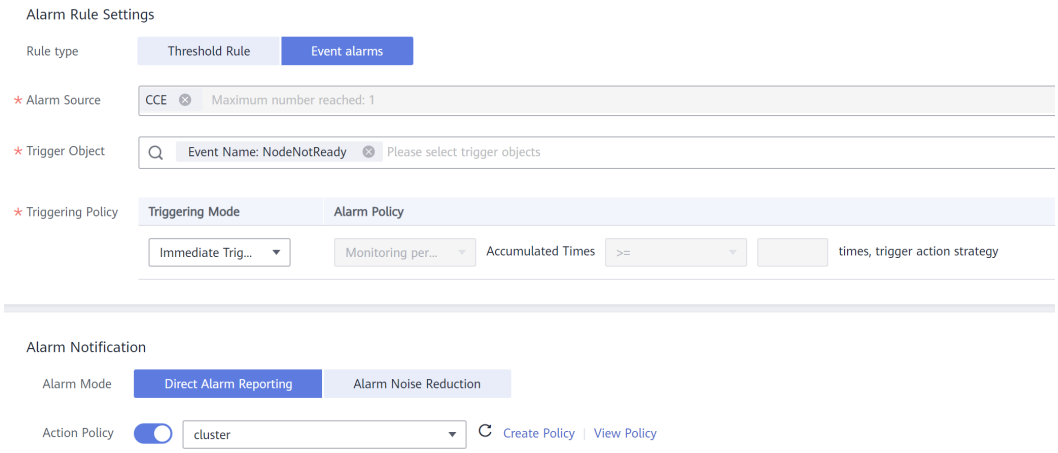
Step 3 Configure an alarm rule.

- **Rule Type**: Select **Event alarm**.
- **Alarm Source**: Select **CCE**.
- **Select Object**: Select **Event Name** and then **NodeNotReady**. You can filter trigger objects by notification type, event name, alarm severity, custom attribute, namespace, and cluster name.
- **Triggering Policy**: Select **Immediate Triggering**.
- **Alarm Mode**: Select **Direct Alarm Reporting**.
- **Action Policy**: Select the action policy created in **Creating an Action Policy**.

This alarm rule works as follows:

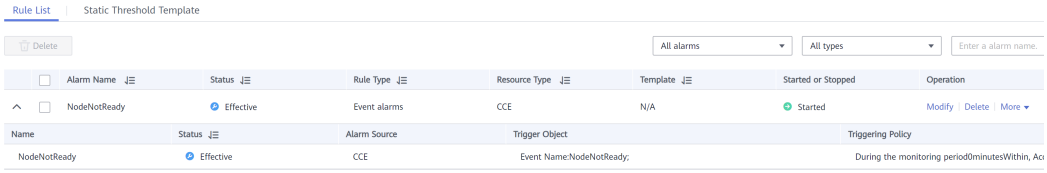
If a node in the cluster becomes abnormal, CCE reports the **NodeNotReady** event to AOM. AOM immediately notifies you through SMN based on the action policy.

Figure 3-251 Creating an event alarm



Step 4 Click **Create Now**.

If the following information is displayed in the rule list, the rule is created successfully.

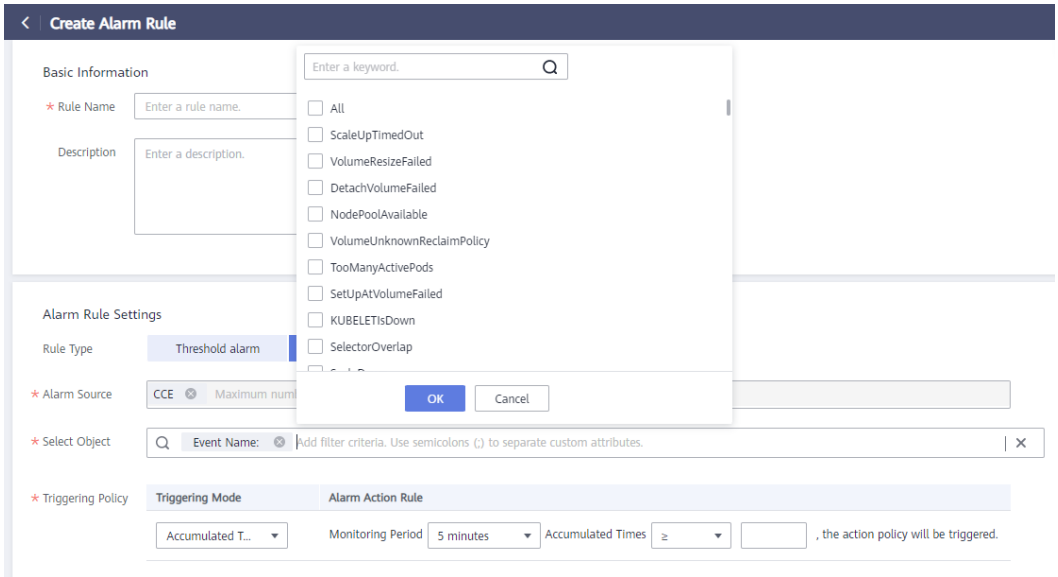


----End

CCE Events

Event alarms are generated based on the events reported by CCE to AOM. CCE reports a series of events to AOM. You can view specific events in the **Alarm Rule Settings** areas and add event alarms as required.

Figure 3-252 Events reported by CCE



Data plane events and **control plane events** can be reported for CCE clusters.

Table 3-325 Data plane events

Type	Event Name	Severity	Remarks
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 Logging (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 a pod is created.
Pod	Unhealthy	Minor	Check whether the pod is running normally.
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.

Type	Event Name	Severity	Remarks
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.
Service	CreatingLoadBalancerFailed	Minor	Check whether a load balancer is created.
Service	DeletingLoadBalancerFailed	Minor	Check whether the load balancer is deleted.
Service	UpdateLoadBalancerFailed	Minor	Check whether the load balancer is updated.
Namespace	DeleteNodeWithNoServer	Minor	Check whether discarded nodes are cleared.
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.
PV	VolumeFailedRecycle	Minor	Check whether the data volume is reclaimed.

Type	Event Name	Severity	Remarks
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 unmounted.
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 working properly.
PVC	ProvisioningFailed	Minor	Check whether the data volume is created.
PVC	ProvisioningCleanupFailed	Minor	Check whether the data volume has been cleared.
PVC	ClaimMisbound	Minor	Check whether the PVC is bound to an incorrect volume.
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 the kubelet status 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.

Type	Event Name	Severity	Remarks
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 node conntrack table 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.
Node	OOMKilling	Minor	Check whether OOM occurs on the node.
Node	DOCKERIsDown	Minor	Check whether the container engine of the node is working properly.
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.
Autoscaler	ScaleUpTimedOut	Major	Check whether adding nodes to the node pool timed out.
Autoscaler	NodePoolAvailable	Major	Check whether the node pool resources are sufficient.

Type	Event Name	Severity	Remarks
Autoscaler	ScaleDown	Major	Nodes are being deleted from the cluster.
Autoscaler	NotTriggerScale-Up	Major	Check whether a node scale-out is triggered.
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.

Table 3-326 Control plane events

Event ID	Severity	Description
Internal error	Major	Check whether an internal error occurs in the cluster.

Event ID	Severity	Description
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.
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 unbound 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 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.

Event ID	Severity	Description
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.
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 properly 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.

Event ID	Severity	Description
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.
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.

Event ID	Severity	Description
Failed to sync the cluster cert	Major	Check whether the cluster certificate is synchronized.

Adding Metric Alarms

The following uses promql: 'kube_persistentvolume_status_phase{phase=~"Failed|Pending"} > 0' as an example to describe how to add metric alarms.

This function is provided by AOM. For details, see **Creating a Metric Alarm Rule**.

NOTICE

The pod CPU usage, physical memory usage, and file system usage alarms must be configured for the everest-csi-controller, everest-csi-driver, coredns, autoscaler, and Yangtse components. Upgrade the specifications in the case of high resource usage to prevent system failures.

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:

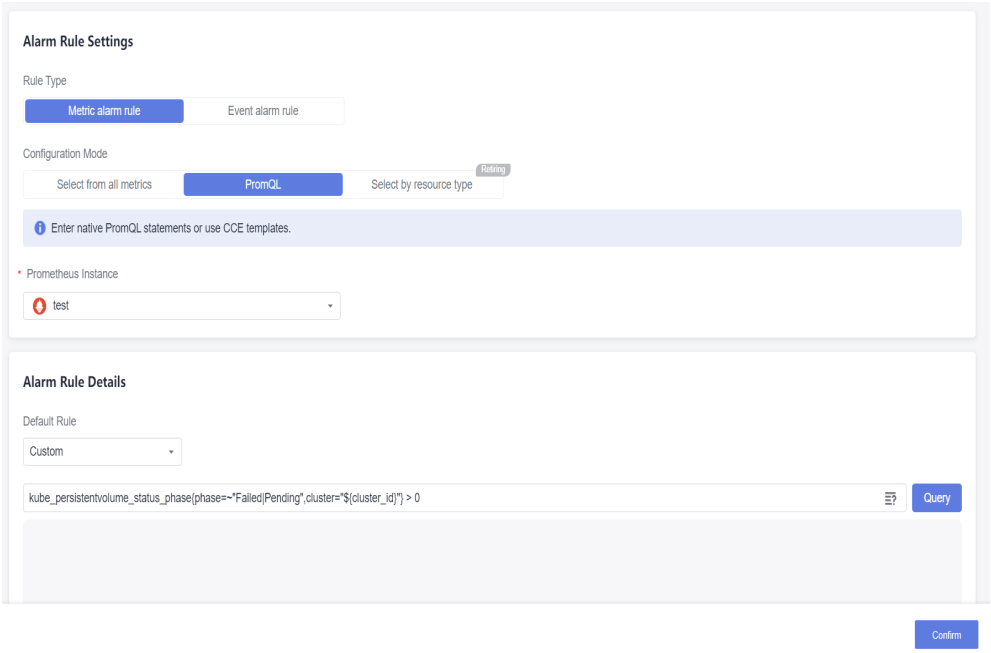
- **Rule Type**: Select **Metric alarm rule**.
- **Configuration Mode**: Select **PromQL**. You are advised to specify the cluster for which alarms are generated. Example:

  kube_persistentvolume_status_phase{phase=~"Failed|Pending",cluster="${cluster_id}"} > 0
- **Prometheus Instance**: Select the AOM instance interconnected with the cloud native cluster monitoring add-on.
- **Alarm Mode**: Select **Direct alarm reporting**.
- **Action Rule**: Select the action rule created for the cluster when Alarm Assistant is enabled. The rule name can be **auto-cluster-$*{cluster_id}***.

This alarm rule works as follows:

When a PromQL rule is triggered, AOM immediately notifies you through SMN based on the action policy.

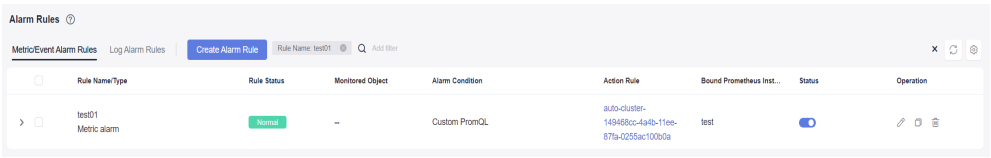
Figure 3-253 Custom metric alarms



Step 4 Click **Confirm**.

If the following information is displayed in the rule list, the rule is created successfully.

Figure 3-254 Alarm rule list



----End

3.9.6 Best Practices

3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring

CCE provides a 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 the Cloud Native Cluster Monitoring Add-on**

   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.

   You can customize monitoring metrics by these ways:

   - **Method 1: Configuring Custom Metrics for Pod Annotations**
   - **Method 2: Configuring Custom Metrics for Service Annotations**
   - **Method 3: Configuring Custom Metrics for PodMonitor**
   - **Method 4: Configuring Custom Metrics for ServiceMonitor**

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.

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:

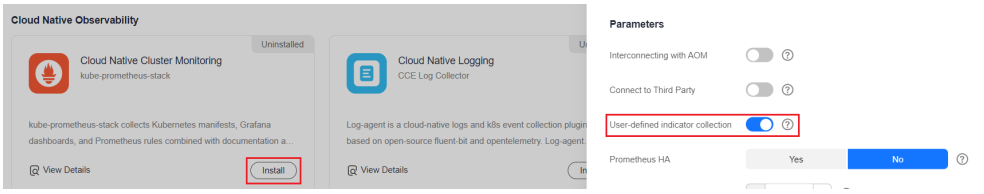
```
# TYPE nginx_connections_active gauge
nginx_connections_active 2
# TYPE nginx_connections_reading gauge
nginx_connections_reading 0
```

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**.

Installing the Cloud Native Cluster Monitoring Add-on

3.14.17 Cloud Native Cluster Monitoring is available only in clusters v1.17 or later. In addition to the monitoring capabilities, this add-on interconnects monitoring data with Container Intelligent Analysis.

- For 3.8.0 and later versions, ensure that custom metric collection is enabled.



- For versions earlier than 3.8.0, you do not need to enable custom metric collection.

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 build 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**. On the displayed dialog box, click **Generate a temporary login command** and 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 Custom Metrics for 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"
```



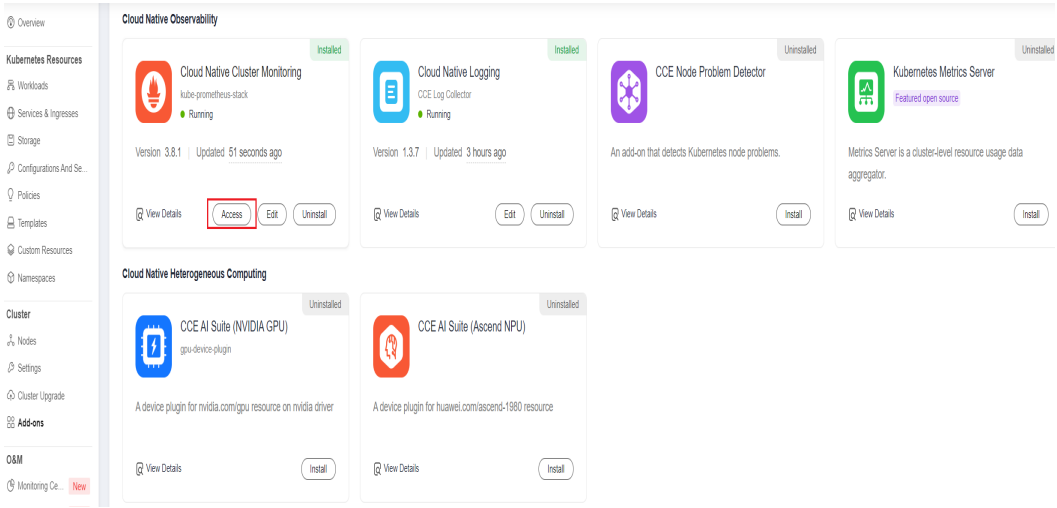
```
    prometheus.io/path: "/metrics"
    prometheus.io/scheme: "http"
spec:
  containers:
    - name: container-0
      image: 'nginx:exporter'      # Replace it with the address of the image you uploaded to SWR.
      resources:
        limits:
          cpu: 250m
          memory: 512Mi
        requests:
          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 the cloud native cluster monitoring add-on to query custom monitoring metrics.

Figure 3-255 Accessing the cloud native cluster monitoring add-on



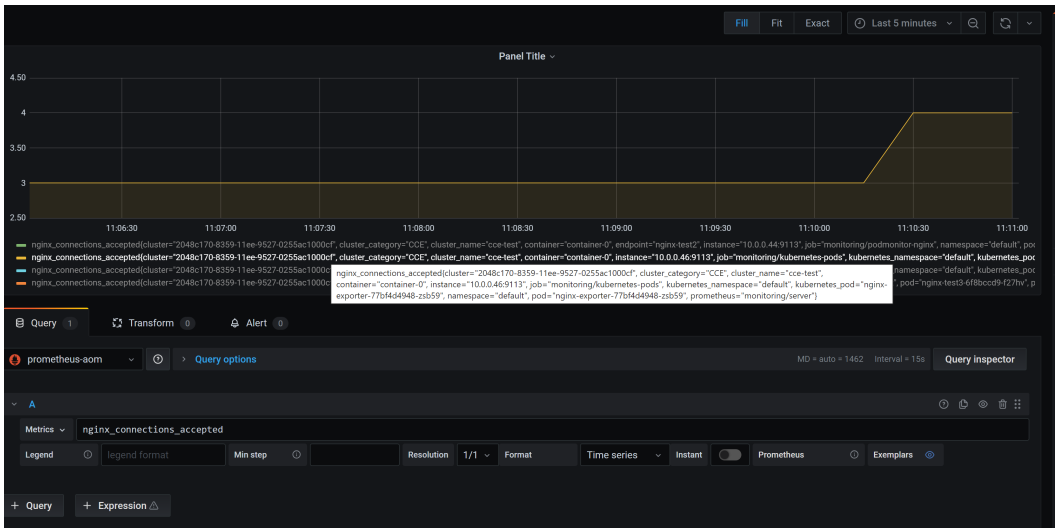
The custom monitoring metrics related to Nginx can be queried. You can use the job name to determine whether the metrics are reported based on the pod settings.

```
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"}
```

Figure 3-256 Viewing monitoring metrics



Method 2: Configuring Custom Metrics for 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
```

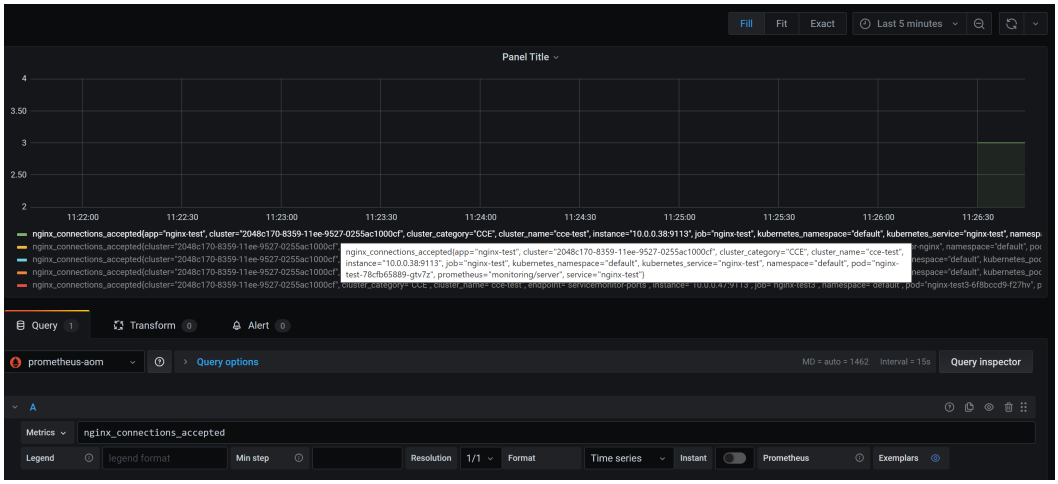
The following is an example service configuration:

```
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
```

View the metric. You can use the service name to determine whether the metric is 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"}
```

Figure 3-257 Viewing monitoring metrics



Method 3: Configuring Custom Metrics for PodMonitor

The cloud native cluster monitoring add-on 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:
```

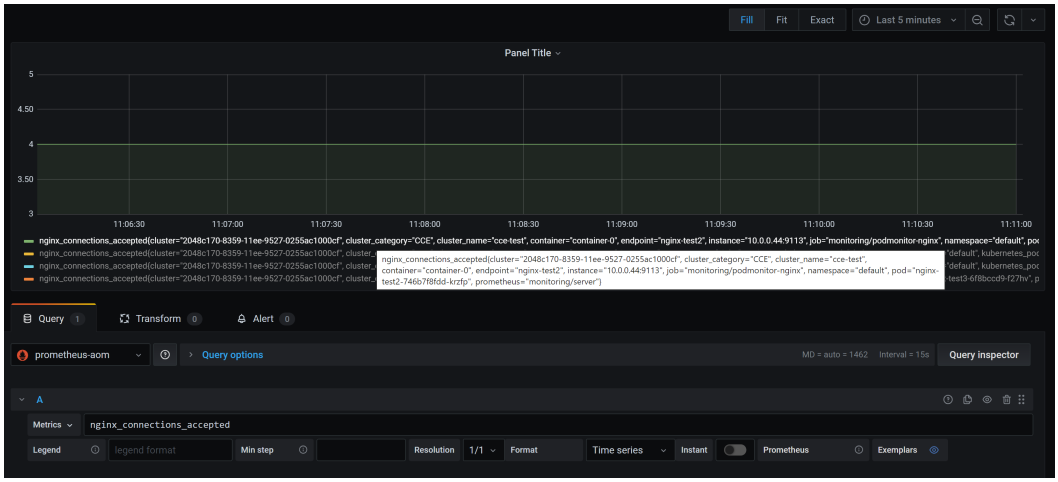


```
    insecureSkipVerify: true
selector:
  matchLabels:
    app: nginx-test2   # Label carried by the pod, which can be selected by the selector
```

View the metric. You can use the job name to determine whether the metric is reported based on the PodMonitor settings.

```
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"}
```

Figure 3-258 Viewing monitoring metrics



Method 4: Configuring Custom Metrics for ServiceMonitor

The cloud native cluster monitoring add-on 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:

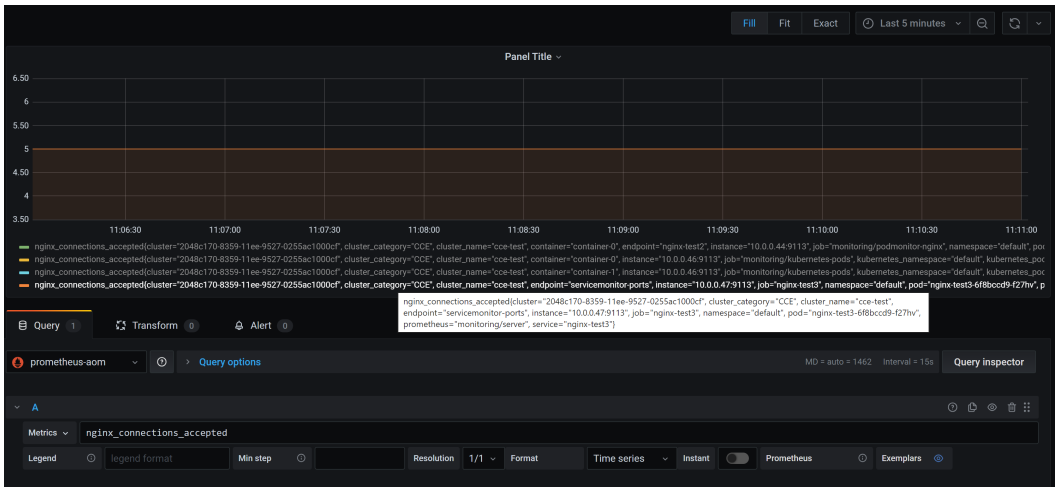
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-test3
  labels:
    app: nginx-test3
  namespace: default
spec:
  selector:
    app: nginx-test3
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 8080
      protocol: TCP
    - 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
```

View the metric. You can use the endpoint name to determine whether the metric is reported based on the ServiceMonitor settings.

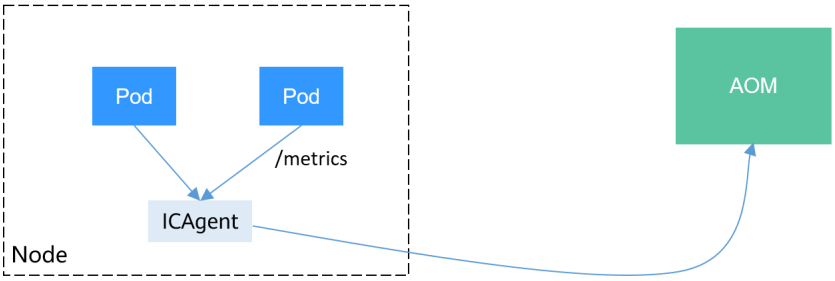
```
nginx_connections_accepted{cluster="2048c170-8359-11ee-9527-0255ac1000cf", cluster_category="CCE",
cluster_name="cce-test", endpoint="servicemonitor-ports", instance="10.0.0.47:9113", job="nginx-test3",
namespace="default", pod="nginx-test3-6f8bccd9-f27hv", prometheus="monitoring/server", service="nginx-
test3"}
```

3.9.6.2 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 3-259 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 build 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**. On the displayed dialog box, click **Generate a temporary login command** and 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.

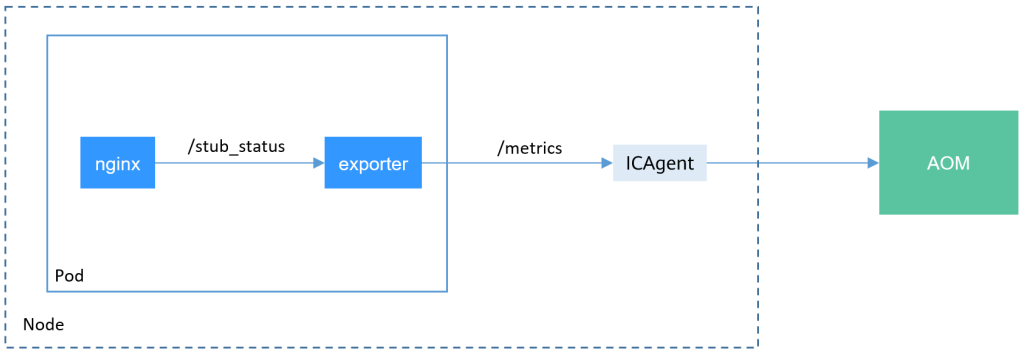
   ```
   # curl http://127.0.0.1:8080/stub_status
   Active connections: 3
   server accepts handled requests
    146269 146269 212
   Reading: 0 Writing: 1 Waiting: 2
   ```

----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 3-260 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.

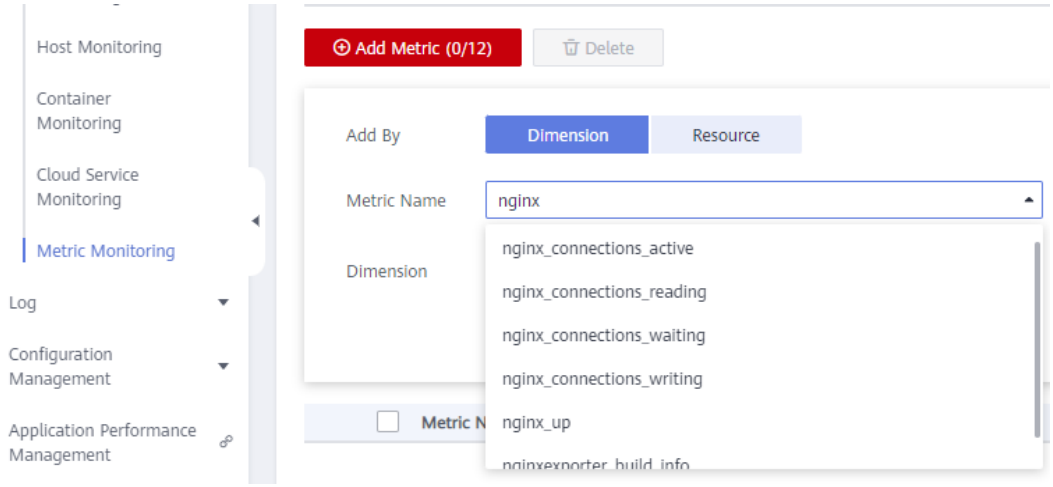
```
$ kubectl get pod
NAME                              READY   STATUS    RESTARTS   AGE
nginx-exporter-78859765db-6j8sw   2/2     Running   0          4m
```

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 3-261 Viewing monitoring metrics



----End

3.9.6.3 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 (**3.14.17 Cloud Native Cluster Monitoring** 3.5.0 or later installed) in the cluster, you can view the API metrics in the API server view (**3.9.3.8.3 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.

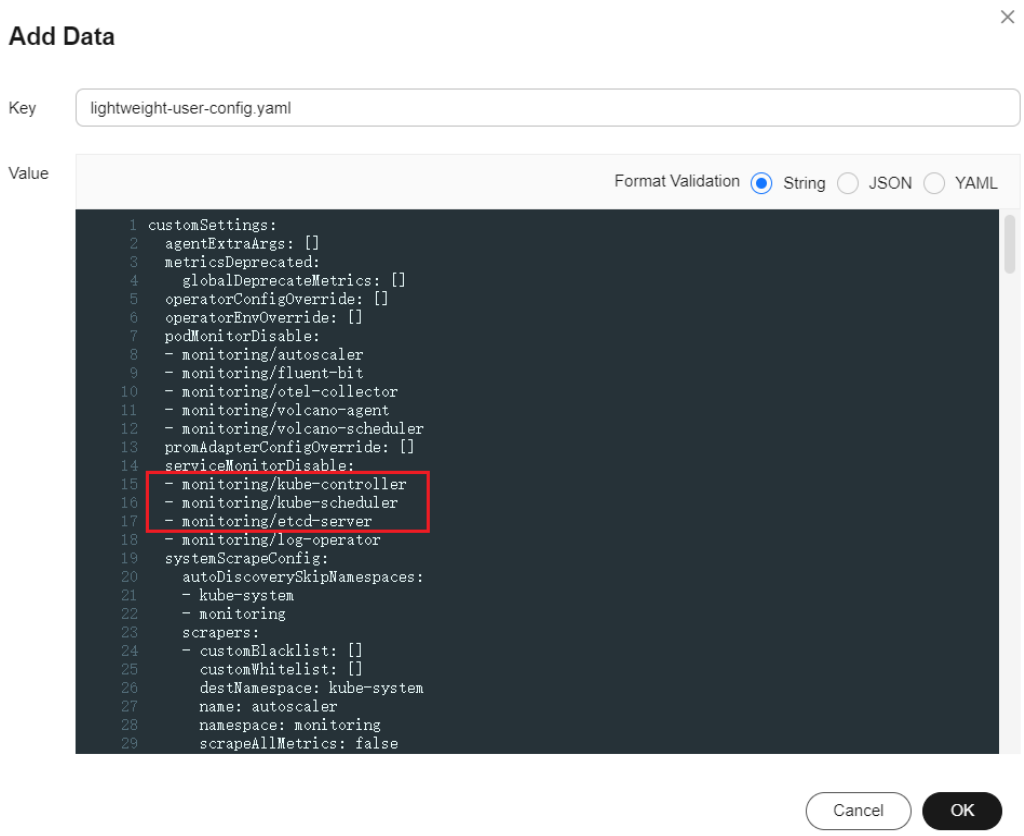
Step 1 Log in to the CCE console and click the cluster name to access the details page.

Step 2 In the navigation tree, 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 3-262 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 3-263 Viewing metrics

----End

Collecting Metrics of Master Node Components Using Prometheus

This section describes how to collect metrics of master node components using Prometheus.

NOTICE

- The cluster version must be 1.19 or later.
- You need to install Prometheus using Helm by referring to **Prometheus**. You need to use prometheus-operator to manage installed Prometheus. For details, see **Prometheus Operator**.

  The Prometheus (**3.14.23 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:
```



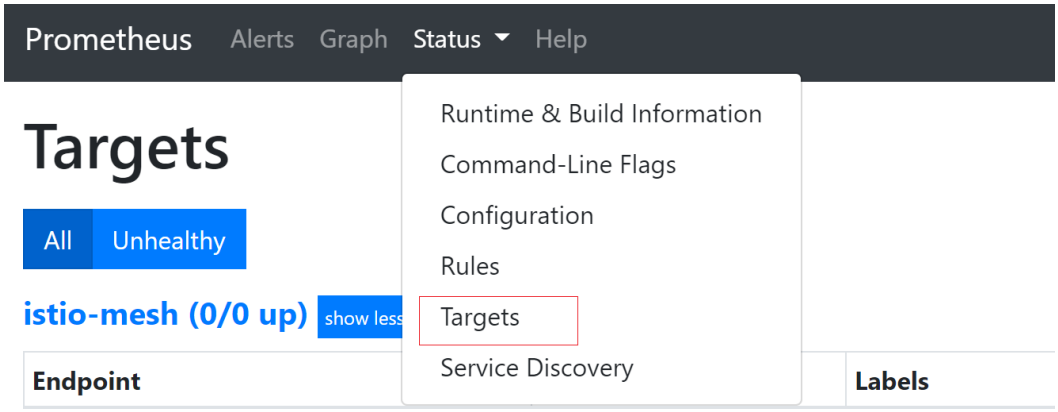
```
        - __address__
        targetLabel: instance
      - replacement: kubernetes.default.svc.cluster.local:443
        targetLabel: __address__
    scheme: https
    tlsConfig:
      caFile: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
jobLabel: app
namespaceSelector:
  matchNames:
    - 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

3.9.6.4 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 **3.14.17 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 **3.14.7 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.

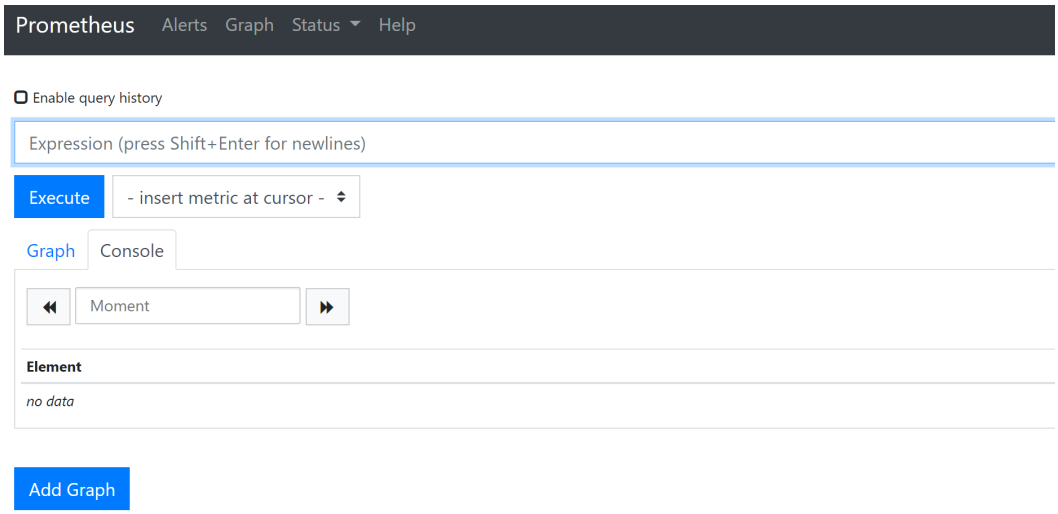
Step 1 Log in to the CCE console and click the name of the cluster with Prometheus installed to access the details page. In the navigation pane, choose **Services & Ingresses**.

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
```

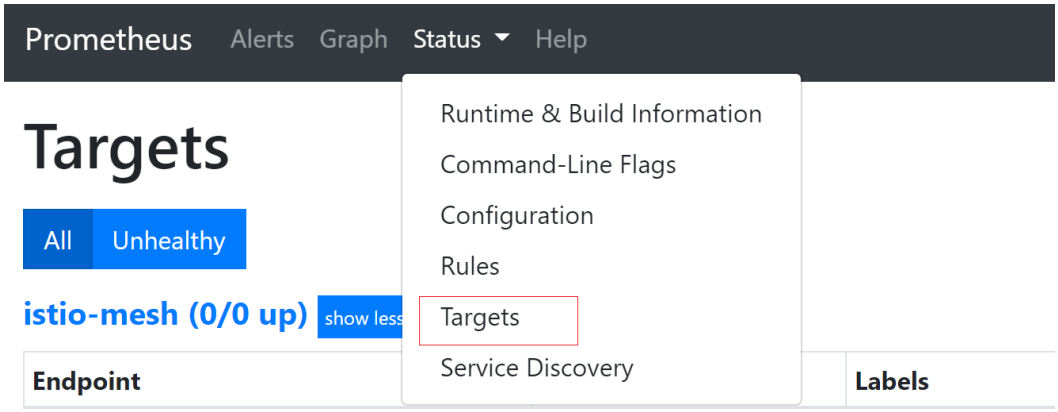
Step 3 After the Service is created, enter *Public IP address of the load balancer*:*Service port* in the address box of the browser to access Prometheus.

Figure 3-264 Accessing Prometheus



Step 4 Choose **Status** > **Targets** to view the targets monitored by Prometheus.

Figure 3-265 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 3-266 Viewing the metrics of NGINX Ingress Controller

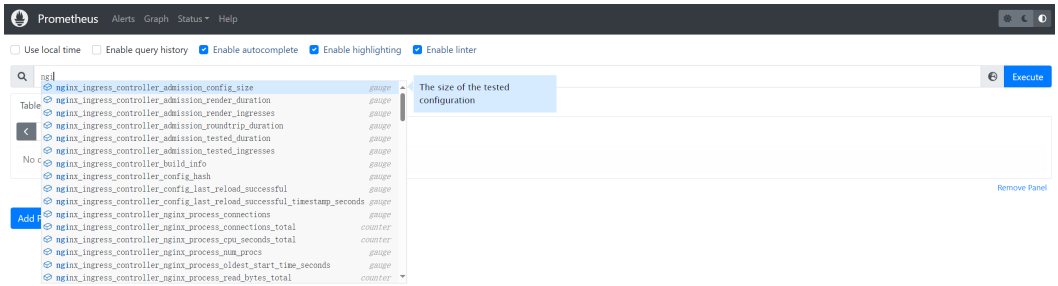


Table 3-327 Metrics of NGINX Ingress Controller

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

3.9.6.5 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 (**3.14.17 Cloud Native Cluster Monitoring**) and CCE Network Metrics Exporter (**3.14.15 CCE Network Metrics Exporter**) add-ons.
- You can access the cluster using kubectl. For details, see **3.2.3.1 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 details page. 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 **3.14.17 Cloud Native Cluster Monitoring**.

- **Deployment Mode**: Select **Server mode**. The monitoring data will be stored locally. You can determine whether to connect this add-on to AOM or a third-party monitoring platform. If the **Agent mode** is used, AOM or a third-party monitoring system must be interconnected. You can select a monitoring platform as required.
- **User-defined indicator 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 the cloud native cluster monitoring add-on of v3.9.0 or later, Grafana is not provided by default.) Locate the independent Grafana add-on and click **Install**.

If you enable node access, a NodePort Service named **grafana-oss** will be created in the **monitoring** namespace. If an EIP is bound to the node, you can enter *Public IP address of the node*:*Service port* in the address box of a browser to access the node.

NOTICE

Node access exposes the open-source Grafana service to the public network. You are advised to evaluate security risks and control access 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, for example, labelSelector, monitored by the CCE Network Metrics Exporter add-on. By default, all containers 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
```



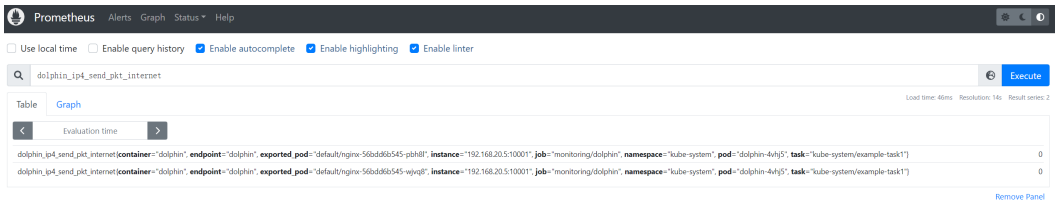
```
  healthCheck:                # (Optional) Whether to collect statistics about the latest health check results and the total numbers of health checks in which pods are considered healthy and of health checks in which pods are considered unhealthy. This option is disabled by default.
    enable: true              # 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 3-267 Accessing Prometheus



----End

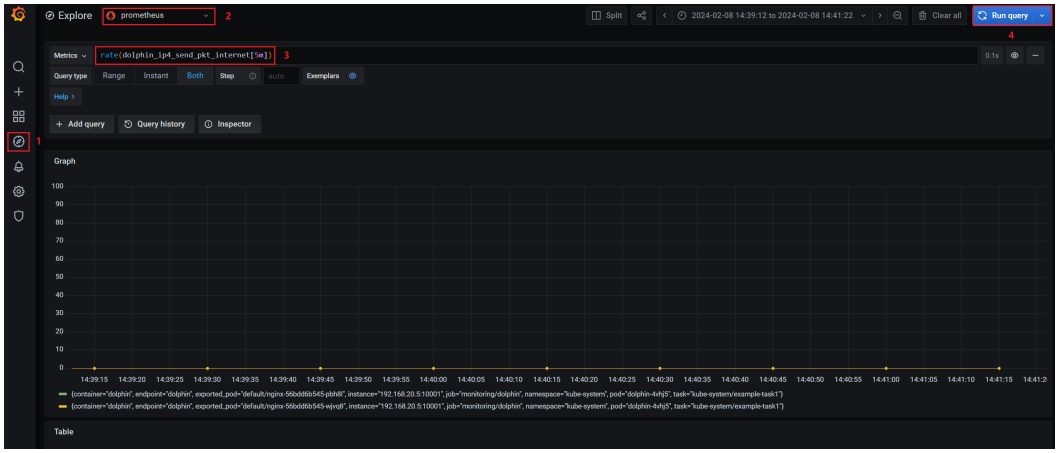
(Optional) Viewing Graphs on Grafana

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 3-268 Grafana graph



Step 4 (Optional) Use common graphs as Grafana dashboards. For details, see **Create a dashboard**.

----End

3.9.7 Cloud Trace Service

3.9.7.1 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 3-328 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
Updating a node pool	Node pool	updateNodePool
Deleting a node pool	Node pool	claimNodePool
Migrating a node pool	Node pool	migrateNodepool

Operation	Resource Type	Event Name
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
Creating an Ingress	Kubernetes resource	createIngresses
Creating a job	Kubernetes resource	createJobs
Creating a namespace	Kubernetes resource	createNamespaces

Operation	Resource Type	Event Name
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
Deleting a namespace	Kubernetes resource	deleteNamespaces
Deleting a node	Kubernetes resource	deleteNodes
Deleting a Pod	Kubernetes resource	deletePods

Operation	Resource Type	Event Name
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
Replacing a specified pod	Kubernetes resource	updatePods
Replacing a specified replica set	Kubernetes resource	updateReplicasets
Replacing a specified resource quota	Kubernetes resource	updateResourcequotas

Operation	Resource Type	Event Name
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

3.9.7.2 Querying Real-Time Traces

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 OBS buckets. CTS stores operation records generated in the last seven days.

This section describes how to query and 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 more than seven days, you must configure an OBS bucket to transfer records to it. Otherwise, you cannot query the operation records generated seven days ago.
- After performing operations on the cloud, you can query management traces on the CTS console 1 minute later and query data traces on the CTS console 5 minutes later.

Viewing Real-Time Traces in the Trace List of the New Edition

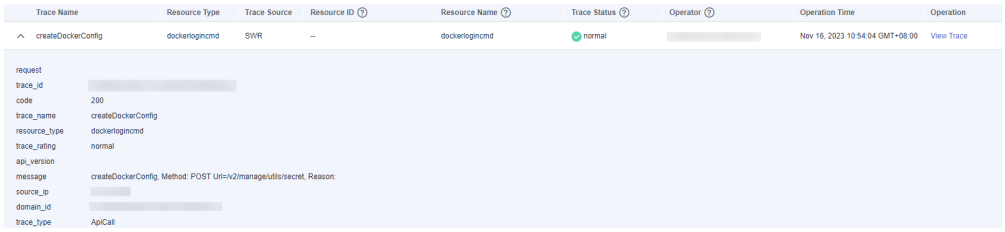
1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & Governance** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. On the **Trace List** page, use advanced search to query traces. You 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.
   - Time range: Select **Last 1 hour**, **Last 1 day**, or **Last 1 week**, or specify a custom time range.
5. On the **Trace List** page, you can also export and refresh the trace list, and customize the list display settings.
   - 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 5000 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.

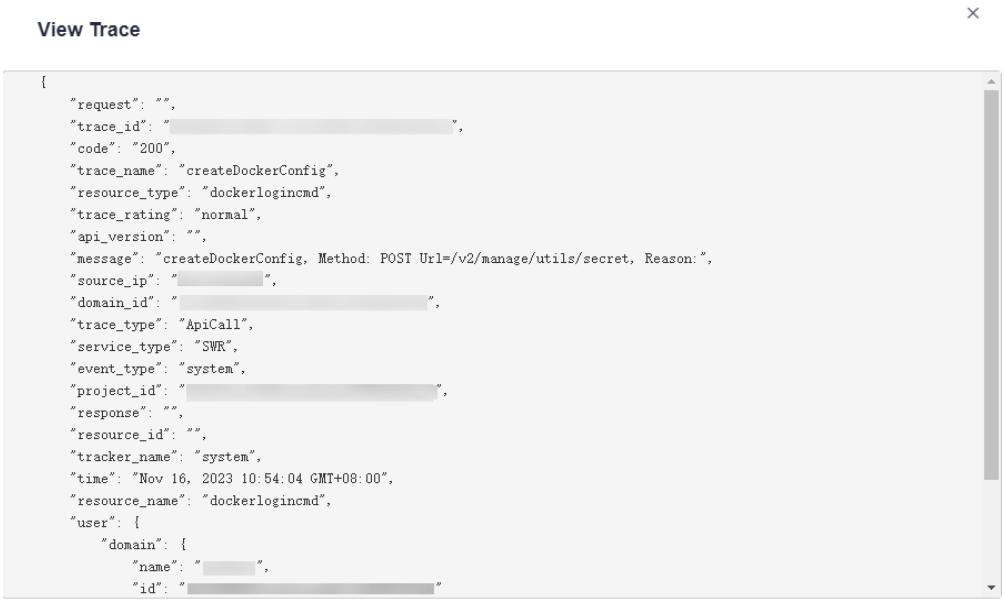
6. For details about key fields in the trace structure, see **Trace Structure** and **Example Traces**.
7. (Optional) On the **Trace List** page of the new edition, click **Go to Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

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: You can query traces generated during any time range in the last seven days.
   - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
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 5000 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.

3.10 Namespaces

3.10.1 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.

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 created in either of the following ways:

- 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).
  - **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.

Creating 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 and click **Create Namespace** in the upper right corner.

Step 3 Set namespace parameters based on **Table 3-329**.

Table 3-329 Parameters for creating a namespace

Parameter	Description
Name	Unique name of the created namespace.
Description	Description about the namespace.

Parameter	Description
Quota Management	Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace. **NOTICE** **You are advised to set resource quotas in the namespace as required to prevent cluster or node exceptions caused by resource overload.** For example, the default number of pods that can be created on each node in a cluster is 110. If you create a cluster with 50 nodes, you can create a maximum of 5,500 pods. Therefore, you can set a resource quota to ensure that the total number of pods in all namespaces does not exceed 5,500. Enter an integer. If the quota of a resource is not specified, no limit is posed 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

Using kubectl to Create a Namespace

Define a namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

Run the **kubectl** command to create it.

```
$ kubectl create -f custom-namespace.yaml
namespace/custom-namespace created
```

You can also run the **kubectl create namespace** command to create a namespace.

```
$ kubectl create namespace custom-namespace
namespace/custom-namespace created
```

3.10.2 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.

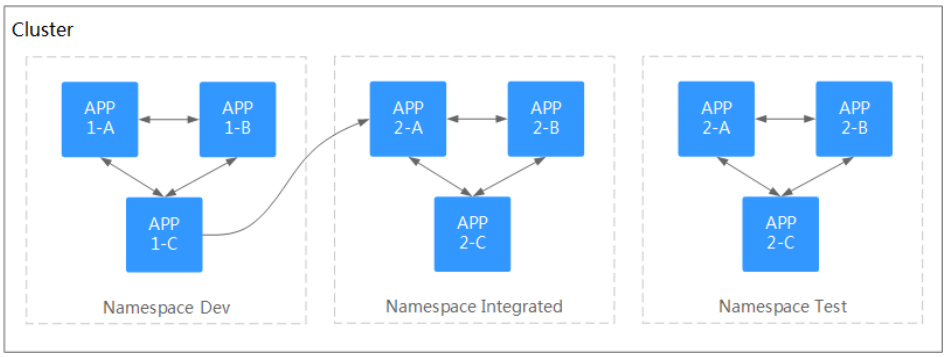
    Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

  – 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.

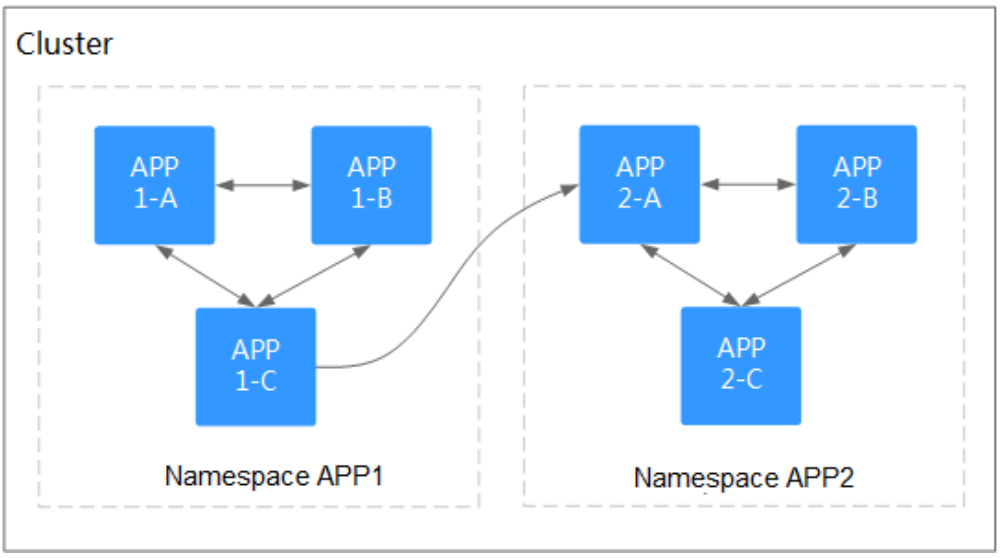
    Figure 3-269 One namespace for one environment

- **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 3-270 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.

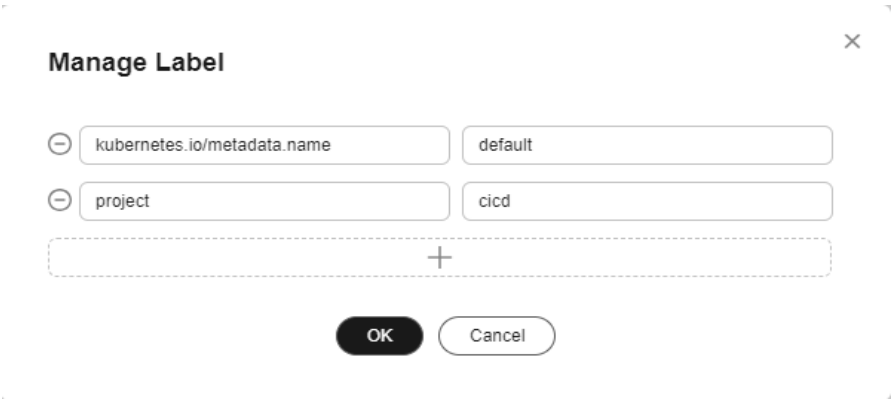
Step 3 In the dialog box that is displayed, the existing labels of the namespace are displayed. Modify the labels as needed.

- 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 ⊖ next the label to be deleted and then **OK**.

Figure 3-271 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 **3.3.7.1 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

3.10.3 Configuring Resource Quotas

Namespace-level resource quotas limit the amount of resources available to teams or users when these teams or users use the same cluster. The quotas include the total number of a type of objects and the total amount of compute resources (CPU and memory) consumed by the objects.

Usage

By default, running pods can use the CPUs and memory of a node without restrictions. This means the pods in a namespace may exhaust all resources of the cluster.

Kubernetes provides namespaces for you to group workloads in a cluster. By setting resource quotas for each namespace, you can prevent resource exhaustion and ensure cluster reliability.

You can configure quotas for resources such as CPU, memory, and the number of pods in a namespace. For more information, see **Resource Quotas**.

The following table recommends how many pods you can configure for your clusters of different sizes.

Cluster Scale	Recommended Number of Pods
50 nodes	2,500 pods
200 nodes	10,000 pods
1000 nodes	30,000 pods
2000 nodes	50,000 pods

In clusters of v1.21 and 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 3-330** lists the resource quotas based on cluster specifications. You can modify them according to your service requirements.

Table 3-330 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

Constraints

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. The resource quota limits the total resource consumption of each namespace and records the resource information in the cluster. Therefore, after the **enable-resource-quota** option is enabled, the probability of resource creation conflicts increases in large-scale concurrency scenarios, affecting the performance of batch resource creation.

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 Set the resource quotas and click **OK**.

NOTICE

- After setting 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.

----End

3.11 ConfigMaps and Secrets

3.11.1 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.

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 3-331 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 connect to the cluster. For details, see **3.2.3.1 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 3-332 Key parameters

Parameter	Description
apiVersion	The value is fixed at **v1**.

Parameter	Description
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 3-333**.

Table 3-333 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 3-331**. 3. Click **OK**.
Deleting a ConfigMap	Select the configuration you want to delete and click **Delete**. Follow the prompts to delete the ConfigMap.

3.11.2 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

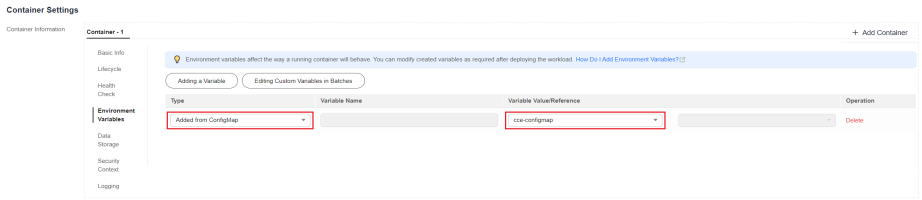
Using the CCE console

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**. In the dialog box displayed, click **Create Workload** in the upper right corner.

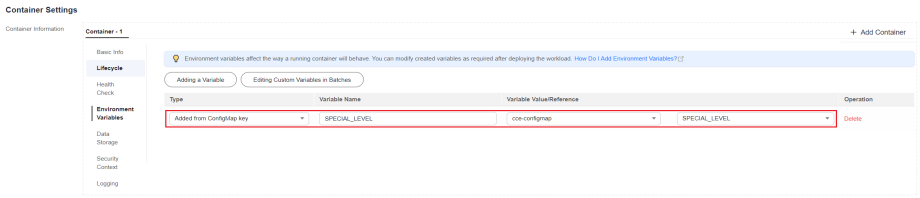
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **Added from 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 connect to the cluster. For details, see **3.2.3.1 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).

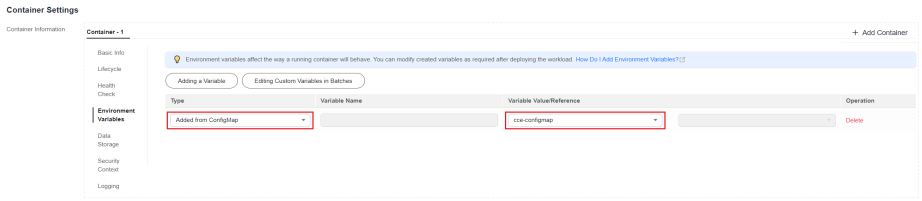
Using the CCE console

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**.

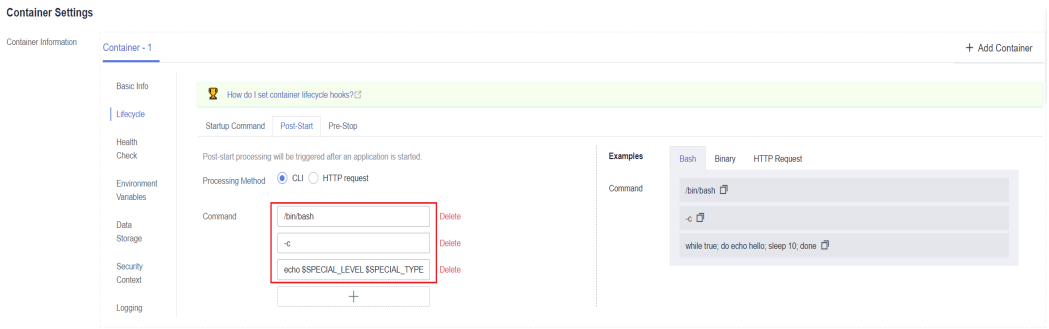
- **Added from ConfigMap**: Select a ConfigMap to import all of its keys as environment variables.



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, that is, the 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 connect to the cluster. For details, see **3.2.3.1 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, the **cce-configmap** ConfigMap is imported to the workload. *SPECIAL_LEVEL* and *SPECIAL_TYPE* are the environment variable names in the workload, that is, the 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 After the workload runs properly, the following content is entered into 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 **3.11.1 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.

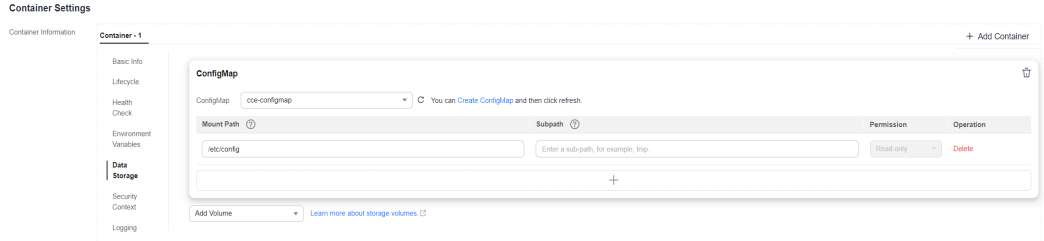
When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **ConfigMap** from the drop-down list.

Step 3 Select parameters for mounting a ConfigMap volume, as shown in **Table 3-334**.

Table 3-334 Mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created beforehand. For details, see **3.11.1 Creating a ConfigMap**.
Mount Path	Enter a mount point. 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, which leads to a container startup failure or workload creation failure. **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 is used by default. • The subpath can be the key and value of a ConfigMap or 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 ConfigMap/secret updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 3-272 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 connect to the cluster. For details, see **3.2.3.1 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

3.11.3 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.

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 3-335 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.

Parameter	Description
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.
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:

```
eyJhdXRocyI6eyJhZGRyZXNzIjp7InVzZXJuYW1lIjoidXNlcm5hbWUiLCJwYXNzd29yZCI6InBhc3N3b3J
kIiwiYXV0aCI6ImRYTmxjbTVoYldVNmNHRnpjM2R2Y21RPSJ9fX0=
```

     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 connect to the cluster. For details, see **3.2.3.1 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 3-336**.

NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

Table 3-336 Related Operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target secret resides to edit its YAML file.

Operation	Description
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 3-335**. 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
******
```

3.11.4 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 **3.11.5 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

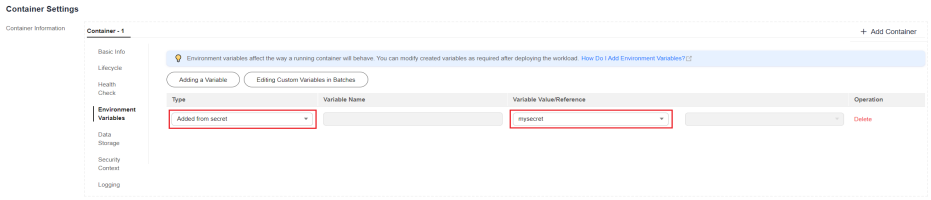
Using the CCE console

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.

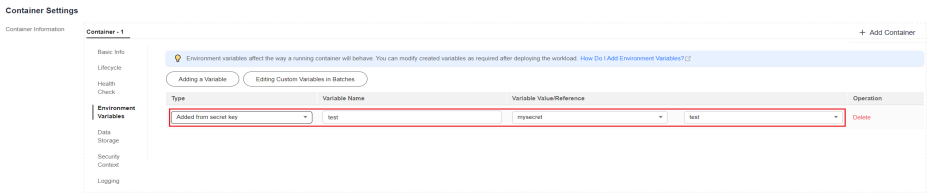
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **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 connect to the cluster. For details, see **3.2.3.1 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 **3.11.3 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.

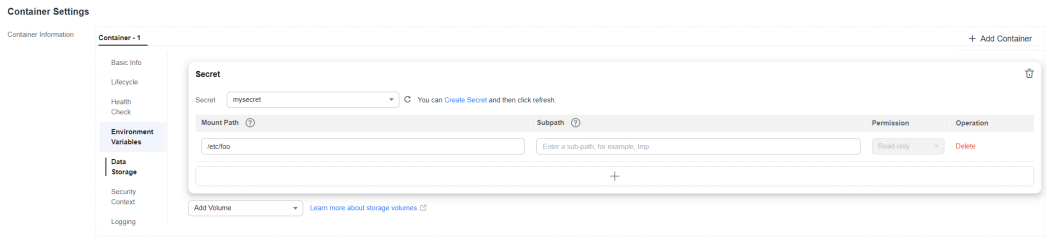
Step 3 Select parameters for mounting a secret volume, as shown in **Table 3-337**.

Table 3-337 Mounting a secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created beforehand. For details, see **3.11.3 Creating a Secret**.

Parameter	Description
Mount Path	Enter a mount point. 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 cause 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, which leads to a container startup failure or workload creation failure. **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 is used by default. • The subpath can be the key and value of a ConfigMap or 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 ConfigMap/secret updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 3-273 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 connect to the cluster. For details, see **3.2.3.1 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 are 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

3.11.5 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 of 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 data of **paas.elb** is the temporary AK/SK data, which is used to create ELB load balancers during Service and ingress creation. The data of paas.elb is periodically updated and expires after a certain period of time.

In practice, you will not directly use paas.elb. However, do not delete it. Otherwise, ELB load balancers will fail to be created.

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.

```
$ 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>
```


3.12 Auto Scaling

3.12.1 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 are developed based 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.

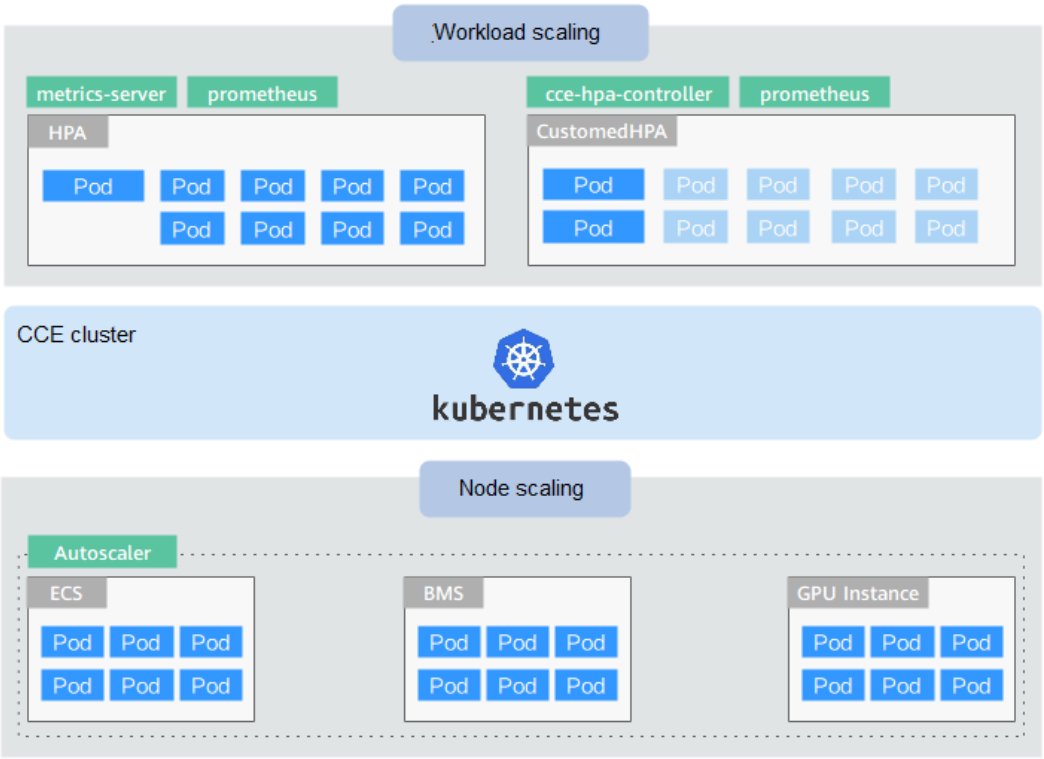
Auto Scaling in CCE

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 **3.12.4 Using HPA and CA for Auto Scaling of Workloads and Nodes**.

Components



Workload scaling components are described as follows:

Table 3-338 Workload scaling components

Type	Component Name	Component Description	Reference
HPA	3.14.8 Kubernetes Metrics Server	A built-in component of Kubernetes, which enables horizontal scaling of pods. It adds the application-level cooldown time window and scaling threshold functions based on the HPA.	3.12.2.2 HPA Policies
Customed HPA	3.14.9 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).	3.12.2.4 CustomedHPA Policies

Type	Component Name	Component Description	Reference
	3.14.23 Prometheus (EOM) 3.14.17 Cloud Native Cluster Monitoring	An open-source system monitoring and alarm framework, which collects public metrics (CPU usage and memory usage) of kubelet in the Kubernetes cluster.	
CronHPA	3.14.9 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.	3.12.2.3 CronHPA Policies

Node scaling components are described as follows:

Table 3-339 Node scaling components

Component Name	Component Description	Application Scenario	Reference
3.14.6 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
3.14.10 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	3.12.5 Elastic Scaling of CCE Pods to CCI

3.12.2 Scaling a Workload

3.12.2.1 Workload Scaling Rules

CCE supports multiple workload scaling modes. Comparisons between the scaling policies are listed in the following table.

Table 3-340 Comparisons between auto scaling policies

Item	HPA	CronHPA	CustomedHPA
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.
Rules	Scales Deployments based on **metrics** (CPU usage and memory usage).	Scales Deployments **periodically** (daily, weekly, monthly, or yearly at a specific time).	Scales 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).
Enhancement	Adds the application-level cooldown time window and scaling threshold functions based on the Kubernetes HPA.	Compatible with HPA objects, which allows 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.
Usage	**3.12.2.2 HPA Policies**	**3.12.2.3 CronHPA Policies**	**3.12.2.4 CustomedHPA Policies**

How HPA Works

HPA is a controller that controls horizontal pod scaling. HPA periodically checks the pod metrics, calculates the number of replicas 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).

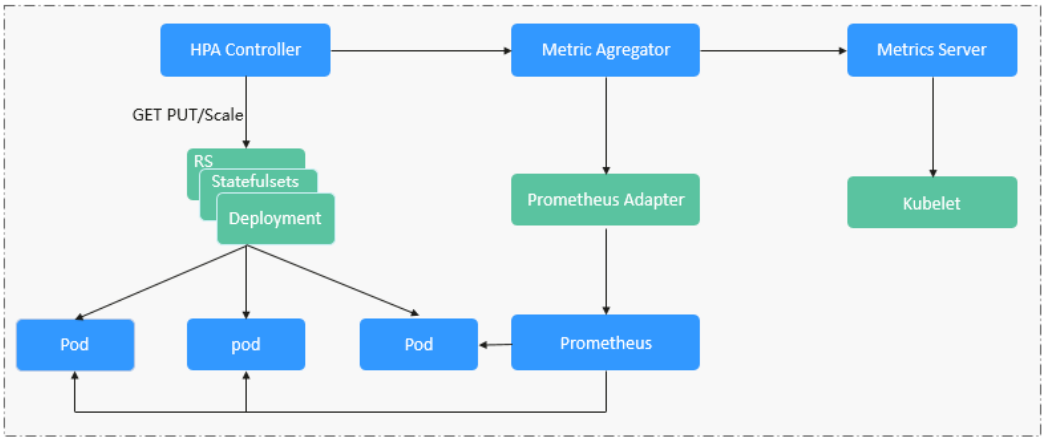
A prerequisite for auto scaling is that your container running data can be collected, such as number of cluster nodes/pods, and CPU and memory usage of containers. Kubernetes does not provide such monitoring capabilities itself. You can use extensions to monitor and collect your data. CCE integrates **Prometheus** and **Metrics Server** to realize such capabilities:

- **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 to implement auto scaling based on the CPU and memory usage. It can also work with Prometheus for auto scaling based on custom monitoring metrics.

Figure 3-274 shows how HPA works.

Figure 3-274 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:

  desiredReplicas = ceil[currentReplicas x (currentMetricValue/desiredMetricValue)]

  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 replicas 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.

    Use the formula: ratio = currentMetricValue/desiredMetricValue

    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.

3.12.2.2 HPA Policies

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.

- **3.14.8 Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.

- **3.14.17 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 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**.
- **3.14.23 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.

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.

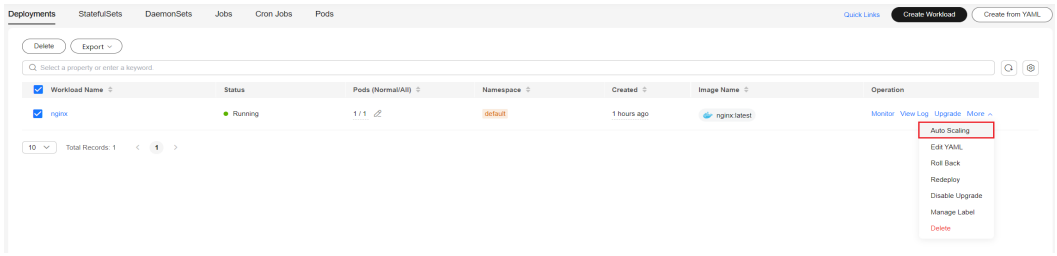
  For clusters of v1.19.10 and later, if an HPA policy is used to scale out a workload with EVS volume mounted, a new pod cannot be started because EVS disks cannot be attached.

Creating an HPA Policy

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 3-275 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 **3.12.2.3 CronHPA Policies**.

Figure 3-276 Enabling the HPA policy

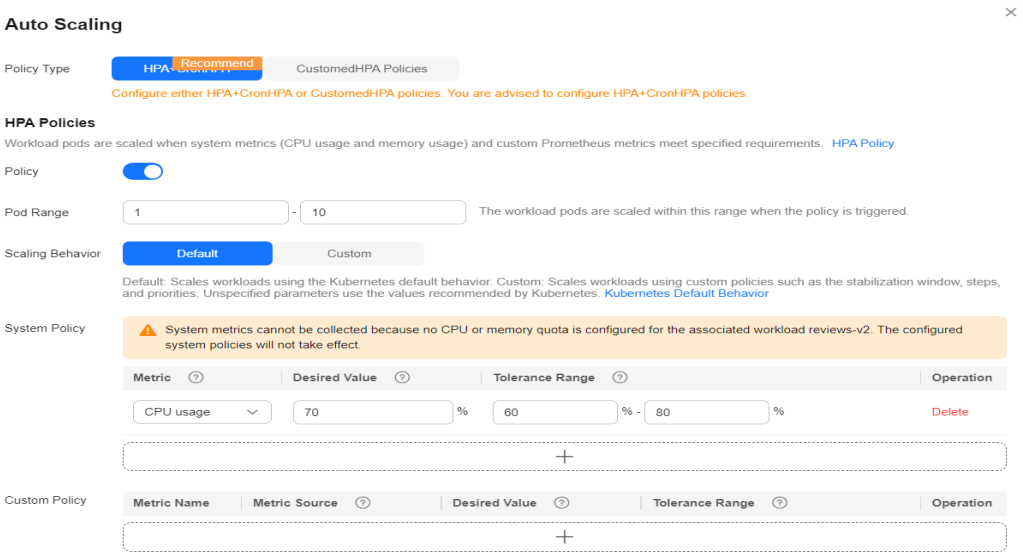


Table 3-341 HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
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 = CPUs or memory used by pods/Requested CPUs or memory. • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. 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. 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.**

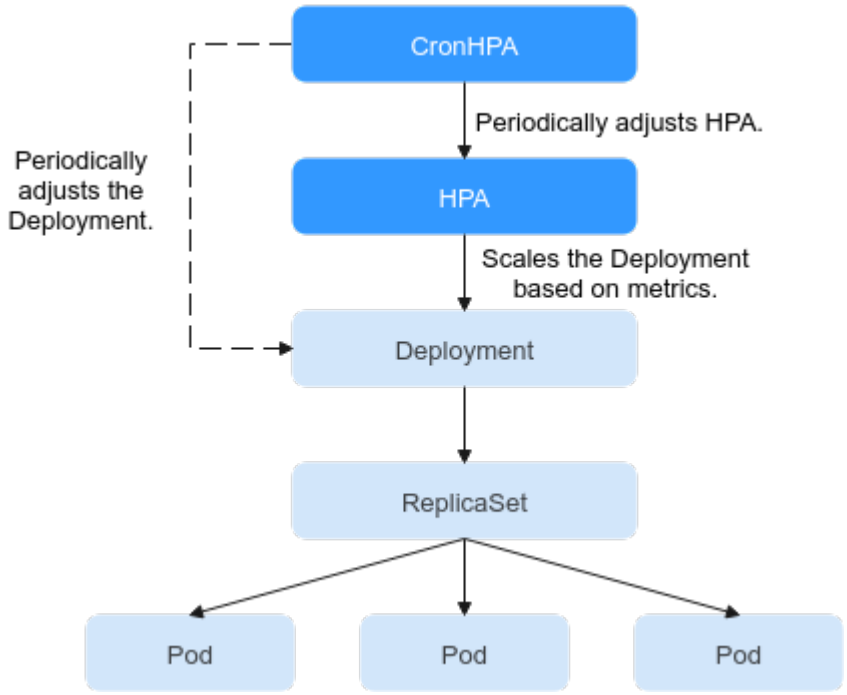
Parameter	Description
Custom Policy (supported only in clusters of v1.15 or later)	**NOTE** Before creating a custom policy, install an add-on that supports custom metric collection (for example, Prometheus) in the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see **3.9.6.1 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

3.12.2.3 CronHPA Policies

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.

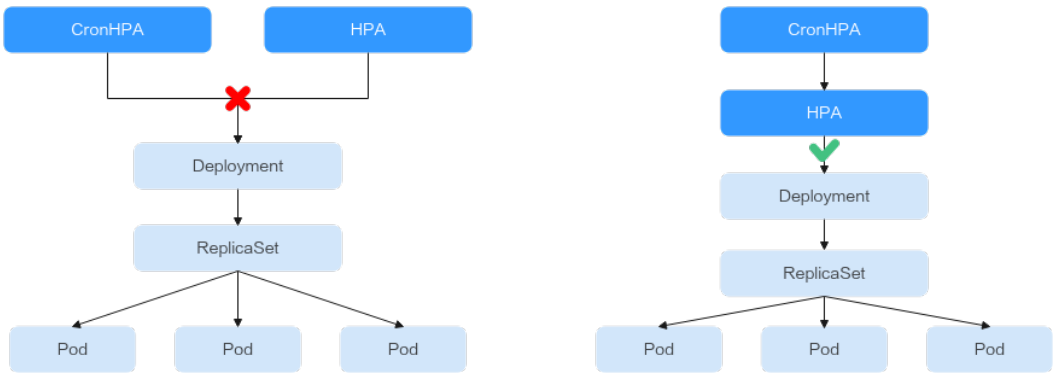
Prerequisites

The add-on **3.14.9 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 3-277 CronHPA scaling scenarios

Figure 3-277 shows possible scaling scenarios. The following examples detail how CronHPA modifies the number of pods in HPAs.

Table 3-342 CronHPA scaling parameters

Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
1	**targetReplicas** < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Deployments: 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 Deployments: 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.
3	minReplicas < **targetReplicas** < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Deployments: 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.

Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
4	minReplicas < **targetReplicas** = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Deployments: 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.
5	minReplicas ≤ replicas < **targetReplicas** < maxReplicas	6	5	1/10	HPA: 6/10 Deployments: 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 Deployments: 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**.

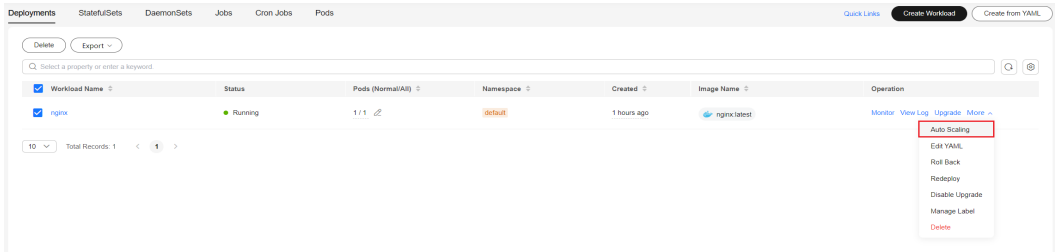
Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
7	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Deployments: 11	When the value of **targetReplicas** is greater than that of **maxReplicas**: • 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 3-278 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 **3.12.2.2 HPA Policies**.

Figure 3-279 Enabling the HPA policy

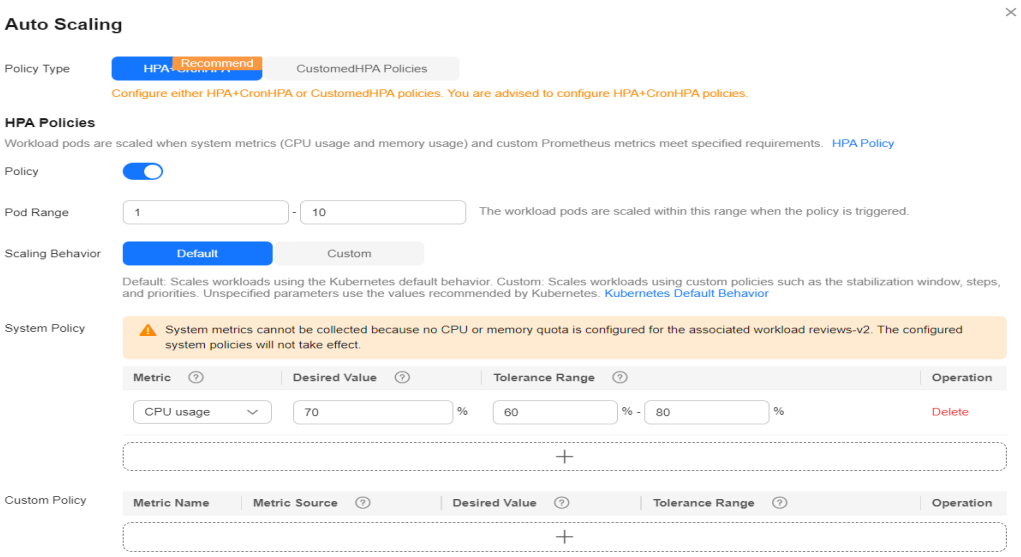


Table 3-343 HPA policy

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
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 = CPUs or memory used by pods/Requested CPUs or memory. • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. 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. 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 **3.9.6.1 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 3-280 Enabling the CronHPA policy

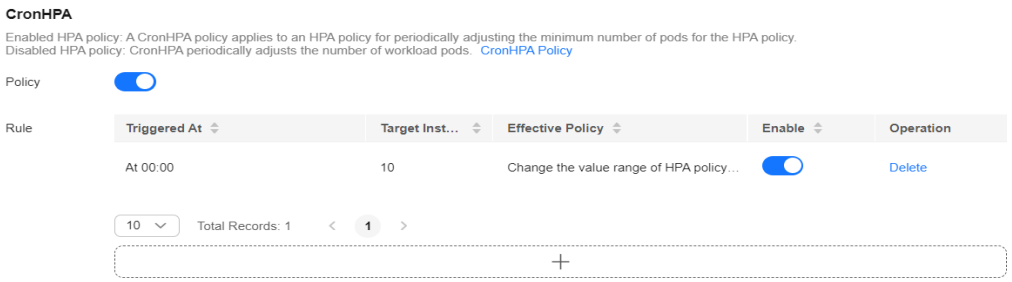


Table 3-344 CronHPA policy parameters

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 3-342**.
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 3-345 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

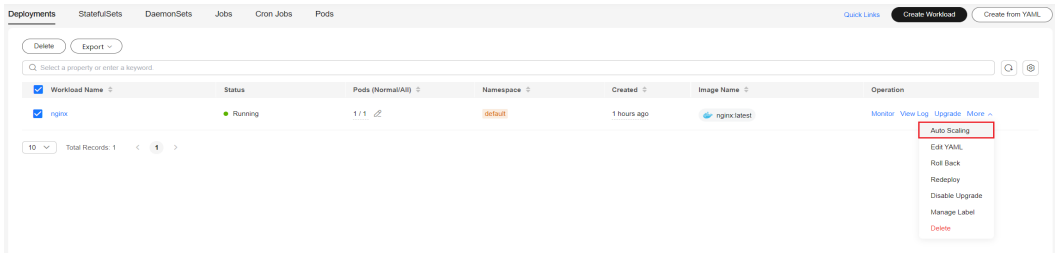
CronHPA adjusts associated Deployments separately to periodically adjust the number of Deployment pods. The method is as follows:

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 3-281 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 3-282 Using CronHPA to adjust the number of workload pods

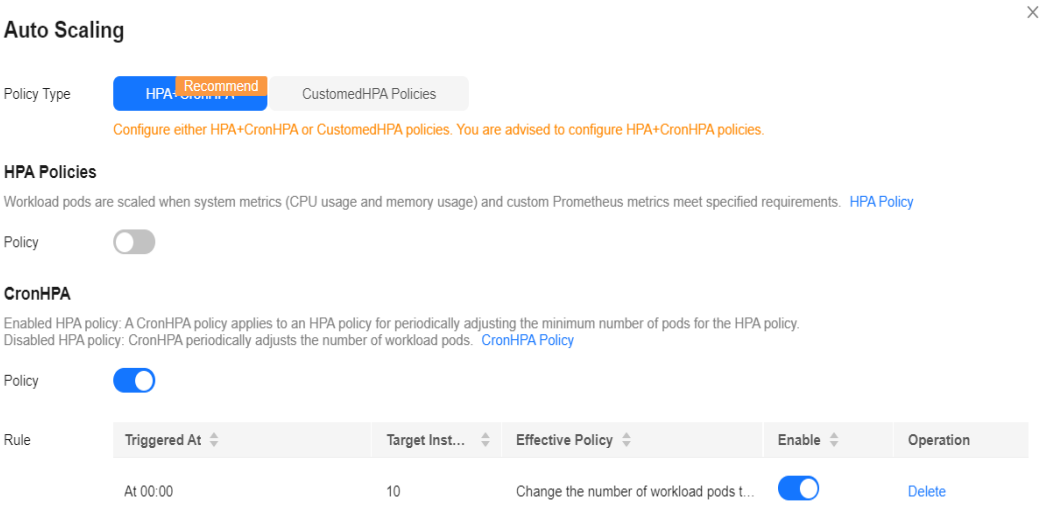


Table 3-346 CronHPA policy parameters

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
```

3.12.2.4 CustomedHPA Policies

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 **3.14.9 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:

- **3.14.8 Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
- **3.14.17 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 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**.
- **3.14.23 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.

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 volume mounted, a new pod cannot be started because EVS disks cannot be attached.
- The specifications of the CCE Advanced HPA add-on are decided based on the total number of containers in the cluster and the number of scaling policies. Configure 500m CPU cores and 1000 MiB memory for every 5000 containers, and 100m CPU cores and 500 MiB memory for every 1000 scaling policies.
- After a CustomedHPA policy is created, the type of its associated workload cannot be changed.

Creating a CustomedHPA policy

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 3-347 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.
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** The cooldown period takes effect only for metric-based policies. Periodic policies are not affected by the cooldown period.

Parameter	Description
Rules	Click +. In the dialog box displayed, set the following parameters: • **Type**: You can select **Metric-based** (Table 3-348) or **Periodic** (Table 3-349). 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 3-348 Metric-based rules

Parameter	Description
Trigger	Select **CPU usage** or **Memory usage**, choose **>** or **<**, and enter a percentage. **NOTE** Usage = CPUs or memory used by pods/Requested CPUs or memory.

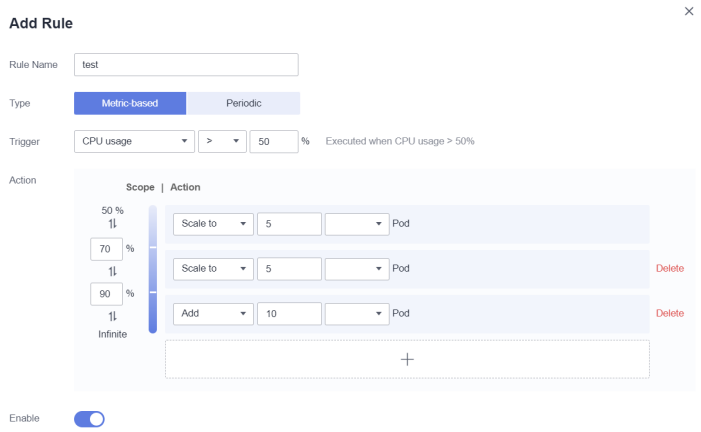
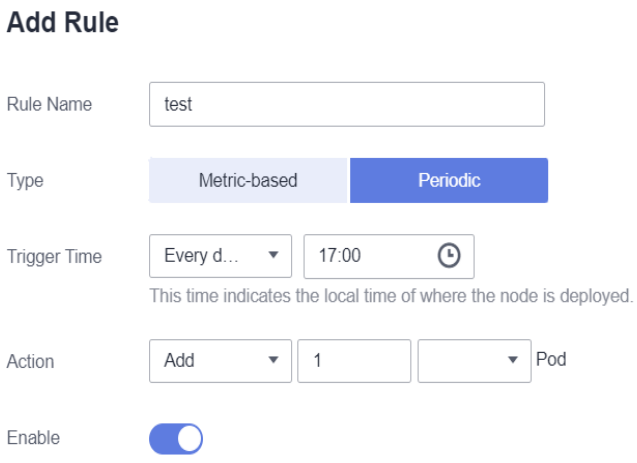
Parameter	Description
Action	Set an action to be performed when the trigger condition is met. Multiple actions can be added. • **Scale To**: Adjust the number of pods to the specified value. Both a number and a percentage will do. This action can be used to scale in or out pods. If the current number 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 3-283** Setting a trigger condition 

Table 3-349 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 3-284** Periodic triggering (Daily) 

Step 4 Click **Create**.

----End

Using kubectl

A CustomHPA 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
```

3.12.2.5 Managing Workload Scaling Policies

Scenario

After an HPA or CustomedHPA policy is created, you can update and delete the policy, as well as edit the YAML file.

Checking an HPA Policy

You can view the rules, status, and events of an HPA 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 page displayed, click the **HPA Policies** tab and then ⌄ next to the target HPA policy.

Step 3 In the expanded area, choose **View Events** in the **Operation** column. If the policy malfunctions, locate and rectify the fault based on the error message displayed on the page.

NOTE

You can also view the created HPA 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 view the HPA policies or CustomedHPA policies. You can also view the scaling policies you configured on the **Policies** page.

Table 3-350 Event types and names

Event Type	Event Name	Description
Normal	SuccessfulRescale	The scaling is performed successfully.
Abnormal	InvalidTargetRange	Invalid target range.
	InvalidSelector	Invalid selector.
	FailedGetObjectMetric	Objects fail to be obtained.
	FailedGetPodsMetric	Pods fail to be obtained.
	FailedGetResourceMetric	Resources fail to be obtained.
	FailedGetExternalMetric	External metrics fail to be obtained.
	InvalidMetricSourceType	Invalid metric source type.
	FailedConvertHPA	HPA conversion failed.
	FailedGetScale	The scale fails to be obtained.
	FailedComputeMetricsReplicas	Failed to calculate metric-defined replicas.
	FailedGetScaleWindow	Failed to obtain ScaleWindow.
	FailedRescale	Failed to scale the service.

----End

Viewing a CustomedHPA Policy

You can view the rules and latest status of a CustomedHPA policy and rectify faults 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 page displayed, click the **CustomHPA Policies** tab and then ⌄ next to the target CustomHPA policy.

Step 3 In the expanded area, if the policy is abnormal on the **Rules** tab page, click **Details** in **Latest Status** and locate the fault based on the information displayed.

NOTE

You can also view the created HPA 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 view the HPA policies or CustomedHPA policies. You can also view the scaling policies you configured on the **Policies** page.

----End

Editing an HPA or CustomedHPA Policy

An HPA policy is used as an example.

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 **HPA Policies** tab. Locate the row containing the target policy and choose **More** > **Edit** in the **Operation** column.

Step 3 On the **Edit HPA Policy** page, configure policy parameters listed in **Table 3-341**.

Step 4 Click **OK**.

----End

Editing the YAML File (HPA 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 **HPA Policies** tab. Locate the row containing the target policy and click **Edit YAML** in the **Operation** column.

Step 3 In the dialog box displayed, edit or download the YAML file.

----End

Viewing the YAML File (CustomedHPA 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 **CustomedHPA Policies** tab, locate the row containing the target policy, and choose **More** > **View YAML** in the **Operation** column.

Step 3 In the dialog box displayed, copy and download the YAML file but you are not allowed to modify it.

Step 4 Click the close button in the upper right corner.

----End

Deleting an HPA or CustomedHPA 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**. Choose **More** > **Delete** in the **Operation** column of the target policy.

Step 3 In the dialog box displayed, click **Yes**.

----End

3.12.3 Scaling a Node

3.12.3.1 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.

3.14.6 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 **3.14.6 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 user-defined percentage for scale-in, Autoscaler simulates whether the pods on the node can be migrated to other nodes. If yes, the node will be removed after an idle time window.

  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)

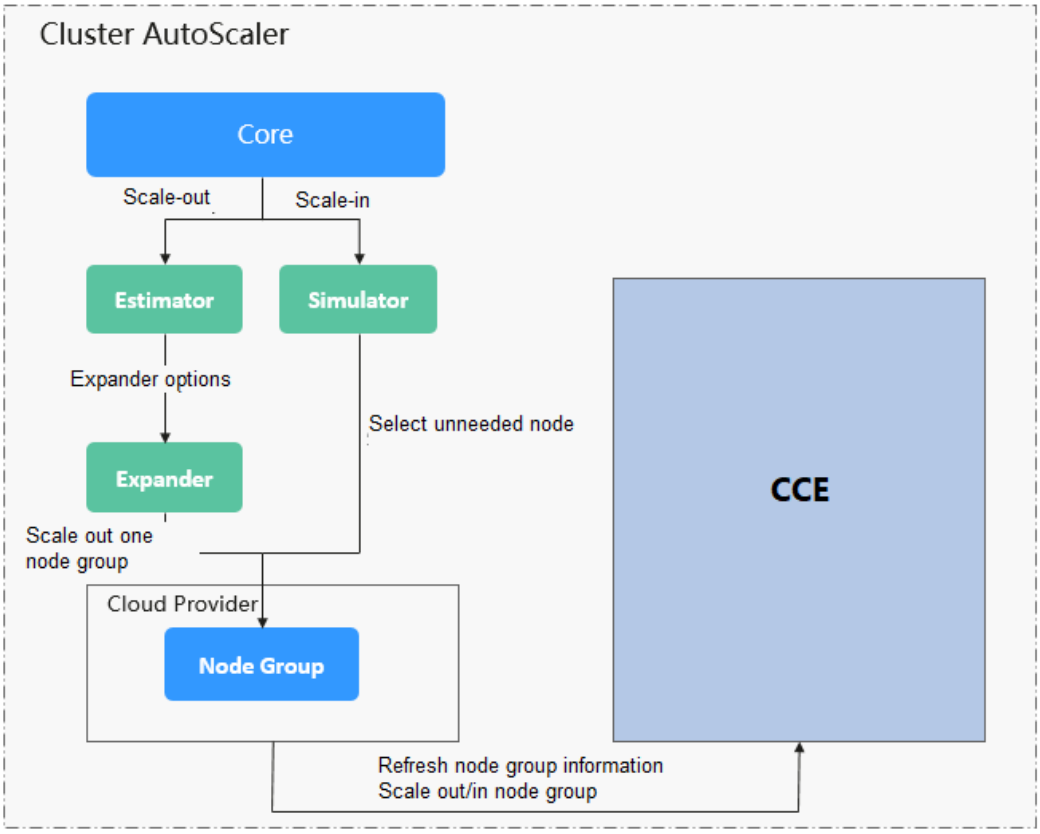
  NOTE

  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 3-285 shows the Cluster Autoscaler architecture and its core modules.

Figure 3-285 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. Currently, the Expander has the following policies:

Table 3-351 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 > 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.	This policy allows you to configure and manage the priorities of node pools or scaling groups through the console or API, while the least-waste policy can reduce the resource waste ratio in common scenarios. This policy has wider applicability and is used as the default selection policy.	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
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).

3.12.3.2 Creating a Node Scaling Policy

CCE provides auto scaling through the **3.14.6 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 auto scaling add-on take effect, for example, there are pods that cannot be scheduled and the value of a metric reaches the threshold, scale-out is 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 **3.14.6 CCE Cluster Autoscaler** add-on of v1.13.8 or later in the cluster.

Constraints

- Only pay-per-use node pools support auto scaling.
- 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 determines that the node is not fully available and the node scaling rules triggered using the CPU or memory metrics will not take effect.
- 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, 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.
- When 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 affects 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 the auto scaling add-on 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 **3.14.6 CCE Cluster Autoscaler**.
- If the auto scaling add-on has been installed, directly configure auto 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 3-352 Custom rules

Rule Type	Configuration
Metric-based	– **Trigger**: Select **CPU allocation rate** or **Memory allocation rate** and enter a value. The value must be greater than the scale-in percentage configured in the auto scaling add-on. **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 based on the **CPU allocation rate** and **a periodic rule** are configured and they both meet the scale-out conditions, one of them will be executed randomly. The rule executed first (rule A) changes the node pool to the scaling state. As a result, the other rule (rule B) cannot be executed. After rule A is executed and the node pool status becomes normal, rule B will not be executed. ▪ If rules based on the **CPU allocation rate** and **memory allocation rate** are configured, 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. ▪ When the number of nodes in the cluster reaches the upper limit, or the CPU or memory usage reaches the upper limit of the autoscaler add-on, node 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
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

Specification selection: Configure whether to enable auto scaling for node flavors in a node pool.

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 After the configuration is complete, click **OK**.

----End

Configuring an Auto Scaling Policy for a Cluster

NOTE

An auto scaling policy takes effect on 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.

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 node has been configured to be affinity for pods, no node will not be automatically added when pods cannot be scheduled. Such auto scaling typically works with an HPA policy. For details, see **3.12.4 Using HPA and CA for Auto Scaling of Workloads and Nodes**.

  If this function is not enabled, scaling can be performed only using **custom scaling policies**.
- **Upper limit of resources to be expanded**: Configure an upper limit for the total resources in the cluster. When the upper limit is reached, nodes will not be automatically added.

  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 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 could not be removed. Default value: 5 minutes.
- **Cooldown Time**
  - **Scale-in Cooldown Time After Scale-out**: 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 Node Deletion**: Default value: 10 minutes.
  - **Scale-in Cooldown Time After Failure**: 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, and scaling policies). This interval takes effect in 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 takes effect in 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 takes effect in 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 for a scale-out in the next 10 seconds until the expected node is added or all node pools are 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, Cluster 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, Cluster Autoscaler tolerates the node for a maximum of 15 minutes. After the tolerance period expires, Cluster 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 3-353 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.
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.

Parameter	Type	Description
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. Currently, only **>** is supported.
spec.rules[x].metricTrigger.metricValue	String	Metric threshold of a metric-based rule. The value can be any integer from 1 to 100 and must be a character string.
spec.rules[x].metricTrigger.Unit	String	Unit of the metric-based rule threshold. Currently, only **%** is supported.
spec.targetNodepoolIds	Array	All node pools associated with the scaling policy.
spec.targetNodepoolIds[x]	String	ID of the node pool associated with the scaling policy.

3.12.3.3 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.

----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.

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.

Step 3 On the **Edit Node Scaling Policy** page displayed, configure policy parameters listed in **Table 3-353**.

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.

Step 3 On the **Clone 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.

Step 3 In the dialog box displayed, confirm whether to disable or enable the node policy.

----End

3.12.4 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 pods or containers are used for deploying applications, the upper limit of available resources is typically required to set for pods or containers to prevent unlimited usage of node resources during peak hours. However, after the upper limit is reached, an application error may occur. To resolve this issue, scale in the number of pods to share workloads. If the node resource usage increases to a certain extent that newly added pods cannot be scheduled, scale in the number of nodes based on the node resource usage.

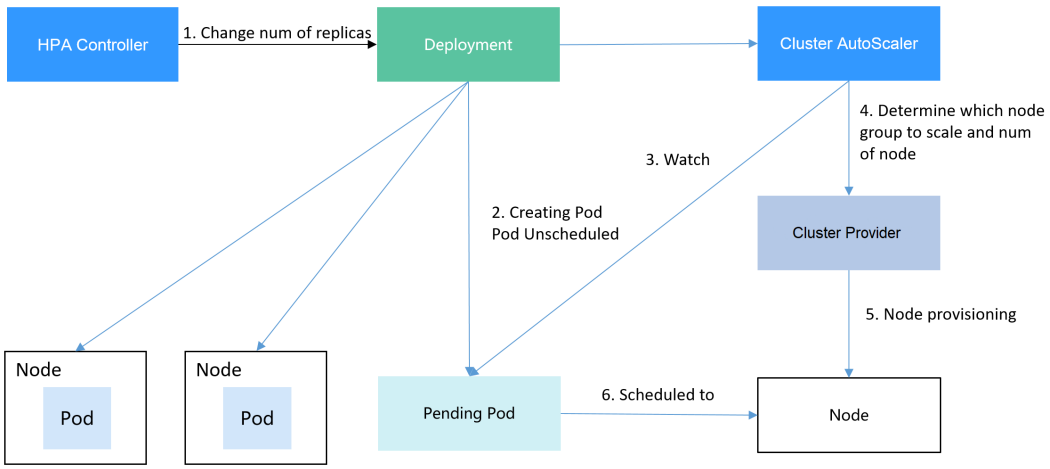
Solution

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 3-286**, 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 3-286 HPA and CA working flows

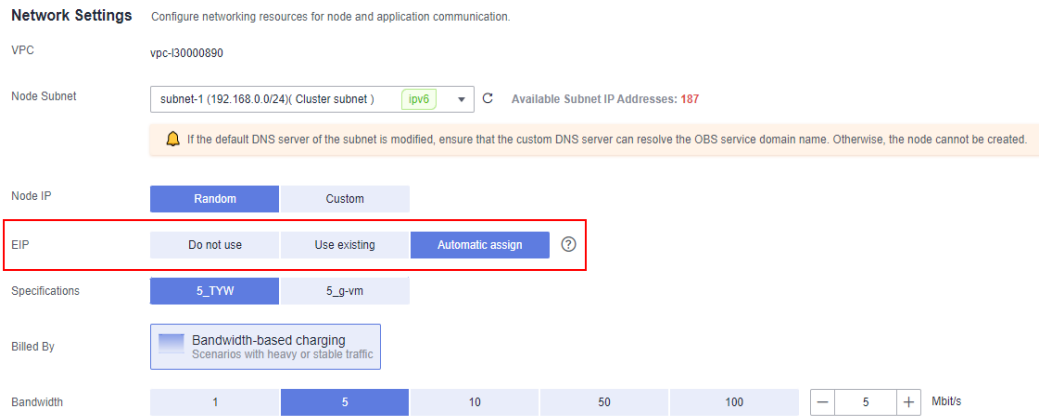


Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

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 to create an organization.

   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.

- **Nodes**: Set it to **1**, indicating that one node is created by default when a node pool is created.
- **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**.

- **Automatic scale-out**: If this function is enabled, nodes in a node pool will be automatically added based on the cluster load.
- **Customized Rule**: 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.
- **Automatic scale-in**: If this function is enabled, nodes in a node pool will be automatically deleted based on the cluster load. For example, trigger scale-in when the node resource utilization is less than 50%.
- **AS Configuration**: Modify the node quantity range. During autoscaling, the number of nodes in a node pool is always within the configured quantity range.
- **AS Object**: Enable autoscaling for node specifications in a node pool.

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 Shared 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
```

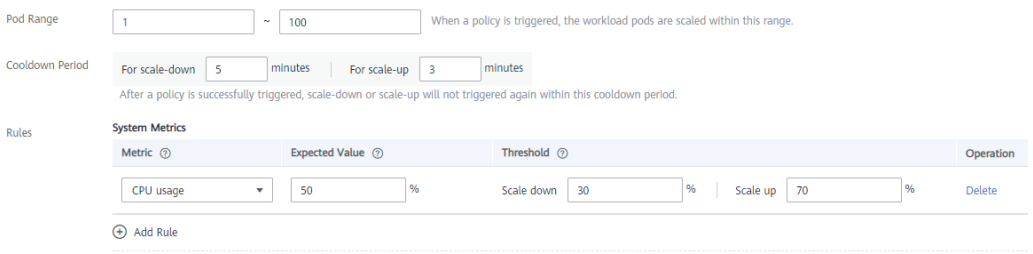
Creating an HPA Policy

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 Pending state. During this period, nodes are added.

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

Using HPA and CA can easily implement auto scaling in most scenarios. In addition, the scaling process of nodes and pods can be easily observed.

3.12.5 Elastic Scaling of CCE Pods to CCI

CCE Cloud Bursting Engine for CCI 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 scale Deployments, StatefulSets, Jobs, and CronJobs running in CCE clusters to **Cloud Container Instance (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 on the left, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Install**.
5. Configure the add-on parameters.

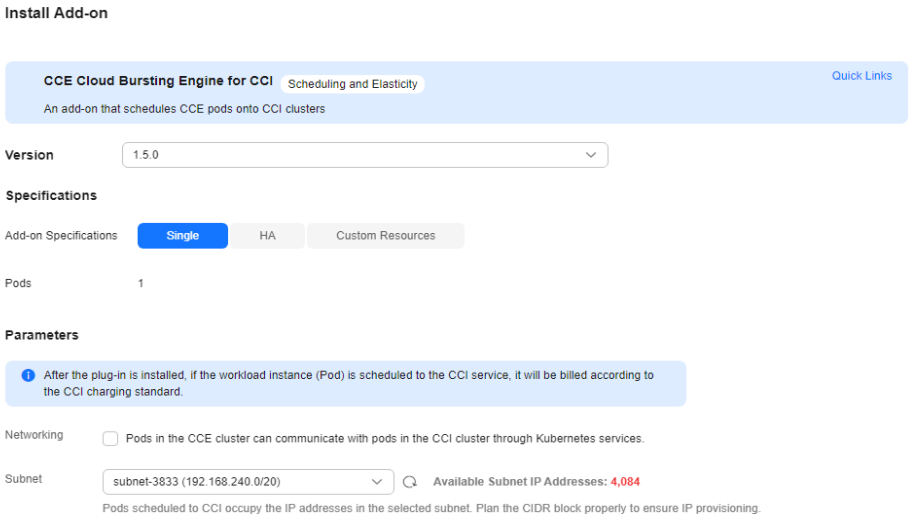
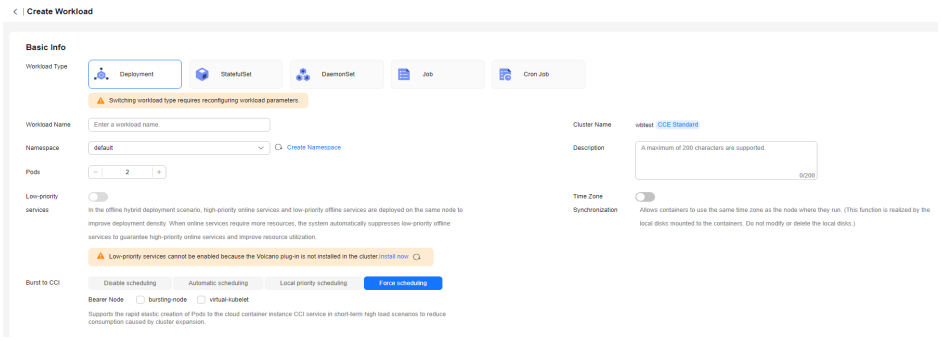


Table 3-354 Add-on parameters

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 a workload.
Networking	If this option is enabled, pods in a CCE cluster can communicate with the pods in CCI. 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 on the left, 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 **Scaling Pods to CCI**.



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 on the left, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

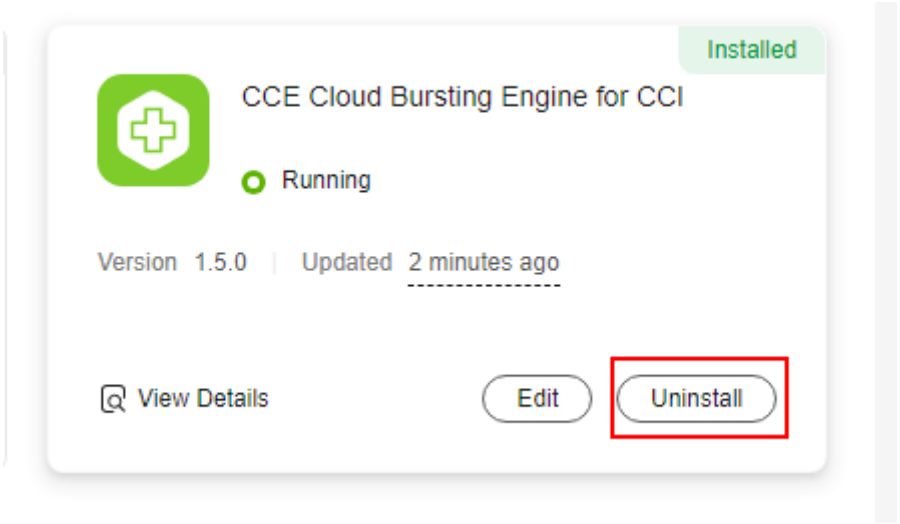


Table 3-355 Special scenarios for uninstalling the add-on

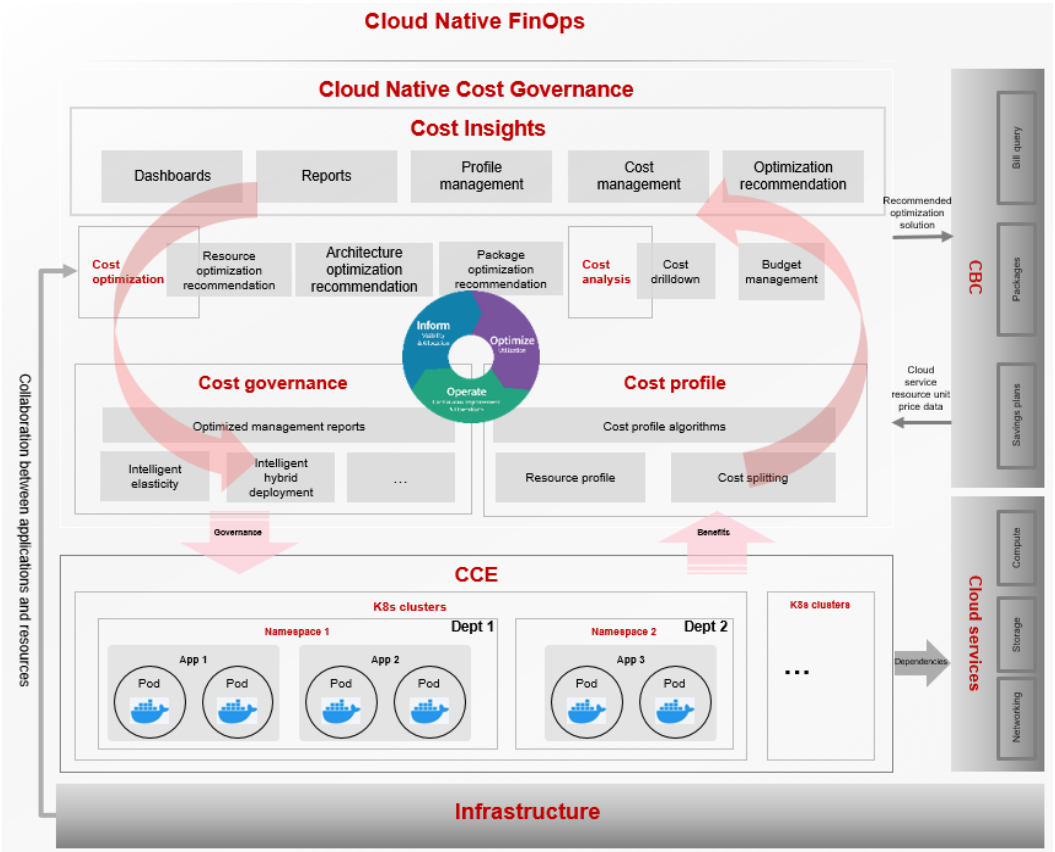
Scenario	Symptom	Description
There are no nodes in the CCE cluster that the bursting add-on needs to be uninstalled from.	Failed to uninstall the bursting add-on.	If the bursting add-on is uninstalled from the cluster, a job for clearing resources will be started in the cluster. To ensure that the job 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**.

3.13 Cloud Native Cost Governance

3.13.1 Overview

Cloud native cost governance is a FinOps-based solution for managing container costs. It provides cost and resource profiles from the department, cluster, and namespace dimensions. It leverages optimization techniques like workload resource recommendations to help IT cost management personnel improve resource utilization and reduce costs of container clusters.



Cost Insights

Cost insights use in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on actual bills and cluster resource usage. Cost insights help cost management personnel analyze cluster costs and resource usage, as well as identify resource waste for cost optimization.

3.13.2 Cost Insights

3.13.2.1 Overview

Cost insights use in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on actual bills and cluster resource usage. Cost insights help cost management personnel analyze cluster costs and resource usage, as well as identify resource waste for cost optimization.

Cost insights display container cost usage of a region and cluster.

- The cost insights for a region provide enterprise management personnel with an overall analysis report of container costs in that region. Users can divide departments by cluster or namespace and obtain the cost analysis report of each department. By checking reports, enterprise management personnel can identify trends in cost growth and compare department costs to design better cost management solutions.
- The cost insights for a cluster help cost O&M personnel analyze the cluster cost and resource usage from multiple dimensions, such as namespaces, applications, and node pools, to identify applications that can be optimized.

Key Capabilities

- Diverse cost types: The costs consist of management costs of a CCE cluster and costs of ECS and EVS resources associated with this CCE cluster.
- Precise cost calculation based on bills: Real bills are used to accurately collect statistics on cluster costs for cost allocation.
- Flexible cost allocation policies: Cost visualization and cost allocation policies are supported in multiple dimensions, such as clusters, namespaces, node pools, and applications.
- Long-term cost data storage and retrieval: Costs of up to two years can be analyzed.
- Workload billing in minutes: Workload discovery and billing in minutes are available for rapid scaling of applications. All costs will be billed.

Constraints

- Currently, only EVS disk storage costs are collected.
- Node costs are split based on CPUs and memory. The costs of heterogeneous resources, such as GPUs and NPUs, cannot be split. For example, when a GPU node is split, the core-hour unit price is high.
- After Cost Insights is enabled, analysis results can be displayed only after two days.
- Cost insights display cost analysis results by day.

3.13.2.2 Cost Calculation Model

Workload Cost Calculation Principle

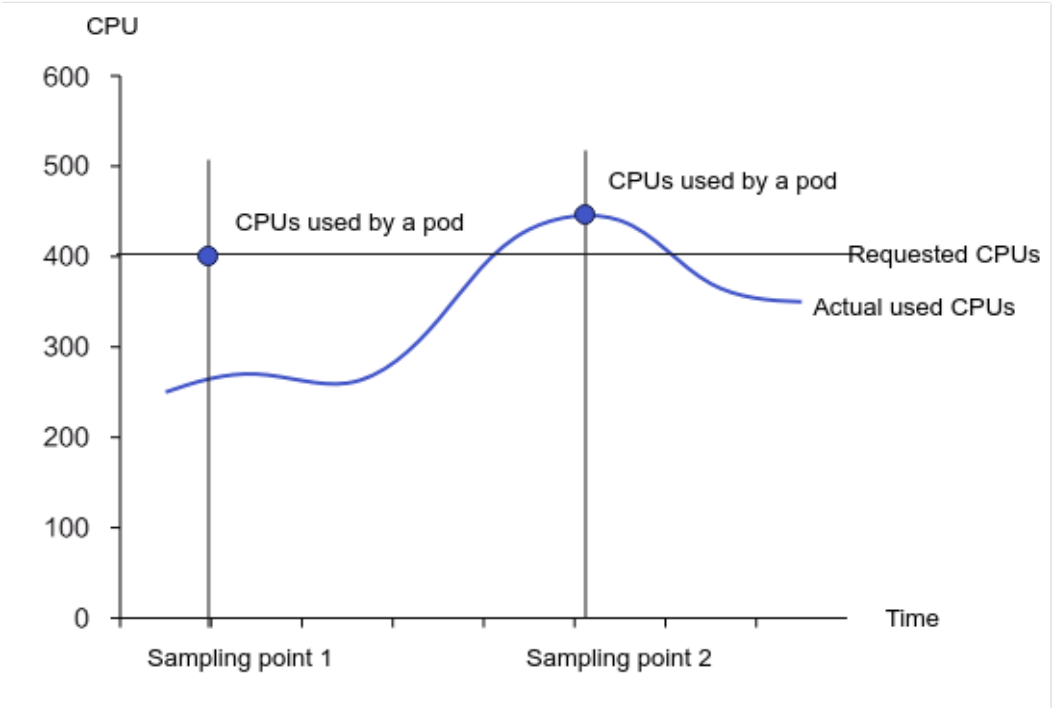
The cost of a workload is the total cost of all pods in that workload.

- Pod cost: the cost 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.

$$\text{Pod cost} = \frac{\max(\text{CPUs requested by a pod}, \text{CPUs used by a pod})}{\text{Node CPU capacity}} \times \text{Node cost} \times \text{CPU usage} + \frac{\max(\text{Memory requested by a pod}, \text{Memory used by a pod})}{\text{Node memory capacity}} \times \text{Node cost} \times \text{Memory usage} + \text{Cost of PVC storage used by that pod}$$

  The larger value between the requested CPUs and the actual used CPUs is the CPUs used by the pod.

Figure 3-287 Workload cost calculation principle



For example, at sampling point 1, the value of requested CPUs is larger than that of actual used CPUs, the used CPUs is the value of requested CPU.

At sampling point is 2, the value of requested CPUs is less than that of actual used CPUs, the used CPUs is the value of actual used CPUs.

- Workload cost: the total cost of all pods in a workload

$$\text{Workload cost} = \sum_{\text{Workload}}^{all} \text{Pod cost}$$

Namespace Cost Calculation Principle

The cost of a namespace is the total cost of all workloads in the namespace.

$$\text{Namespace cost} = \sum_{\text{Namespace}}^{all} \text{Workload cost}$$

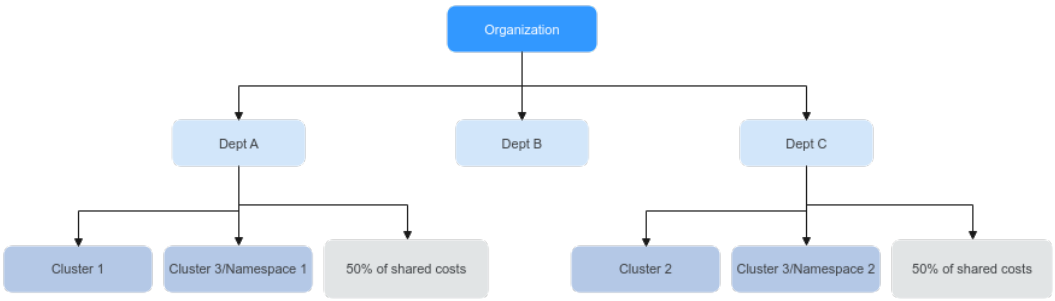
Department Cost Calculation Principle

A department is a logical cost unit used to aggregate and analyze the costs of different clusters and namespaces. To adapt to actual scenarios, the cost unit is set based on an actual business department and associated with the cluster or namespace used by the business department.

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 a department is set by namespace, the department needs to be associated with the namespace and the proportion of shared costs needs to be set.

Figure 3-288 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 department cost, you can allocate costs to corresponding departments by namespace or cluster, set the proportion of the shared cost generated in cluster 3 for department A and department C.

Common Issues During Calculation

- Does the cost of a pod depend on the requested resources or the actual resources used? Since resource metric values like used CPUs and memory change dynamically, how can you accurately estimate the pod cost?

  The larger value between the requested CPUs or memory and the actual used CPUs or memory is the CPUs or memory used by a pod. 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 the unallocated costs of a node handled?

  The unallocated costs of a node are not allocated to the workload or namespace cost, but can be shared among departments. The proportion of unallocated costs allocated to each department can be set.

3.13.2.3 Enabling Cost Insights

Cost insights use in-house cost profile algorithms to split costs by department, cluster, namespace, or application, based on actual bills and cluster resource usage. Cost insights help cost management personnel analyze cluster costs and resource usage, as well as 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 use 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 3-356**.

Table 3-356 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.

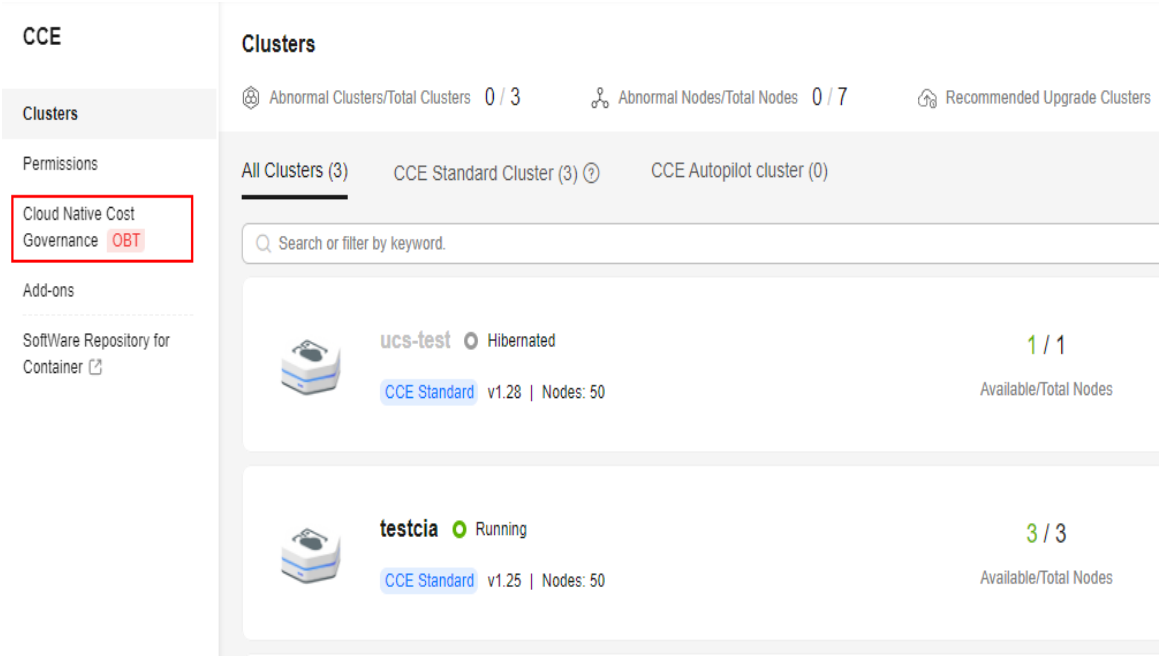
Constraints

- 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 3-289 Cloud Native Cost Governance

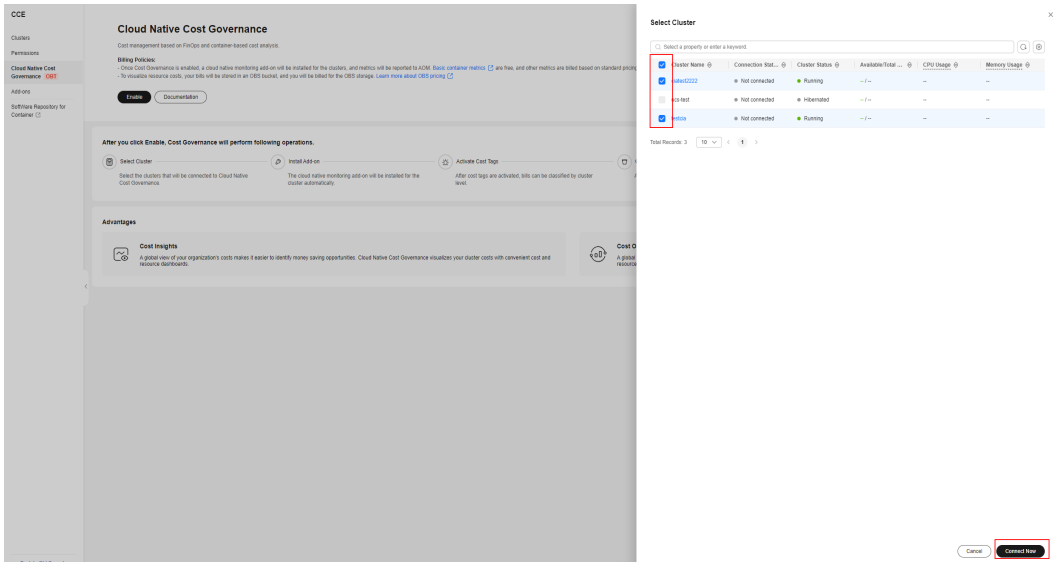


Step 2 Click **Enable**, select clusters that require cost insights, and click **Connect Now**.

The system will automatically: Wait for 3 to 5 minutes. The **Cost Insights** page is displayed.

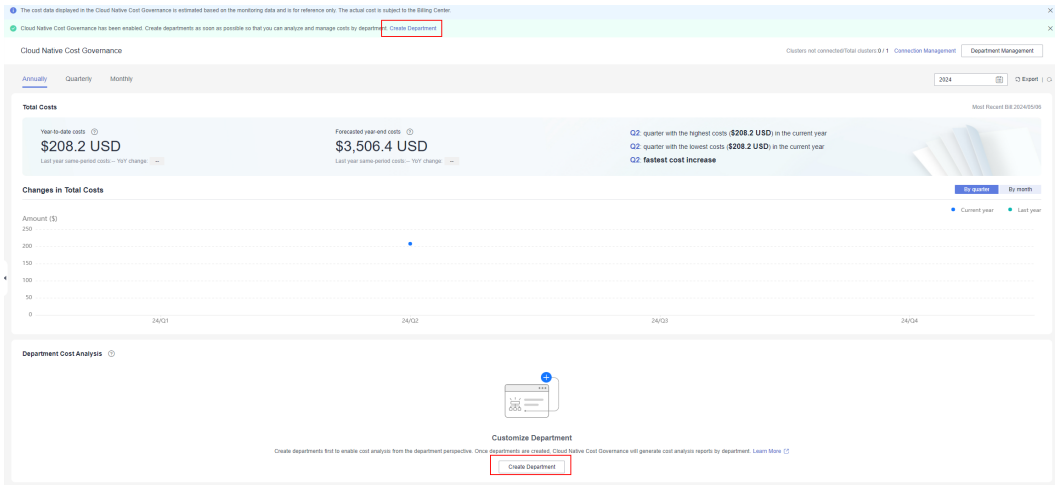
1. Install Cloud Native Cluster Monitoring to provide basic monitoring data for cost insights.
2. Activate cost tags. Cluster tags will be added to the bills exported from the billing center. The cost insights background classifies the bills by cluster. After this step is complete, the **CCE-Cluster-ID** 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 3-290 Enabling cost insights for clusters



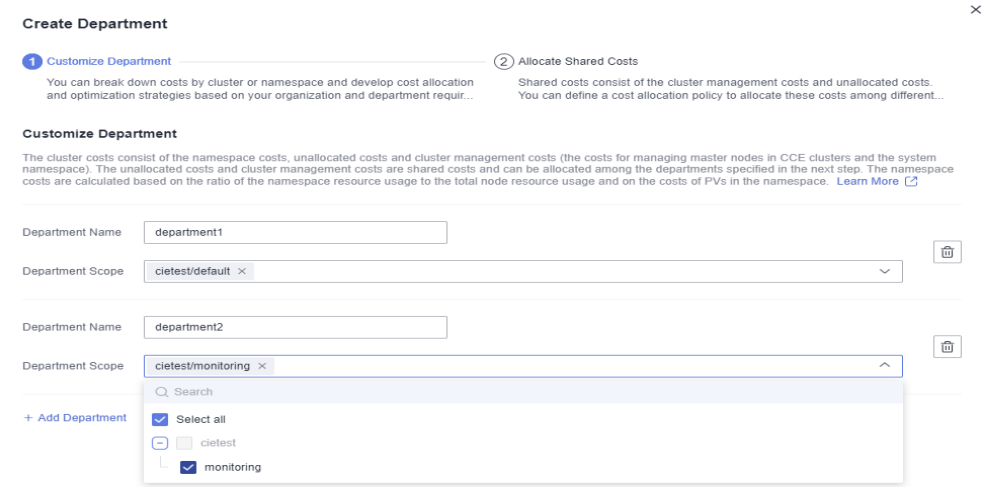
Step 3 (Optional) Click **Create Department**.

Figure 3-291 Creating departments



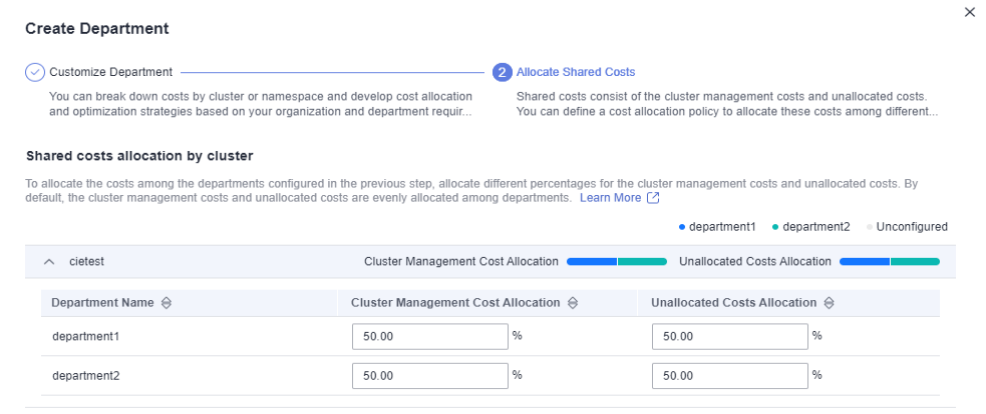
1. Customize departments based on the service requirements and associate them with clusters or namespaces as follows:
   - **Department Name**: Use the actual name of a business department.
   - **Department Scope**: Select the cluster or namespace for the business department. In the following example, **department1** and **department2** are configured by namespace.

   Figure 3-292 Customizing departments



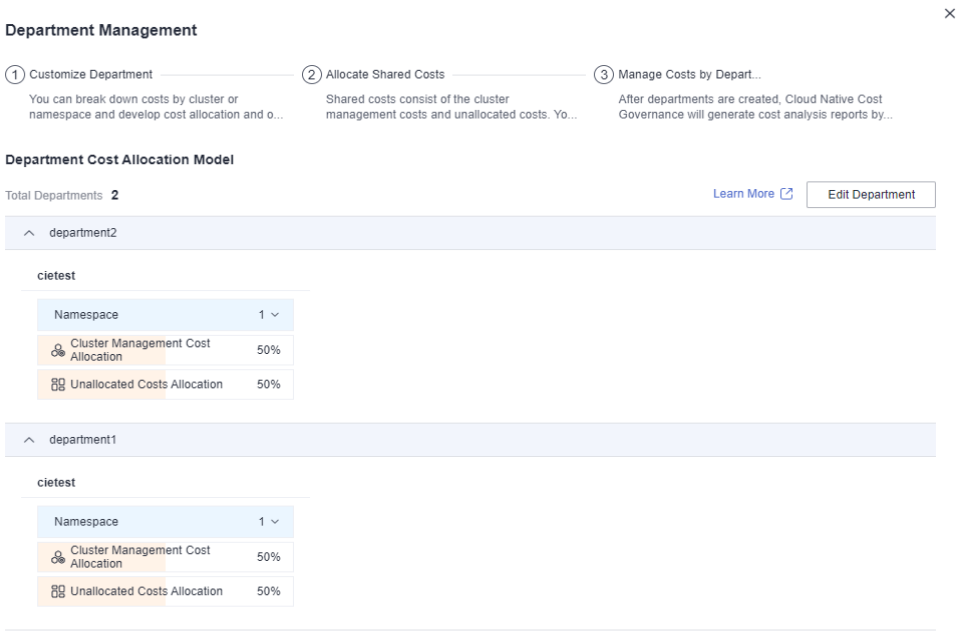
2. Allocate shared costs in a cluster to departments. Management costs and unallocated costs in the default cluster are evenly allocated among departments. The allocation ratio can be changed.

   Figure 3-293 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 3-294 Department configurations

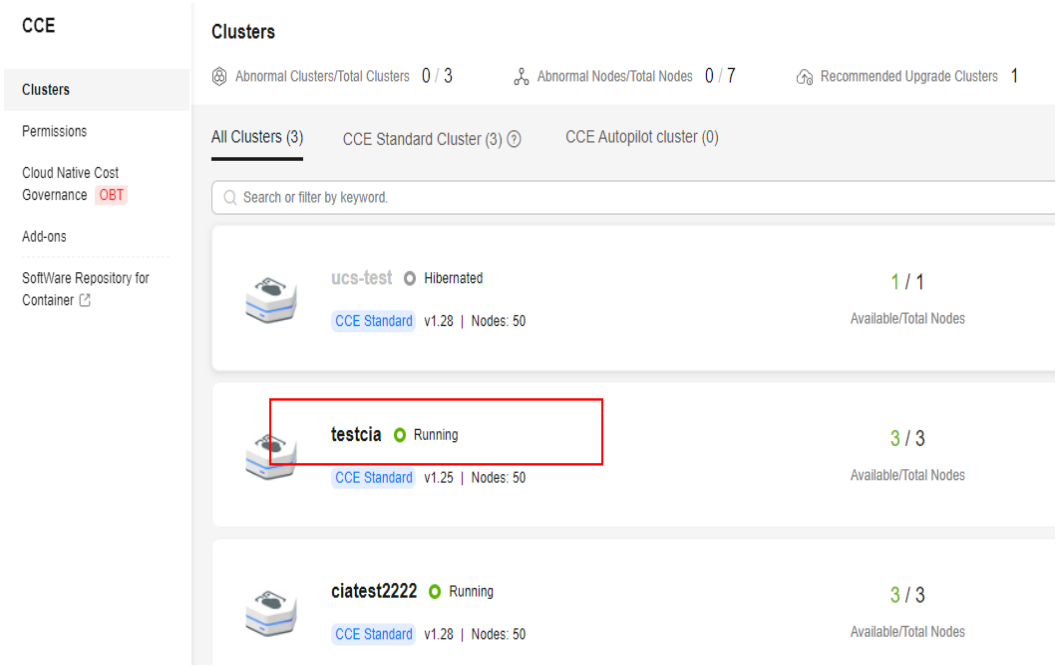


----End

Enabling Cost Insights for a Cluster

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

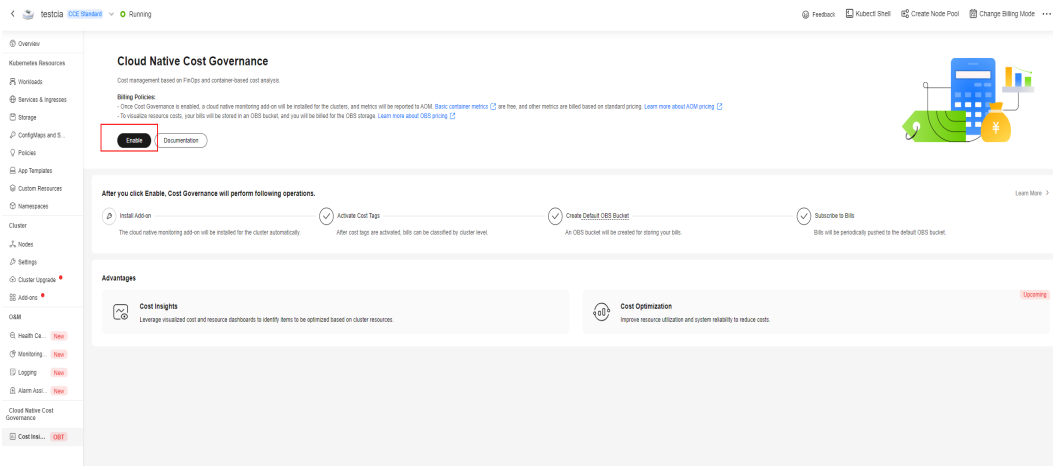
Figure 3-295 Accessing the cluster console



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 3-296 Enabling Cost Insights



----End

3.13.2.4 Cost Insights for a Region

The cost insights for a region provide enterprise management personnel with an overall analysis report of container costs in that region. From the perspective of cloud native, users 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.

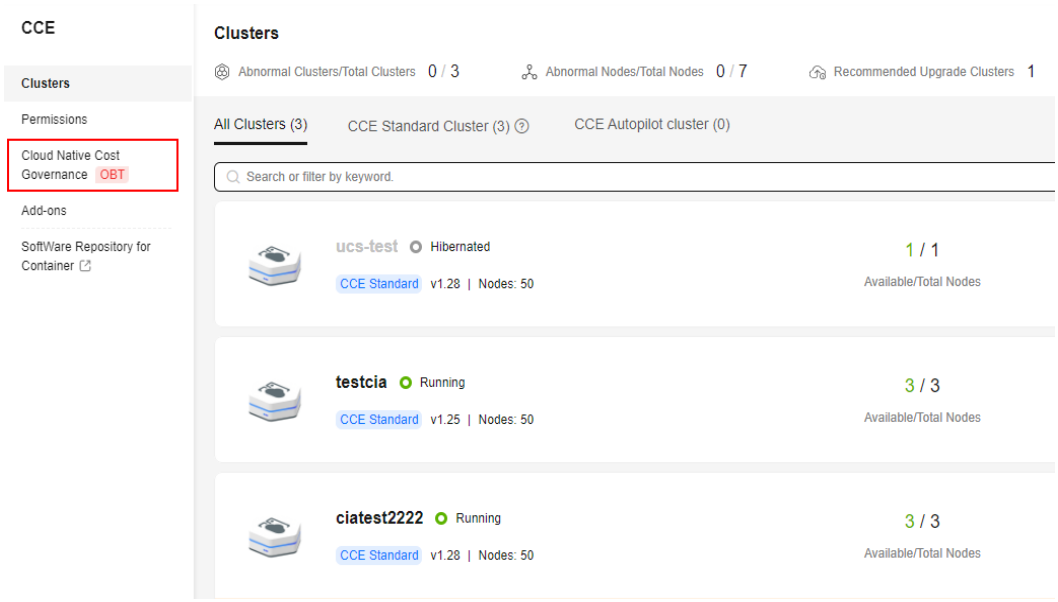
Constraints

- 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

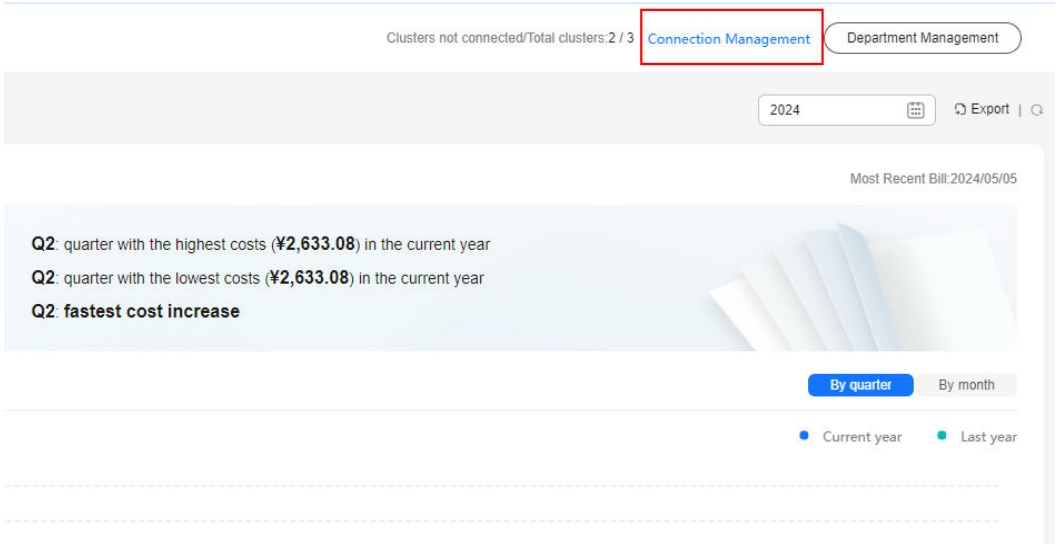
Step 1 Log in to the CCE console. In the navigation pane on the left, choose **Cloud Native Cost Governance**.

Figure 3-297 Cloud Native Cost Governance



Step 2 Click **Connection Management** to view the cluster connection status.

Figure 3-298 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 3-299 Batch connection

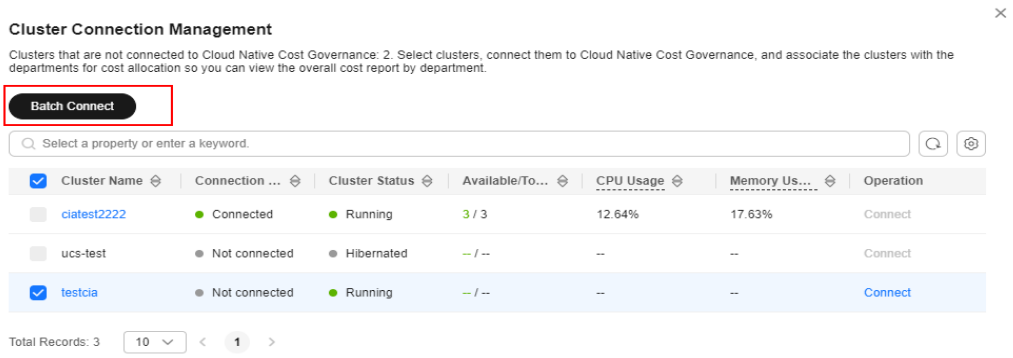
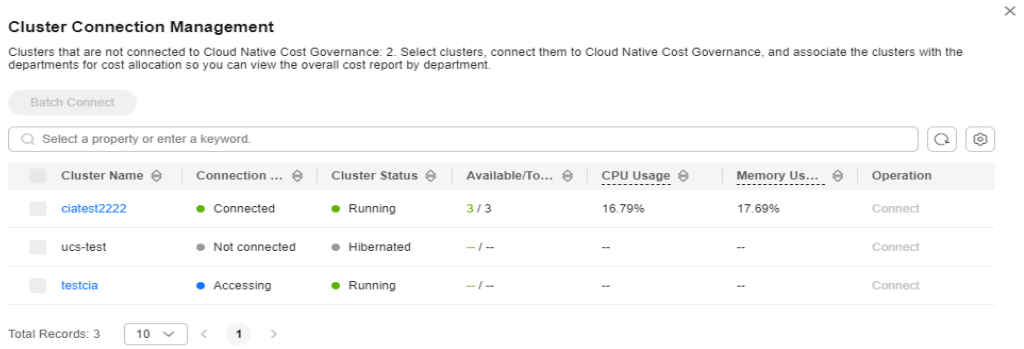


Figure 3-300 Cluster connection management

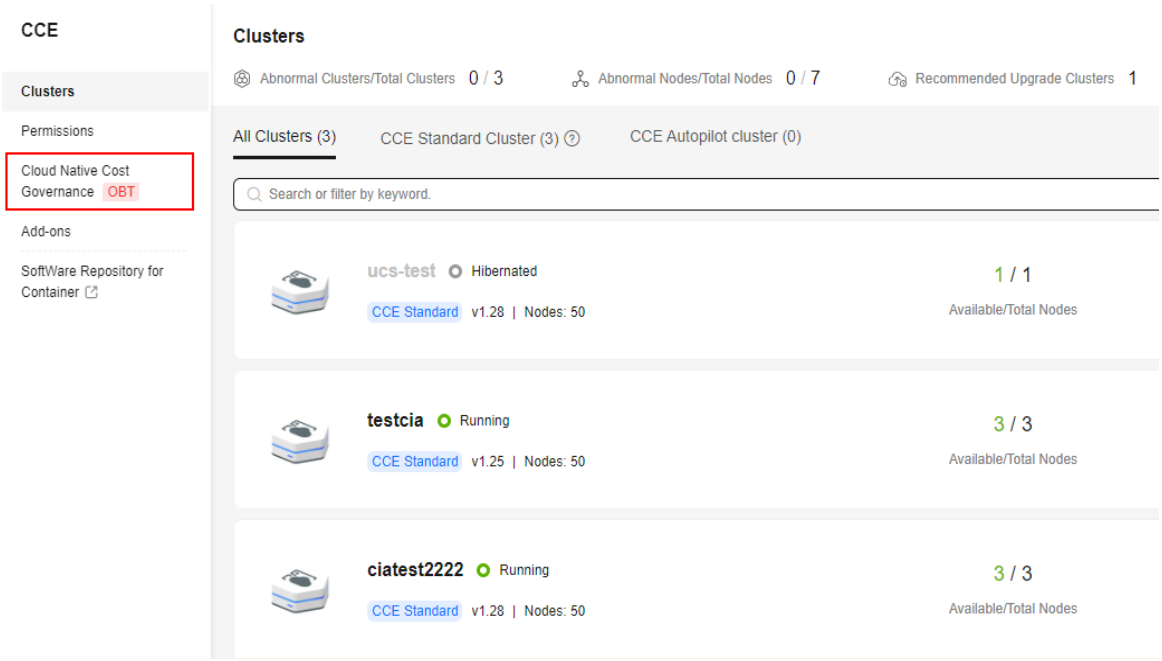


----End

Department Management

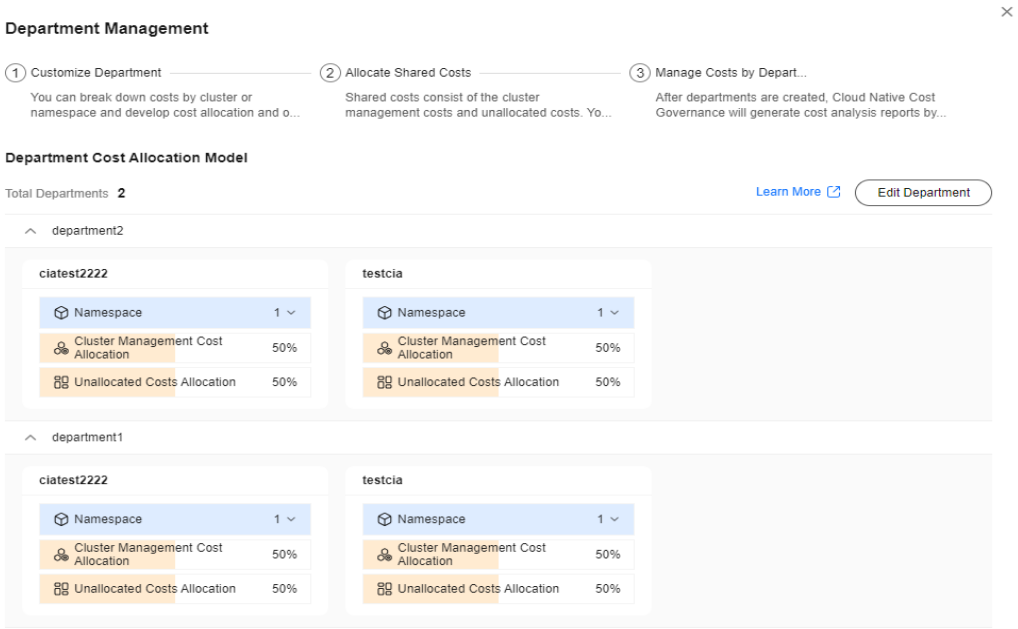
Step 1 Log in to the CCE console. In the navigation pane on the left, choose **Cloud Native Cost Governance**.

Figure 3-301 Cloud Native Cost Governance



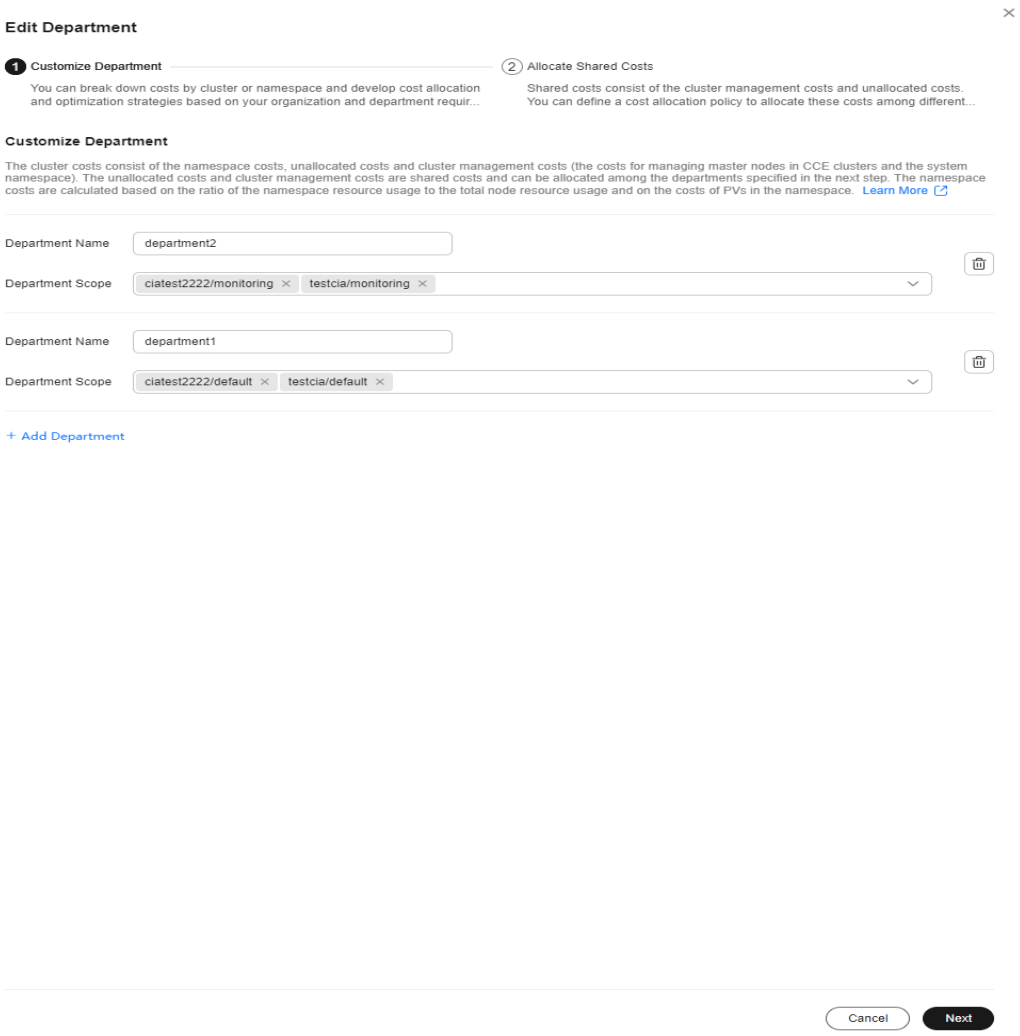
Step 2 Click **Department Management** to view the department configurations.

Figure 3-302 Viewing the department configurations



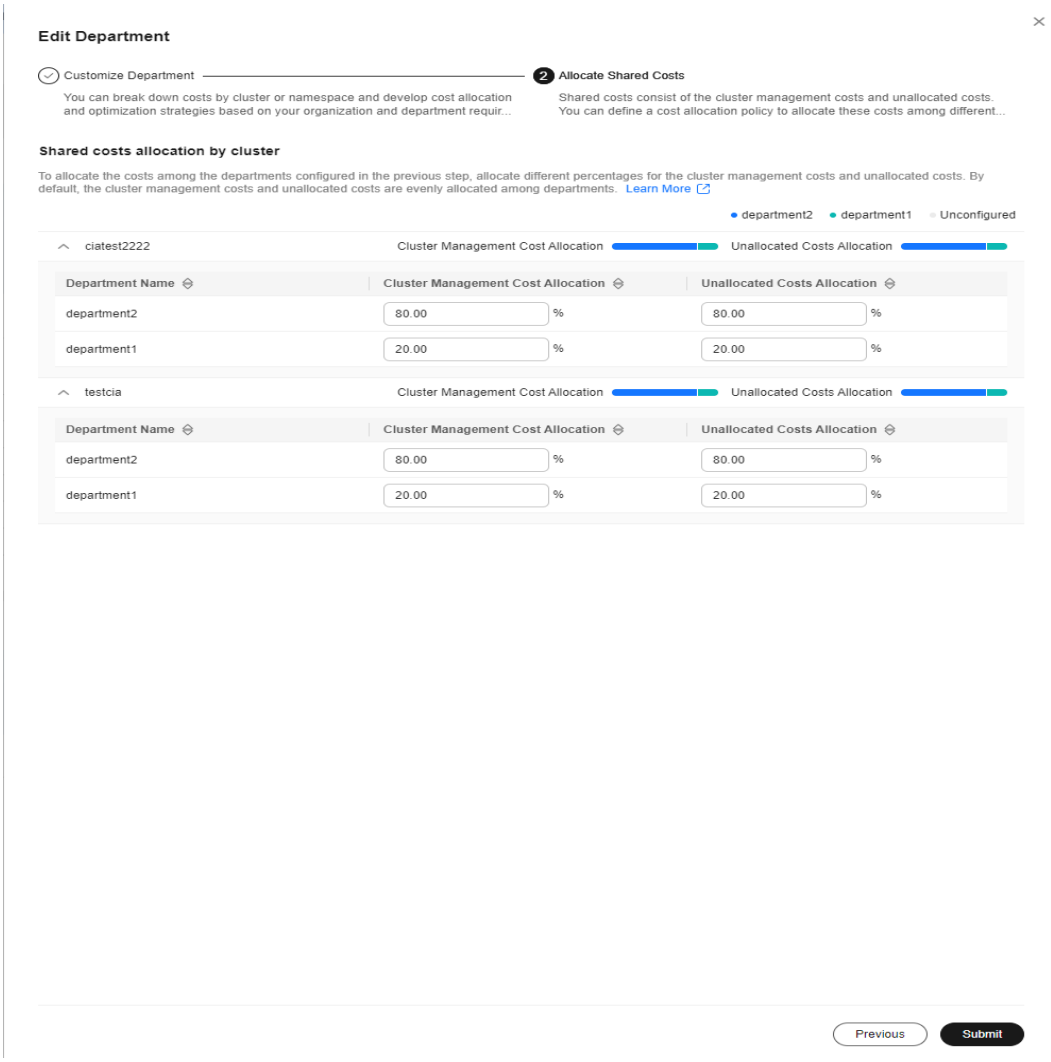
Step 3 Click **Edit Department** to modify the custom department configurations.

Figure 3-303 Modifying the department configurations



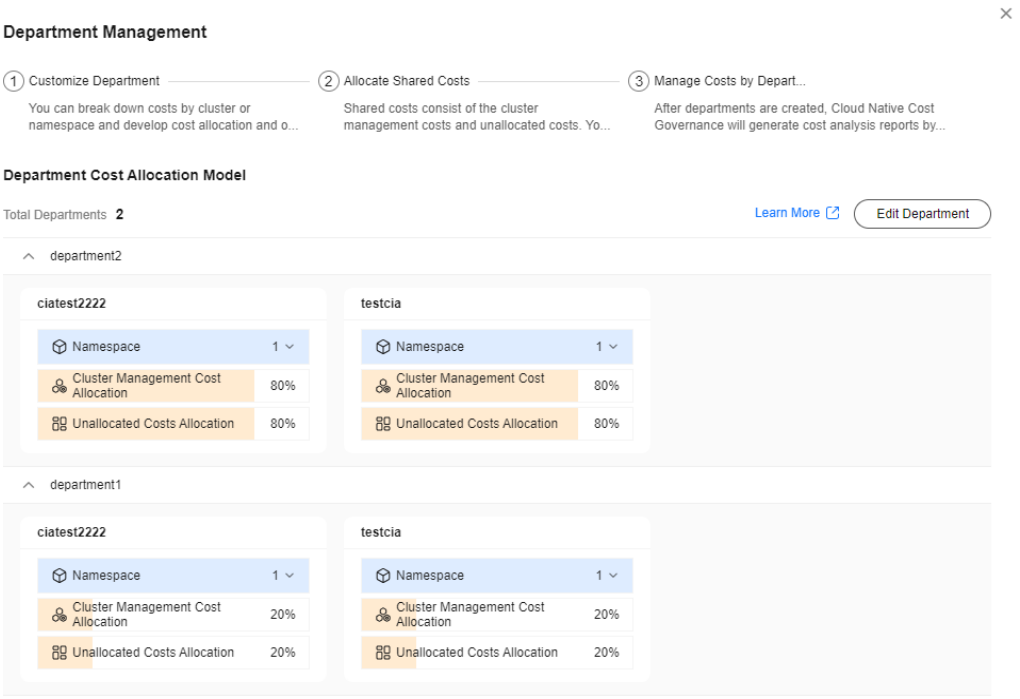
Step 4 Click **Next** to modify the percentages of shared costs allocated to departments.

Figure 3-304 Modifying shared costs allocation



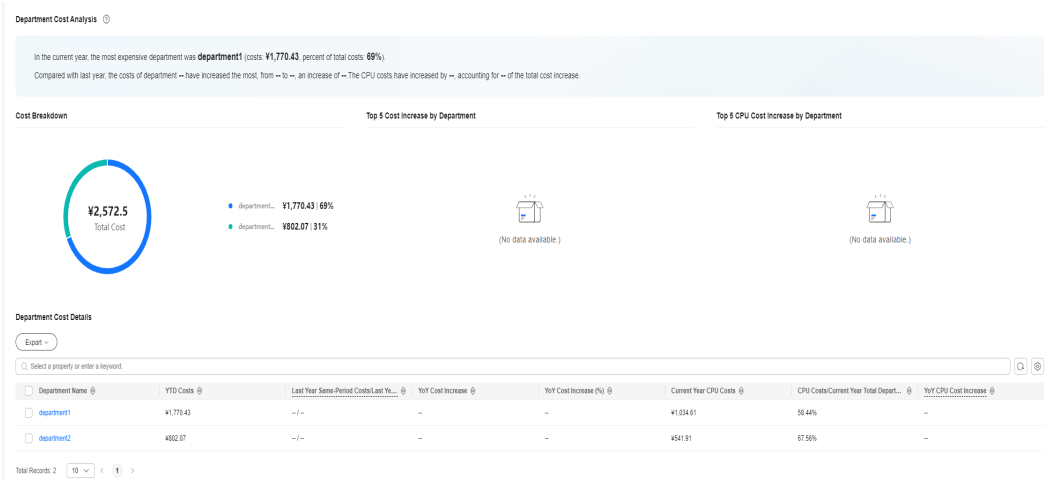
Step 5 Click **Submit**. The configurations are displayed on the department management page.

Figure 3-305 Submitting the modification



Step 6 Close the department management page and view cost analysis reports in **Department Cost Analysis**.

Figure 3-306 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 3-307 Viewing the cost status

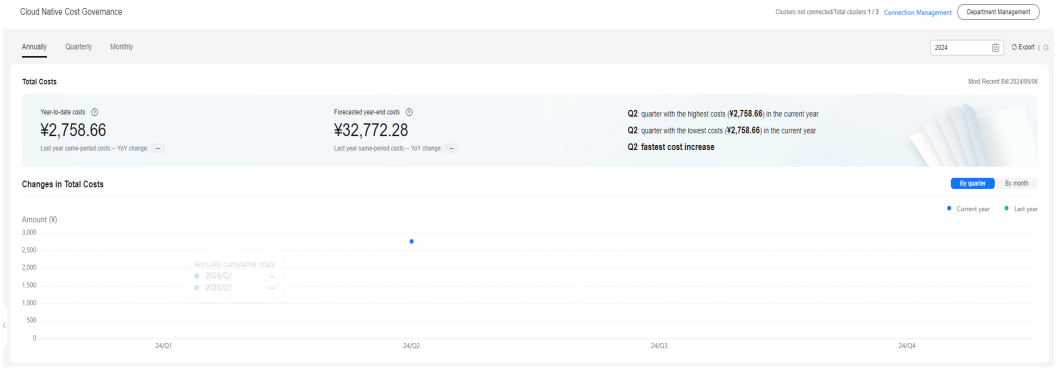


Table 3-357 Functions on the Cost Insights page

Function	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
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

Function	Report	Description
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
Changes in Total Costs	Annual, quarterly, and monthly	Cost details of the current year, quarter, and month and cost changes compared with the last year, quarter, and month

Department Cost Analysis

View department cost analysis reports in **Department Cost Analysis**.

Figure 3-308 Viewing department cost analysis reports

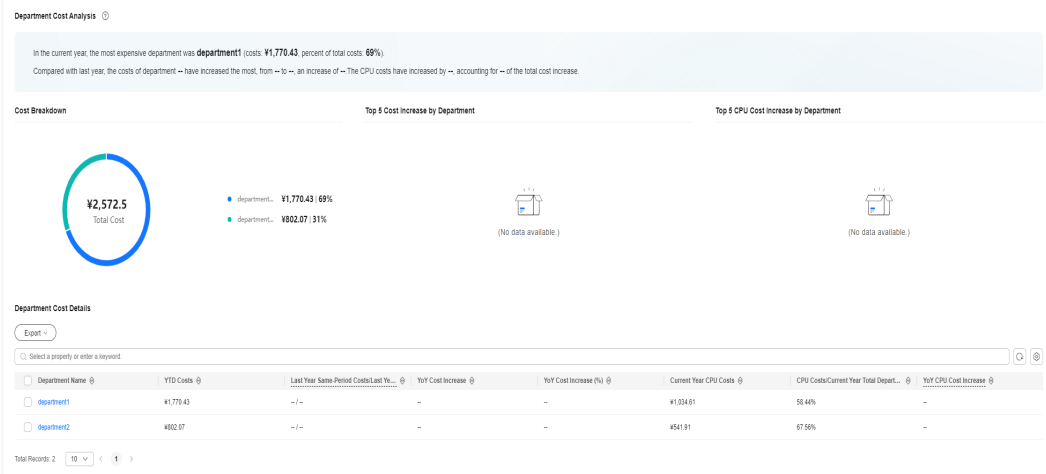


Table 3-358 Parameters in Department Cost Details

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
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

Parameter	Report	Description
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
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

3.13.2.5 Cost Insights for a Department

The cost insights for a department provide 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.

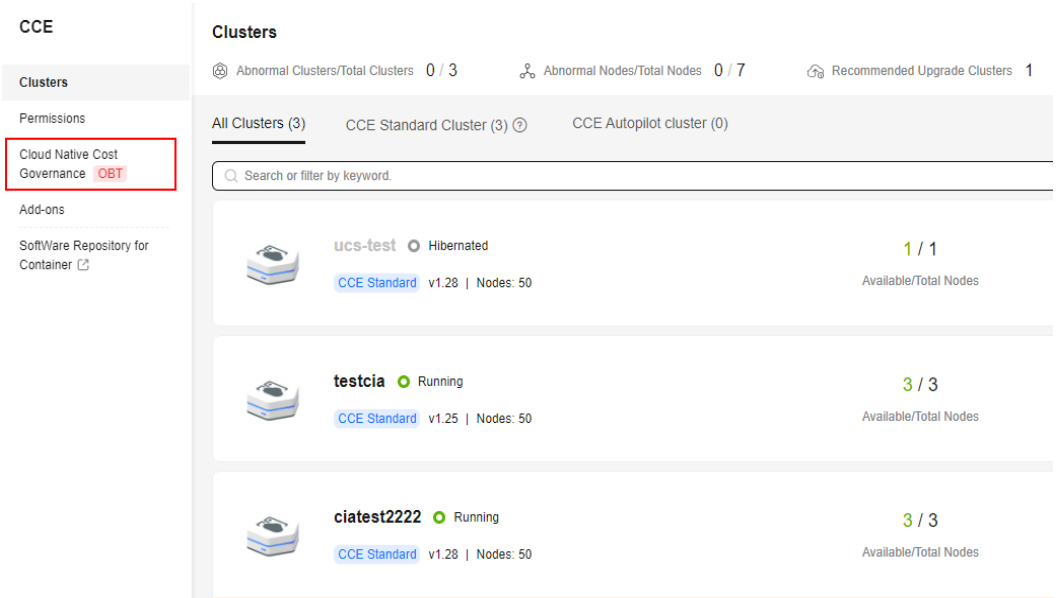
Constraints

- 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

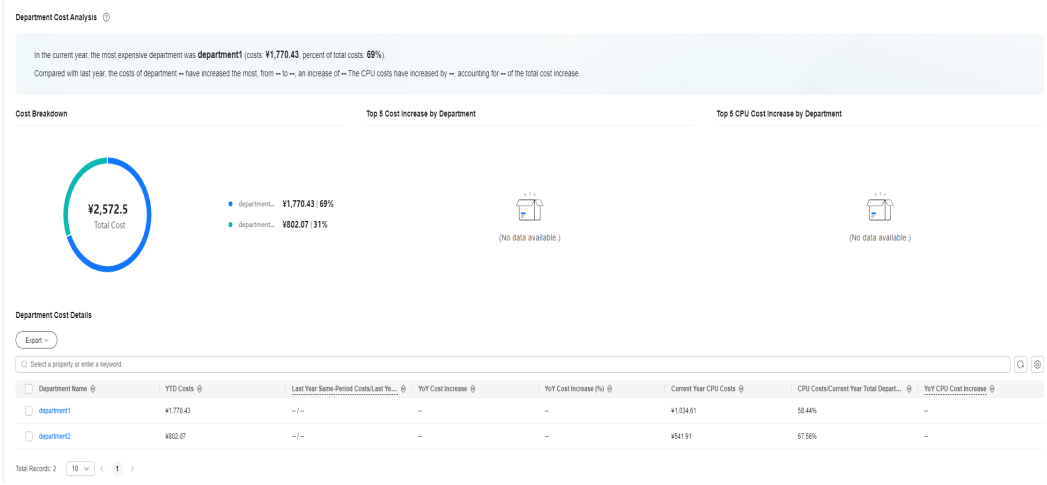
Step 1 Log in to the CCE console. In the navigation pane, choose **Cloud Native Cost Governance**.

Figure 3-309 Cloud Native Cost Governance



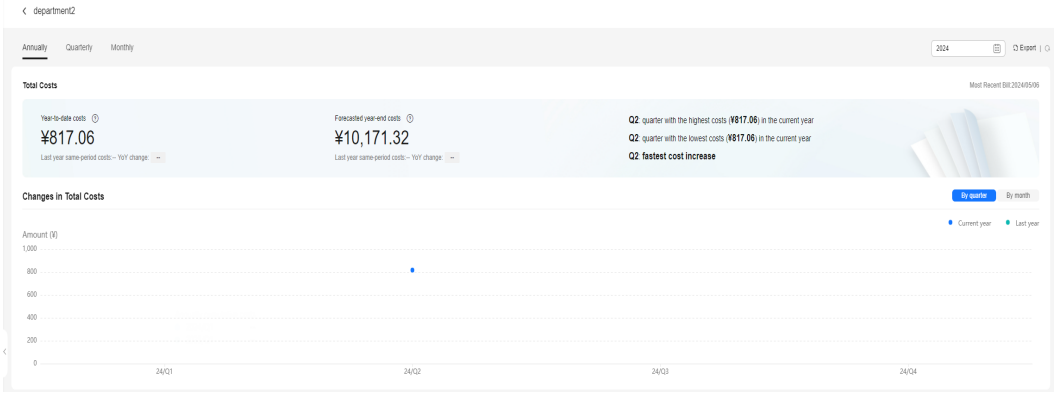
Step 2 View cost analysis reports in **Department Cost Analysis**.

Figure 3-310 Viewing department cost analysis reports



Step 3 In **Department Cost Details**, click the name of a department to view the cost details.

Figure 3-311 View the costs of a department



----End

Table 3-359 Parameters on the cost details page of a department

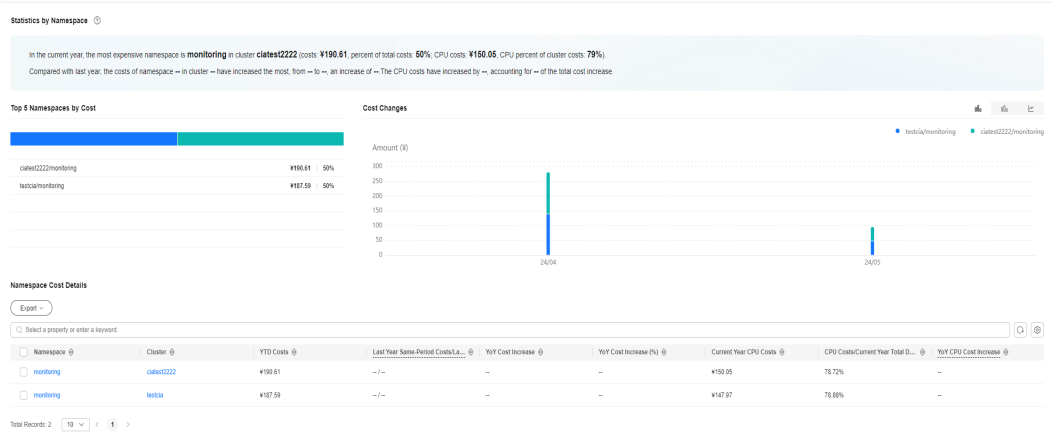
Parameter	Report	Description
Year-to-date costs/Last year same-period costs/YoY change	Annually	**Year-to-date costs**: costs generated by the current department from the start of the current year to the date of the most recent bill **Last year same-period costs**: costs generated by the current department during the same period in the last year **YoY change**: (Year-to-date costs – Last year same-period costs)/Last year same-period costs

Parameter	Report	Description
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
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

Parameter	Report	Description
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
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 3-312 Viewing the cost statistics by namespace



3.13.2.6 Cost Insights for a Cluster

The cost insights for a cluster help cost O&M personnel analyze the cluster cost and resource usage from multiple dimensions, such as namespaces, applications, and node pools, 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.

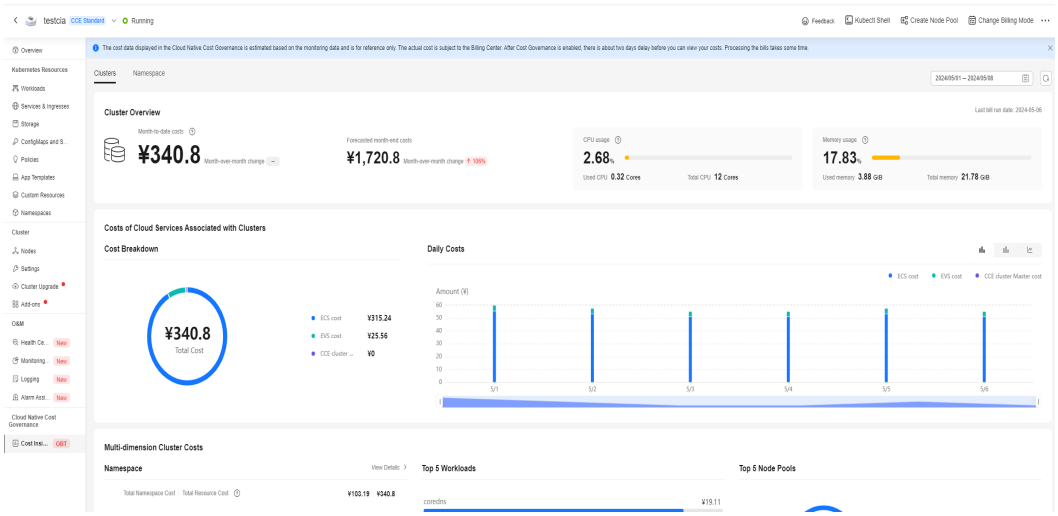
Navigation Path

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 3-313 Cost insights for a cluster



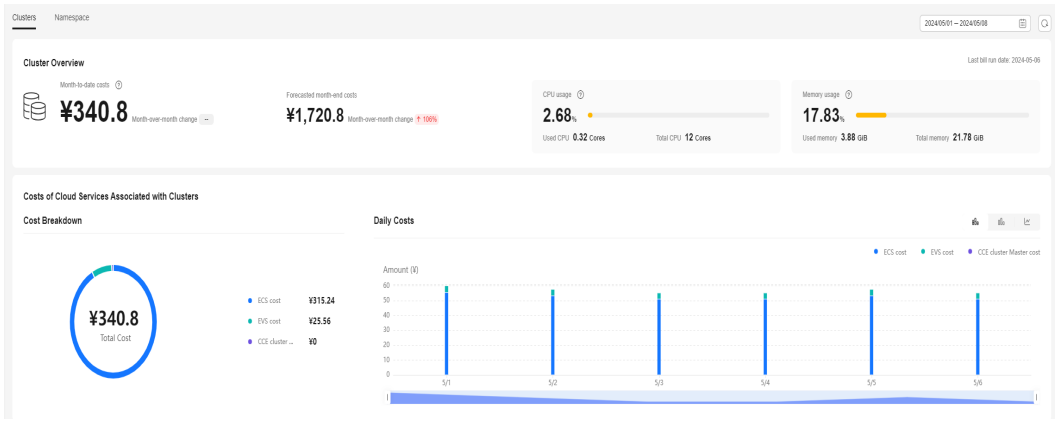
----End

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.

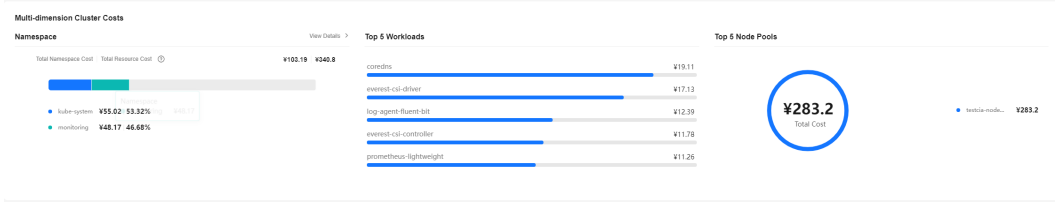
You can filter data by time in the upper right corner.

Figure 3-314 Cost overview of a cluster



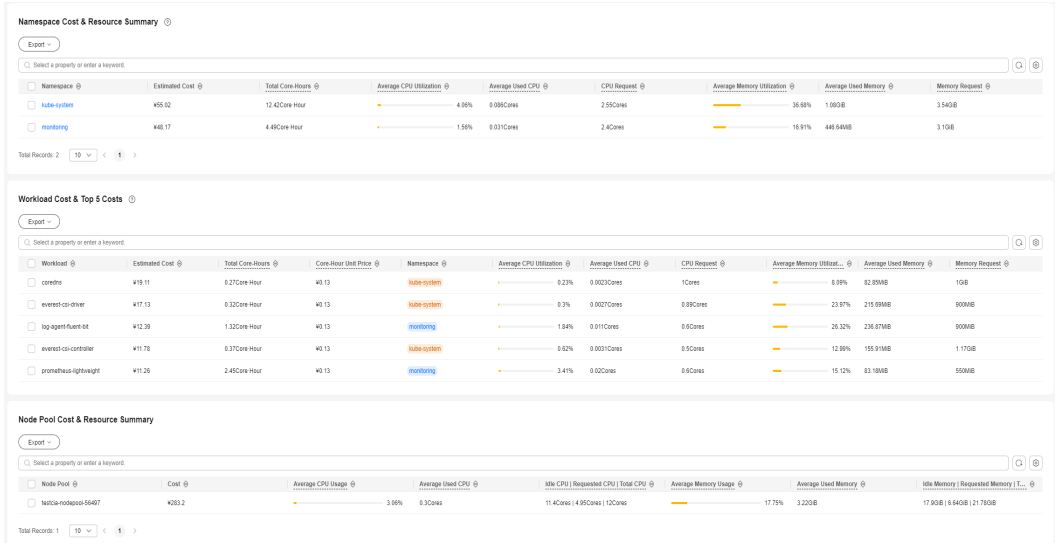
Parameter	Description
Month-to-date costs Month-over-month change	**Month-to-date costs**: costs generated by the cluster from the start of the current month to the date of the most recent bill. If Cost Insights is enabled in the current month, the cost is accumulated from the date when Cost Insights 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 x 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 x 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 3-315 Multi-dimension cluster costs



Function	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 + Uncategorized 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 3-316 Cost & resource summary



Function	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.
	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	The total CPU request of the workloads in a namespace on the date of the most recent bill during the specified time period.
	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 usage of the workloads in a namespace during the specified time period.
	Memory Request	The total memory request of the workloads in a namespace on the date of the most recent bill during the specified time period.
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 the workload belongs to.

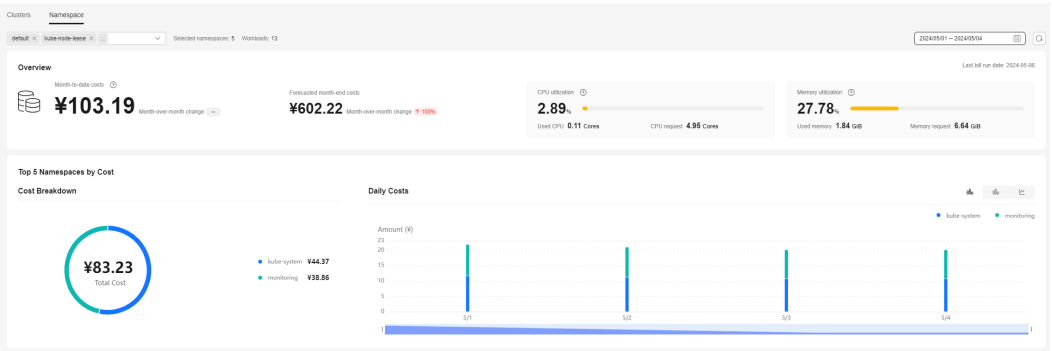
Function	Parameter	Description
	Average CPU Utilization	Average CPU usage of a workload during the specified time period. CPU usage = Used CPU/CPU request x 100%
	Average Used CPU	Average CPU used by a workload during the specified time period.
	CPU Request	CPU request of a workload on the date of the most recent bill during the specified time period.
	Average Memory Utilization	Average memory usage of a workload during the specified time period. Memory usage = Used memory/Memory request x 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 x 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

Function	Parameter	Description
	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.

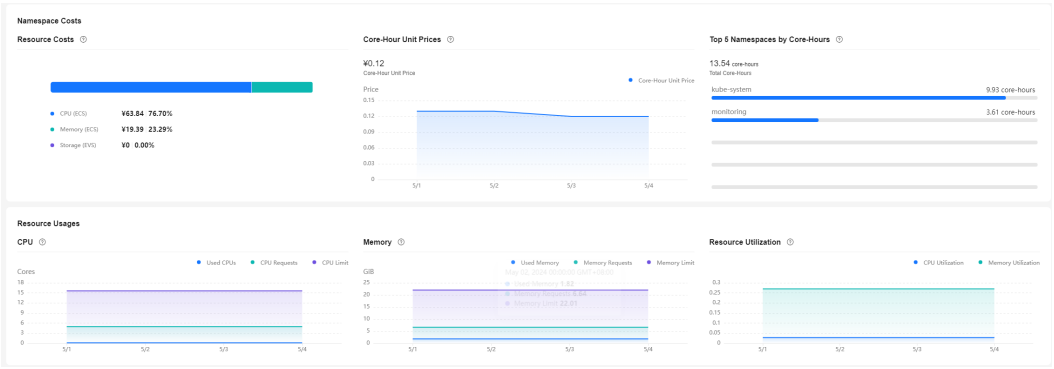
Figure 3-317 Cost overview by namespace



Function	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

Function	Description
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 x 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 x 100%
Cost Breakdown	Cost breakdown of top 5 namespaces among the selected namespaces during the specified time period.
Daily Costs	Daily cost breakdown of the selected namespaces for identifying namespaces with high costs.

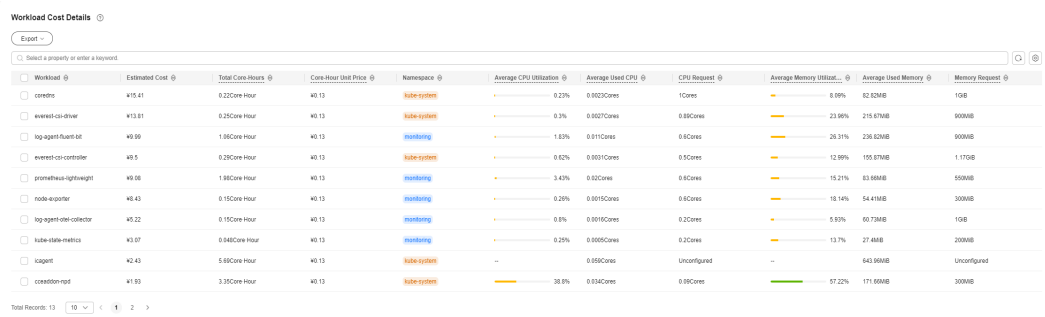
Figure 3-318 Namespace costs



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	Changes of average unit price changes of core-hours consumed by workloads on the nodes in the selected namespaces.
Top 5 Namespaces by Core-Hours	Total core-hours consumed by top 5 namespaces among the selected namespaces during the specified time period.
CPU	Changes of CPU usages, requests, and limits in the selected namespaces.

Parameter	Description
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 3-319 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 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.
Average Memory Utilization	Average memory usage of a workload during the specified time period. Memory usage = Used memory/Memory request x 100%
Average Used Memory	Average memory used by a workload during the specified time period.

Parameter	Description
Memory Request	Memory request of a workload on the last day of the specified time period.

3.14 Add-ons

3.14.1 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.

Scheduling and Elasticity Add-ons

Add-on Name	Description
3.14.13 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.
3.14.6 CCE Cluster Autoscaler	This add-on resizes a cluster based on pod scheduling status and resource usage.
3.14.9 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.
3.14.10 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 CCE clusters, virtual-kubelet uses Cloud Container Instance (CCI) to create pods for Deployments, StatefulSets, and jobs, eliminating the overhead of resizing CCE clusters.

Cloud Native Observability Add-ons

Add-on Name	Description
3.14.17 Cloud Native Cluster Monitoring	This add-on uses Prometheus-operator and Prometheus to provide easy-to-use, end-to-end Kubernetes cluster monitoring. With this add-on, you can access the monitoring center to view monitoring data and configure alarms.
3.14.18 Cloud Native Logging	This add-on collects logs and is built based on open source Fluent Bit and OpenTelemetry. It supports CRD-based log collection policies. It collects and forwards stdout logs, container file logs, node logs, and Kubernetes event logs in a cluster based on configured policies.
3.14.4 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.
3.14.15 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.
3.14.8 Kubernetes Metrics Server	This add-on is an aggregator for monitoring data of core cluster resources.
3.14.20 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.
3.14.23 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
3.14.11 CCE AI Suite (NVIDIA GPU)	NVIDIA GPU is a device management add-on that supports GPUs in containers. It supports only NVIDIA drivers.

Add-on Name	Description
3.14.12 CCE AI Suite (Ascend NPU)	Ascend NPU is a device management add-on that supports Huawei NPUs in containers.

Container Network Add-ons

Add-on Name	Description
3.14.2 CoreDNS	CoreDNS is a DNS server that provides domain name resolution for Kubernetes clusters through a chain add-on.
3.14.7 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.
3.14.16 NodeLocal DNSCache	NodeLocal DNSCache improves cluster DNS performance by running DNS cache proxies as DaemonSets on cluster nodes.

Container Storage Add-on

Add-on Name	Description
3.14.3 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-on

Add-on Name	Description
3.14.14 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.

Other Add-ons

Add-on Name	Description
3.14.5 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 CLI. It allows users to manage applications running in a cluster and troubleshoot faults, as well as manage the cluster itself.
3.14.22 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 3-360 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.
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.

Status	Attribute	Description
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 3-361** on the **Add-ons** page.

Table 3-361 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**.

Operation	Description	Procedure
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.
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

Add-on rollback is supported in certain add-on versions.

- CoreDNS: 1.25.11 and later versions
- Everest: 2.1.19 and later versions
- 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
- kube-prometheus-stack: v3.7.2 and later versions
- Volcano: 1.11.4 and later versions
- NPD: 1.18.22 and later versions

3.14.2 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 **3.7.5 DNS**.

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. Choose **Add-ons** in the navigation pane, locate **CoreDNS** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-362 CoreDNS parameters

Parameter	Description
Pods	Number of pods for the add-on. High availability is not possible with a single add-on pod. If an error occurs on the node where the add-on instance runs, the add-on will fail.
Containers	Queries per second (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. Adjust the number of the CoreDNS pods and their CPU and memory quotas based on the cluster scale. For details, see **Table 3-363**.

Table 3-363 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	2048 MiB

Step 3 Configure the add-on parameters.

Table 3-364 CoreDNS 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

Advance Config	• **parameterSyncStrategy**: indicates whether to configure consistency check when the add-on is upgraded.
	– **ensureConsistent**: indicates that the configuration consistency check is enabled. If the configuration recorded in the cluster is inconsistent with the actual configuration, the add-on cannot be upgraded.
	– **force**: indicates that the configuration consistency check is ignored during an upgrade. In this case, you must ensure that the current effective configuration is the same as the original configuration. After the add-on is upgraded, restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	– **inherit**: indicates that custom settings are automatically inherited during an upgrade. After the add-on is upgraded, restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	• **stub_domains**: sub domains, which allow you to configure a domain name server for a custom domain name. A sub domain is in the format of a key-value pair, where the key is the suffix of a DNS domain name and the value is one or more DNS IP addresses.
	• **upstream_nameservers**: IP address of the upstream DNS server.
	• **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 3-365 describes common plugins. For details, see **Plugins**.
	Example:
	`{`
	`"servers": [`
	`{`
	`"plugins": [`
	`{`
	`"name": "bind",`
	`"parameters": "{$POD_IP}"`
	`},`
	`{`
	`"name": "cache",`
	`"configBlock": "servfail 5s",`

Parameter	Description

	"parameters": 30
	},
	{
	"name": "errors"
	},
	{
	"name": "health",
	"parameters": "{$POD_IP}:8080"
	},
	{
	"name": "ready",
	"{$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": "."
	}
	]
	}
	],
	"stub_domains": {
	"acme.local": [
	"1.2.3.4",
	"6.7.8.9"
	]
	},
	"upstream_nameservers": ["8.8.8.8", "8.8.4.4"]
	}

Table 3-365 Default plugin configuration of the active CoreDNS zone

Plugin Name	Description
bind	Host IP address listened by CoreDNS. Retain the default value **{$POD_IP}**. For details, see **bind**.

Plugin Name	Description
cache	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	Errors are logged to stdout. For details, see **errors**.
health	Health check for CoreDNS. {$POD_IP}:8080 is listened to. Retain the default setting. Otherwise, the CoreDNS health check will fail and the add-on will restart repeatedly. For details, see **health**.
ready	Whether the backend server is ready to receive traffic. {$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	CoreDNS Kubernetes plugin, which provides the service parsing capability in a cluster. For details, see **kubernetes**.
loadbalance	Round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in an answer. For details, see **loadbalance**.
prometheus	API for obtaining CoreDNS metrics. {$POD_IP}:9153 is listened to in the default zone. Retain the default setting. Otherwise, Prometheus cannot collect CoreDNS metrics. For details, see **Prometheus**.
forward	Forwards any queries that are not within the cluster domain of Kubernetes to predefined resolvers (**/etc/resolv.conf**). For details, see **forward**.
reload	Automatically reloads modified Corefiles. After you modify a ConfigMap, wait for two minutes for the modification to take effect. For details, see **reload**.

log	Enables CoreDNS logging. For details, see **log**.
	Example:
	`{`
	`  "name": "log"`
	`}`

template	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 scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-366 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-367 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.

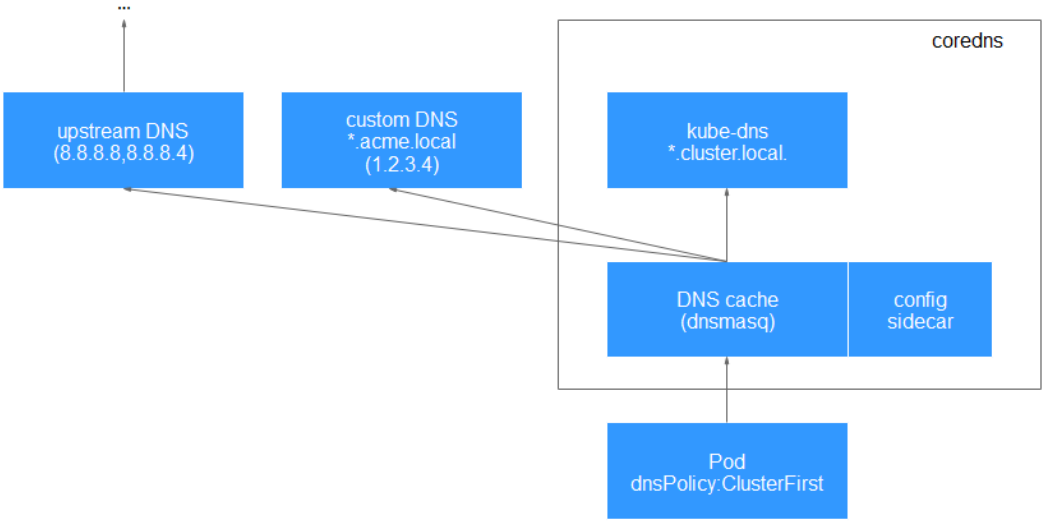
Routing

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 3-320 Routing



Change History

Table 3-368 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
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
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	• Supports association between add-on specifications and cluster specifications. • Synchronizes time zones used by add-ons and nodes.	1.10.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.11	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of 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
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	Updates the add-on to its community version v1.8.4.	1.8.4

Add-on Version	Supported Cluster Version	New Feature	Community Version
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

3.14.3 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.

Constraints

- If your cluster is upgraded from v1.13 to v1.15, **storage-driver** will be replaced by Everest (v1.1.6 or later) for container storage. The takeover does not affect the original storage functions.
- 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.
- By default, this add-on is installed in **clusters of v1.15 and later**. For clusters of v1.13 and earlier, the **storage-driver** add-on is installed by default.
- 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. Click **Add-ons** in the navigation pane, locate **CCE Container Storage (Everest)** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-369 Everest parameters

Parameter	Description
Pods	Number of pods for the add-on. 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	The Everest add-on contains the Everest-csi-controller and everest-csi-driver components. For details, see **Components**. The add-on component specifications can be customized based on your requirements. Retain the default requested CPU and memory values of the add-on components. The limit values can be adjusted based on the number of cluster nodes and PVCs. For details about the configuration suggestions, see **Table 3-370**. In non-typical scenarios, the formulas for estimating the limit values 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 Mi + Number of nodes x 1 Mi + Number of PVCs x 0.2 Mi) 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 Mi for 200 or fewer nodes, 600 Mi for 1000 nodes, and 900 Mi for 2000 nodes

Table 3-370 Recommended configuration limits in typical scenarios

Configuration Scenario			everest-csi-controller		everest-csi-driver	
Nodes	PVs/PVCs	Add-on Instances	vCPUs (Limit = Requested)	Memory (Limit = Requested)	vCPUs (Limit = Requested)	Memory (Limit = Requested)
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
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 3-371 Everest parameters

Parameter	Description
csi_attacher_worker_threads	Number of worker nodes that can be concurrently processed by Everest for attaching EVS volumes. The default value is **60**.
csi_attacher_detach_worker_threads	Number of worker nodes that can be concurrently processed by Everest for detaching EVS volumes. The default value is **60**.
volume_attaching_flow_ctrl	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.
cluster_id	Cluster ID
default_vpc_id	ID of the VPC to which the cluster belongs
disable_auto_mount_secret	Whether the default AK/SK can be used when an object bucket or parallel file system is mounted. The default value is **false**.
enable_node_attacher	Whether to enable the attacher on the agent to process the **VolumeAttachment**.
flow_control	This field is left blank by default. You do not need to configure this parameter.
number_of_reserved_disks	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 has 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.
over_subscription	Overcommitment ratio of the local storage pool (**local_storage**). The default value is **80**. If the size of the local storage pool is 100 GB, it can be overcommitted to 180 GB.
project_id	ID of the project to which a cluster belongs

NOTE

In Everest 1.2.26 or later, the performance of attaching a large number of EVS volumes has been optimized. The following parameters can be configured:

- csi_attacher_worker_threads
- csi_attacher_detach_worker_threads
- volume_attaching_flow_ctrl

The preceding parameters are associated with each other and are constrained by the underlying storage resources in the region where the cluster is located. To attach a large number of volumes (more than 500 EVS volumes per minute), contact customer service and configure the parameters under their guidance to prevent the Everest add-on from running abnormally due to improper parameter settings.

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-372 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-373 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

Component	Description	Resource Type
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 3-374 Key metrics

Metric	Type	Description	Label	Example
everest_action_result_total	Counter	Invoking of different functions	action: indicates different functions. For details, see **Table 3-375**. result: indicates that the invoking is successful or fails.	everest_action_result_total{action="create_snapshot:disk.csi.everest.io",result="success"} 2
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 3-375**.	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 3-375**.	everest_function_duration_seconds_sum{function="create:disk.csi.everest.io"} 24.381399053

Metric	Type	Description	Label	Example
everest_function_duration_seconds_count	Histogram	Number of invoking times of different functions	function: indicates different functions. For details, see **Table 3-375**.	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 3-375 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 3-376 Release history

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 Dedicated Distributed Storage Service (DSS).

Add-on Version	Supported Cluster Version	New Feature
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 pods on nodes in different AZs. • Adapted 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.
2.1.9	v1.19 v1.21 v1.23 v1.25	• Supported graceful exit of the controller. • CCE clusters 1.25 are supported.

Add-on Version	Supported Cluster Version	New Feature
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 pods on nodes in different AZs.
1.2.70	v1.15 v1.17 v1.19 v1.21	Optimized the performance of creating subpath PVCs in batches for SFS Turbo volumes.

Add-on Version	Supported Cluster Version	New Feature
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.
1.2.30	v1.15 v1.17 v1.19 v1.21	Supports emptyDir.

Add-on Version	Supported Cluster Version	New Feature
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	Supports ultra-fast SSD (ESSD) and general-purpose SSD (GPSSD) EVS disks.
1.2.13	v1.15 v1.17 v1.19	Supports EulerOS 2.10.
1.2.9	v1.15 v1.17 v1.19	• Enhances the reliability of PV resource lifecycle maintenance. • Attach/Detach Controller can be used to attach or detach volumes in clusters 1.19.10. • Improves SFS mounting stability. • Changes 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. • Optimizes the authentication function of using OBS, which requires you to upload the access key. • Improves the compatibility of the everest add-on with FlexVolume volumes. • Improves running stability of the add-on.
1.1.12	v1.15 v1.17	Enhances the reliability of the everest-csi-controller component.

Add-on Version	Supported Cluster Version	New Feature
1.1.11	v1.15 v1.17	• Supports security hardening. • Supports third-party OBS storage. • Switches to the EVS query API with better performance. • Uses snapshots to create disks in clone mode by default. • Optimizes and enhances disk status detection and log output for attaching and detaching operations. • Improves the reliability of determining authentication expiration.
1.1.8	v1.15 v1.17	Supports CCE v1.17. If CCE v1.13 is upgraded to v1.15, everest can take over all functions of the FlexVolume driver.
1.1.7	v1.15 v1.17	Supports CCE v1.17. If CCE v1.13 is upgraded to v1.15, everest can take over all functions of the FlexVolume driver.

3.14.4 CCE Node Problem Detector

Introduction

CCE node problem detector (NPD) is an add-on that 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. The NPD add-on can run as a DaemonSet or a daemon.

For more information, see **node-problem-detector**.

Constraints

- When using this add-on, do not format or partition node disks.
- Each NPD process occupies 30 m CPU and 100 MB 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. Choose **Add-ons** in the navigation pane, locate **CCE Node Problem Detector** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-377 NPD configuration

Parameter	Description
Add-on Specifications	The specifications can be **Custom**.
Pods	If you select **Custom**, you can adjust the number of pods as required.
Containers	If you select **Custom**, you can adjust the container specifications as required.

Step 3 Configure the add-on parameters.

Only v1.16.0 and later versions support the configurations.

Table 3-378 NPD parameters

Parameter	Description
common.image.pullPolicy	An image pulling policy. The default value is **IfNotPresent**.
feature_gates	A feature gate
npc.maxTaintedNode	The maximum number of nodes that NPC can add taints to when a single fault occurs on multiple nodes for minimizing impact. The value can be in int or percentage format.
npc.nodeAffinity	Node affinity of the controller

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-379 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-380 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 3-381 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\\."

Check Item	Function	Description
Readonly Filesyste m	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 3-382 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

Check Item	Function	Description
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 3-383 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.

Check Item	Function	Description
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 3-384 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 a temporary 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"`
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` • Threshold: – Average usage: ioutil >= 0.99 – Average I/O queue length: avgqu-sz >= 1 – Average I/O transfer volume: iops (w/s) + ioth (wMB/s) <= 1 **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 3-385 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 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 3-386 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.

Check Item	Function	Description
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 3-387 Parameters

Parameter	Description	Default
npc.enable	Whether to enable NPC This parameter is not supported in 1.18.0 or later versions.	true

Parameter	Description	Default
npc.maxTaintedNode	The maximum number of nodes that NPC can add taints to when a single 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

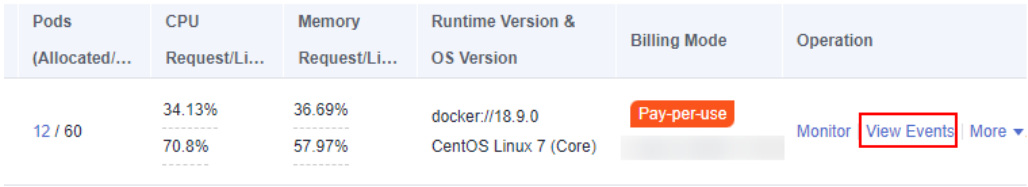
Events reported by the NPD add-on can be queried on the **Nodes** page.

Step 1 Log in to the CCE console.

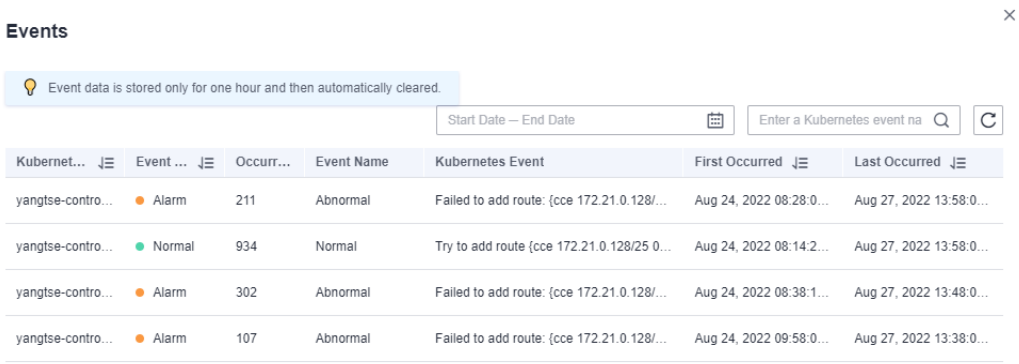
Step 2 Click the cluster name to access the cluster console. Choose **Nodes** in the navigation pane.

Step 3 Locate the row that contains the target node, and click **View Events**.

Figure 3-321 Viewing node events



You can obtain events on the page displayed.



----End

AOM Alarms

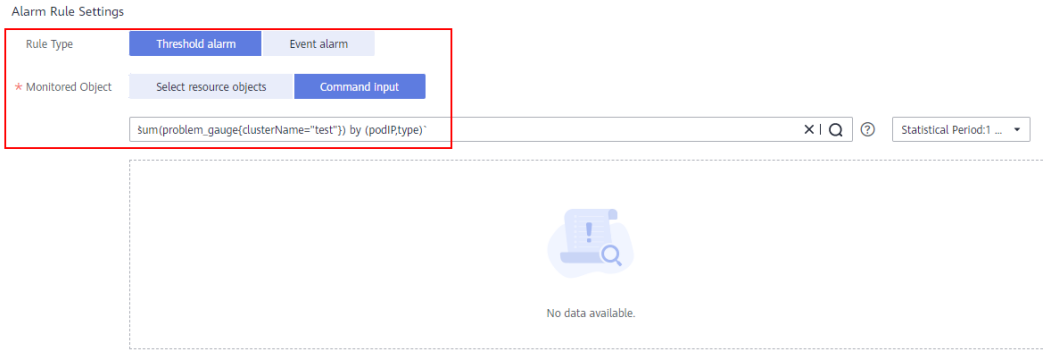
For status-related NPD check items, you can configure Application Operations Management (AOM) to convert abnormal statuses to AOM alarms and notify you via SMS message or email.

Step 1 Log in to the AOM console.

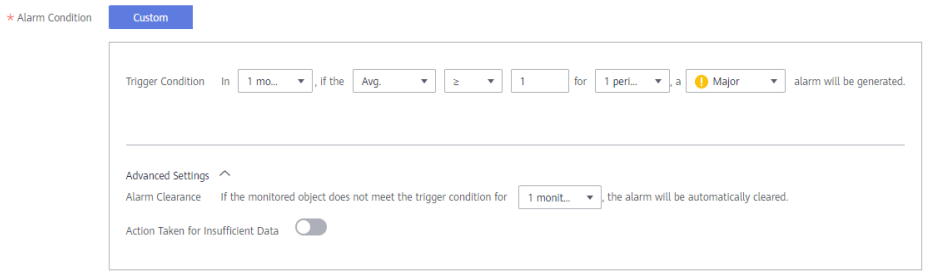
Step 2 In the navigation pane, choose **Alarm Center** > **Alarm Rules** and click **Create Alarm Rule**.

Step 3 Set an alarm rule.

- **Rule Type**: Select **Threshold alarm**.
- **Monitored Object**: Select **Command input**.
- Enter ` **sum(problem_gauge{clusterName="test"}) by (podIP,type)**` in the text box.



- **Alarm Condition**: Trigger a major alarm if the average value is greater than or equal to 1 for one consecutive time in a monitoring period.



- (Optional) **Alarm notification**: To get notified of alarms by email or SMS message, configure action rules for the alarm rule. If no action rule is available, you can create one.

----End

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 3-388 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
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 pods on nodes in different AZs. • Allows adding a taint to a node before the release of a spot ECS for the node to repel a set of pods. • Synchronizes time zones used by add-ons and nodes.	0.8.10
1.18.10	v1.19 v1.21 v1.23 v1.25	• Optimizes the configuration page. • Adds threshold configuration to the DiskSlow check item. • Adds threshold configuration to the NTPProblem check item. • Supported anti-affinity scheduling of pods on nodes in different AZs. • Supports interruption detection for spot ECSs and evicts pods on nodes before the interruption.	0.8.10
1.17.4	v1.17 v1.19 v1.21 v1.23 v1.25	Optimizes DiskHung check item.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
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. • Adds the ProcessZ check item. • Adds the time deviation detection to the NTPProblem check item. • Fixes 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	• Adds the DiskHung check item for disk I/O. • Adds the DiskSlow check item for disk I/O. • Adds the ProcessD check item. • Adds 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**. • Supports clusters 1.25.	0.8.10
1.16.4	v1.17 v1.19 v1.21 v1.23	• Adds 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	Adds the function of checking the ResolvConf configuration file.	0.8.10

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.16.1	v1.17 v1.19 v1.21 v1.23	• Adds node-problem-controller. Supports basic fault isolation. • Adds 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	• Hardens check items comprehensively to avoid false positives. • Supports kernel check. Supports 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	Fixes the issue that monitoring metrics cannot be obtained.	0.7.1
1.14.4	v1.17 v1.19	• Supports 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
1.13.8	v1.15.11 v1.17	• Fixes the CNI health check issue on the container tunnel network. • Adjusts resource quotas.	0.7.1
1.13.6	v1.15.11 v1.17	Fixes the issue that zombie processes are not reclaimed.	0.7.1
1.13.5	v1.15.11 v1.17	Adds taint tolerance configuration.	0.7.1
1.13.2	v1.15.11 v1.17	Adds resource restrictions and enhances the detection capability of the cni add-on.	0.7.1

3.14.5 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. Choose **Add-ons** in the navigation pane, locate **Kubernetes Dashboard** on the right, and click **Install**.

Step 2 In the **Parameters** area, configure the following parameters:

- **Certificate Configuration**: Configure a certificate for the dashboard.
  - Using a custom certification
    - **Certificate File**: Click to view the example certificate file.
    - **Private Key**: Click to view the example private key.
  - Using a 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 3 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**.

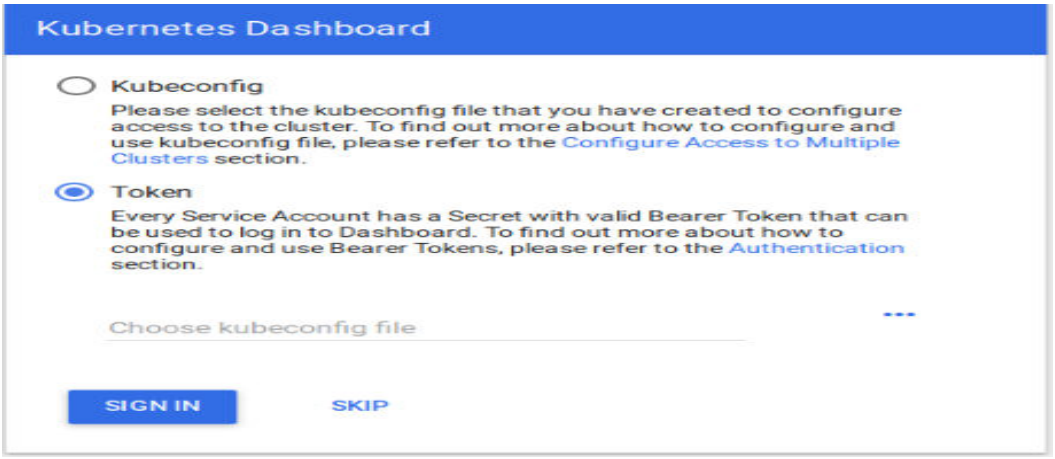
Step 2 Copy the token in the dialog box displayed.

Step 3 On the dashboard login page, select **Token**, paste the copied token, and click **SIGN IN**.

NOTE

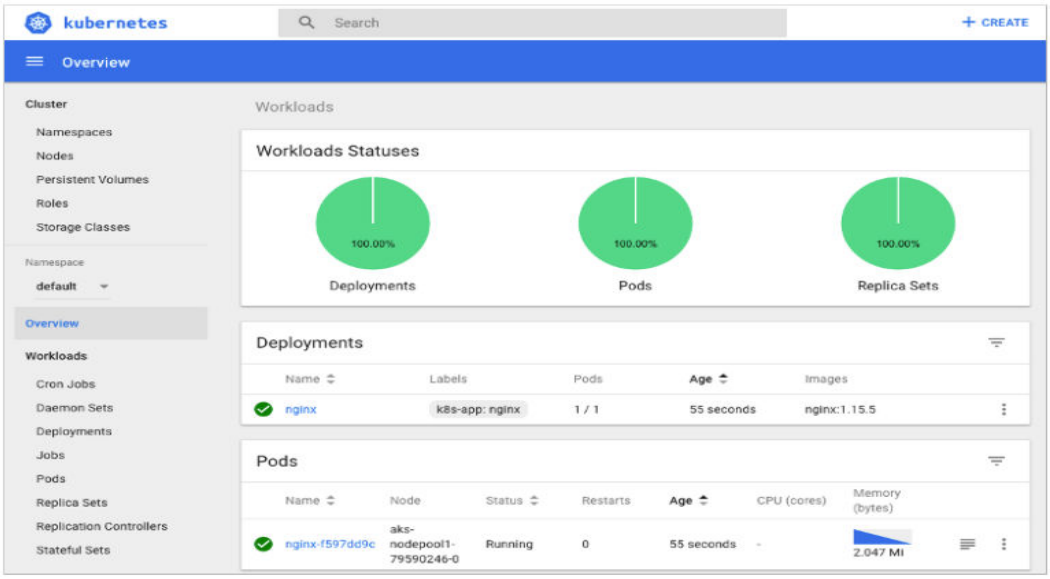
By default, this add-on does not support login using kubeconfig authenticated by certificate. You are advised to use the token mode for login. For details, see **https://github.com/kubernetes/dashboard/issues/2474#issuecomment-348912376**.

Figure 3-322 Token login



Step 4 View the dashboard page as shown in **Figure 3-323**.

Figure 3-323 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

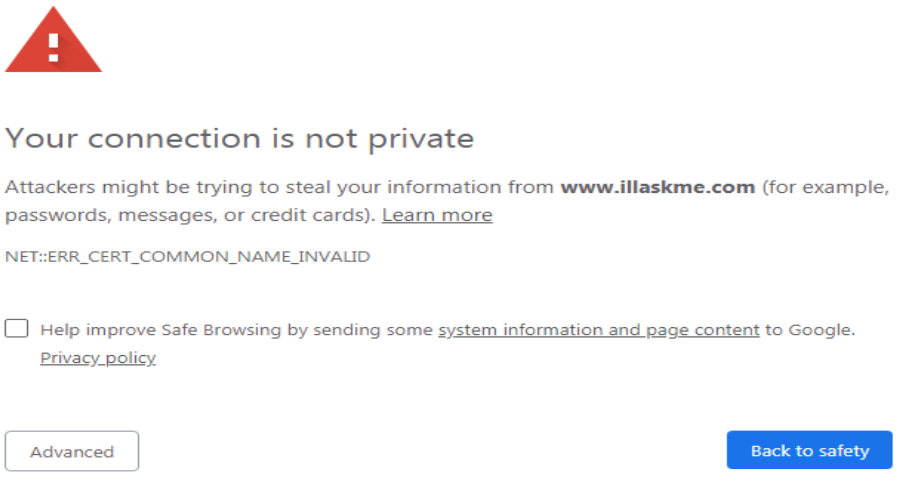
Table 3-389 Add-on components

Component	Description	Resource Type
Dashboard	Visualized monitoring UI	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 3-324 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 3-390 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
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	Synchronizes time zones used by add-ons and nodes.	**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	Adds 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**

Add-on Version	Supported Cluster Version	New Feature	Community Version
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
2.0.0	v1.17	Enables interconnection with CCE v1.17	2.0.0

3.14.6 CCE Cluster Autoscaler

Introduction

Autoscaler is an important Kubernetes controller. It supports microservice scaling and is key to serverless design.

When the CPU or memory usage of a microservice is too high, horizontal pod autoscaling is triggered to add pods to reduce the load. These pods can be automatically reduced when the load is low, allowing the microservice to run as efficiently as possible.

CCE simplifies the creation, upgrade, and manual scaling of Kubernetes clusters, in which traffic loads change over time. To balance resource usage and workload performance of nodes, Kubernetes introduces the Autoscaler add-on to automatically adjust the number of nodes a cluster based on the resource usage required for workloads deployed in the cluster. For details, see **3.12.3.2 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 pods in a cluster cannot be scheduled due to insufficient worker nodes, cluster scaling is triggered to add nodes. The nodes to be added have the same specification as configured for the node pool to which the nodes belong.

    Auto scale-out will be performed when:

    ▪ Node resources are insufficient.

    ▪ No node affinity policy is set in the pod scheduling configuration. If a node has been configured as an affinity node for pods, no node will not be automatically added when pods cannot be scheduled. For details about how to configure the node affinity policy, see **3.5.3.11 Scheduling Policies (Affinity/Anti-affinity)**.

  - When the cluster meets the node scaling policy, cluster scale-out is also triggered. For details, see **3.12.3.2 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.

Constraints

- Ensure that there are sufficient resources for installing the add-on.
- Only **pay-per-use VM nodes** can be added or removed by Autoscaler.
- The default node pool does not support auto scaling. For details, see **Description of DefaultPool**.
- When 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 affects 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. Choose **Add-ons** in the navigation pane, locate **CCE Cluster Autoscaler** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-391 Specifications configuration

Parameter	Description
Pods	Number of pods for the add-on. 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	Adjust the number of the Autoscaler pods and their CPU and memory quotas based on the cluster scale. For details, see **Table 3-392**.

Table 3-392 Recommended Autoscaler quotas

Nodes	Pods	Requested vCPUs	vCPU Limit	Requested Memory	Memory Limit
50	2	1000m	1000m	1000 MiB	1000 MiB
200	2	4000m	4000m	2000 MiB	2000 MiB
1000	2	8000m	8000m	8000 MiB	8000 MiB
2000	2	8000m	8000m	8000 MiB	8000 MiB

Step 3 Configure the add-on parameters.

Table 3-393 Parameters

Parameter	Description
Total Nodes	Maximum number of nodes that can be managed by the cluster, within which cluster scale-out is performed.
Total CPUs	Maximum sum of CPU cores of all nodes in a cluster, within which cluster scale-out is performed.
Total Memory	Maximum sum of memory of all nodes in a cluster, within which cluster scale-out is performed.

NOTE

When the total number of nodes, CPUs, or memory is counted, unavailable nodes and resources on them in the default node pool are not included.

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-394 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 After the configuration is complete, click **Install**.

----End

Components

Table 3-395 Add-on components

Component	Description	Resource Type
Autoscaler	Auto scaling for Kubernetes clusters	Deployment

Scale-In Cool-Down Period

Scale-in cooling intervals can be configured in the node pool settings and the Autoscaler add-on settings.

Scale-in cooling interval configured in a node pool

This interval indicates the period during which nodes added to the current node pool after a scale-out operation cannot be deleted. This interval takes effect at the node pool level.

Scale-in cooling interval configured in the Autoscaler add-on

The interval after a scale-out indicates the period during which the entire cluster cannot be scaled in after the Autoscaler add-on triggers scale-out (due to the unschedulable pods, metrics, and scaling policies). This interval takes effect at the cluster level.

The interval after a node is deleted indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers scale-in. This interval takes effect at the cluster level.

The interval after a failed scale-in indicates the period during which the cluster cannot be scaled in after the Autoscaler add-on triggers scale-in. This interval takes effect at the cluster level.

Change History

Table 3-396 Updates of the add-on adapted to clusters 1.28

Add-on Version	Supported Cluster Version	New Feature	Community Version
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 3-397 Updates of the add-on adapted to clusters 1.27

Add-on Version	Supported Cluster Version	New Feature	Community Version
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 3-398 Updates of the add-on adapted to clusters 1.25

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.25.116	v1.25	Optimized the logic for generating alarms when resources in a bode pool are sold out.	1.25.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
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
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 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 AS 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 3-399 Updates of the add-on adapted to clusters 1.23

Add-on Version	Supported Cluster Version	New Feature	Community Version
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
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

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
1.23.21	v1.23	• Supported anti-affinity scheduling of 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 AS 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 3-400 Updates of 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 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 AS 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 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 AS 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 3-401 Updates of 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 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 AS 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 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 AS 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 3-402 Updates of 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

3.14.7 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 nginx-ingress Works**.

You can visit the **open source community** for more information.

NOTE

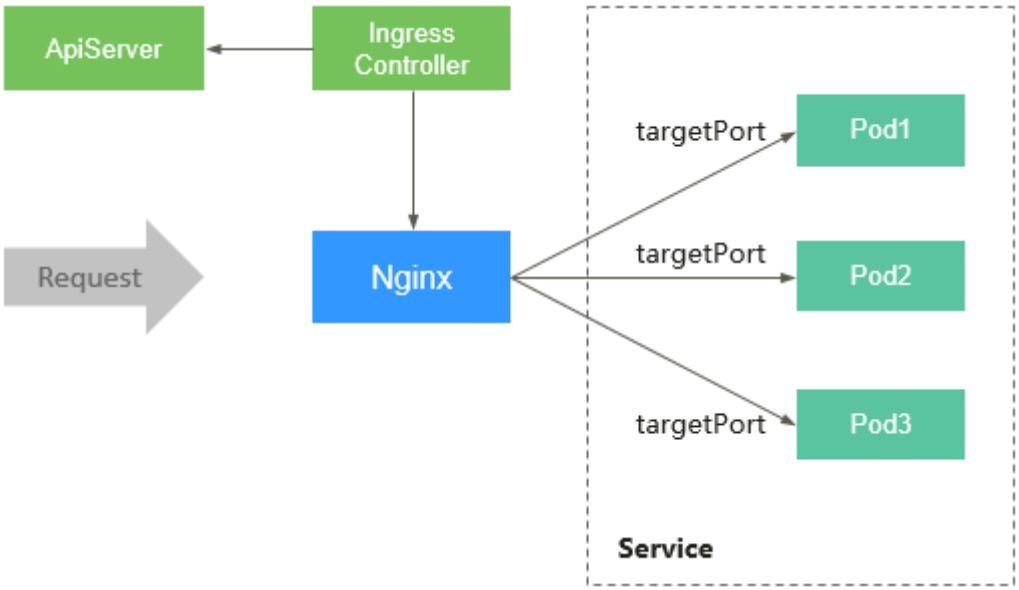
- When installing the NGINX Ingress Controller, you can specify Nginx parameters. These parameters take effect globally and are contained in the **nginx.conf** file. You can search for the parameters in **ConfigMaps**. If the parameters are not included in ConfigMaps, the configurations will not take 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 nginx-ingress Works

nginx-ingress 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 3-325** shows how nginx-ingress 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 3-325 Working principles of nginx-ingress



Constraints

- This add-on can be installed only in clusters of v1.15 or later.
- For clusters earlier than v1.23, **kubernetes.io/ingress.class: "nginx"** must be added to the annotation of the Ingress created through the API.
- Dedicated load balancers must be the network type (TCP/UDP) supporting private networks (with a private IP).
- The node where nginx-ingress-controller is running and the containers running on the node cannot access Nginx Ingress. In this case, perform anti-affinity deployment for the workloads and nginx-ingress-controller. For details, see **Anti-affinity Deployment for Workloads and nginx-ingress-controller**.

- During the nginx-ingress upgrade, 10s is reserved for deleting the nginx-ingress controller at the ELB backend.
- The timeout interval for the graceful exit of nginx-ingress-controller is 300s. If the timeout is more than 300s during the nginx-ingress upgrade, the persistent connection will be disconnected and services will be interrupted for a short period of time.

Prerequisites

Before creating a workload, you must have an available cluster. If no cluster is available, create one according to **3.2.2.1 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. Choose **Add-ons** in the navigation pane, locate **NGINX Ingress Controller** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-403 nginx-ingress configuration

Parameter	Description
Add-on Specifications	Nginx Ingress can be deployed based on customized resource specifications.
Pods	You can adjust the number of add-on instances as required.
Containers	You can adjust the container specifications of an add-on instance as required.

Step 3 Configure the add-on parameters.

- **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.
- **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**: Configuring the **nginx.conf** file will affect all managed ingresses. You can search for related parameters through **ConfigMaps**. If the parameters you configured are not included in the options listed in the **ConfigMaps**, the parameters will not take effect.

  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"
}
```

- **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.

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-404 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-405 Add-on components

Component	Description	Resource Type
cceaddon-nginx-ingress-controller	Nginx Ingress controller, which provides flexible routing and forwarding for clusters	Deployment
cceaddon-nginx-ingress-default-backend	Default backend of Nginx Ingress. The message "default backend - 404" is returned.	Deployment

Anti-affinity Deployment for Workloads and nginx-ingress-controller

The node where nginx-ingress-controller is running and the containers running on the node cannot access Nginx Ingress. To prevent this problem, configure an anti-affinity rule to tell the scheduler not to co-locate the workload and nginx-ingress-controller on the same node.

```
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:     # Use the labels of nginx-ingress-controller to implement anti-affinity.
                - key: app
                  operator: In
                  values:
                    - nginx-ingress
                - key: component
                  operator: In
                  values:
                    - controller
            namespaces:
              - kube-system
            topologyKey: kubernetes.io/hostname
```

Change History

Table 3-406 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.6.4	v1.25 v1.27 v1.28 v1.29	CCE clusters 1.29 are supported.	1.9.3
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

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
2.3.5	v1.27	None	1.8.0
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
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	• Synchronizes time zones used by add-ons and nodes. • Supports IPv4 and IPv6 dual stack.	1.5.1

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.2.3	v1.25	• Supported anti-affinity scheduling of 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
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 pods on nodes in different AZs. • The default taint tolerance duration is changed to 60s. • Synchronizes time zones used by add-ons and nodes. • Supports IPv4 and IPv6 dual stack.	1.2.1
2.1.5	v1.19 v1.21 v1.23	• Supported anti-affinity scheduling of 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

Add-on Version	Supported Cluster Version	New Feature	Community Version
2.1.1	v1.19 v1.21 v1.23	Updated the add-on to its community version v1.2.1.	1.2.1
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
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
1.3.2	v1.15 v1.17 v1.19 v1.21	• CCE clusters 1.21 are supported. • Updates the add-on to its community version v0.49.3.	0.49.3
1.2.6	v1.15 v1.17 v1.19	Adds the default seccomp profile.	0.46.0
1.2.5	v1.15 v1.17 v1.19	Updates 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

Add-on Version	Supported Cluster Version	New Feature	Community Version
1.2.2	v1.15 v1.17	Updates the add-on to its community version v0.43.0.	0.43.0

3.14.8 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 **3.12.2.2 HPA Policies**.

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. Choose **Add-ons** in the navigation pane, locate **Kubernetes Metrics Server** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-407 metrics-server configuration

Parameter	Description
Add-on Specifications	Select **Single**, **Custom**, or **HA** for **Add-on Specifications**.
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select **Custom**, you can adjust the container specifications as required.

Step 3 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-408 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 4 Click **Install**.

----End

Components

Table 3-409 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 3-410 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
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	Synchronizes time zones used by add-ons and nodes.	0.6.2
1.3.6	v1.19 v1.21 v1.23 v1.25	• Supported anti-affinity scheduling of 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 Deployments 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	Updates 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

3.14.9 CCE Advanced HPA

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 CustomedHPA policies. For details, see **3.12.2.4 CustomedHPA Policies**.

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.

Constraints

- This add-on can be installed only in clusters of v1.15 or later.
- If the cce-hpa-controller version is earlier than 1.2.11, the **Prometheus** add-on must be installed. If the cce-hpa-controller version is 1.2.11 or later, the add-ons that can provide metrics API must be installed. Select one of the following add-ons based on your cluster version and actual requirements.
  - **3.14.8 Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.
  - **3.14.17 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 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**.
  - **3.14.23 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. Click **Add-ons** in the navigation pane, locate **CCE Advanced HPA** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-411 cce-hpa-controller configuration

Parameter	Description
Add-on Specifications	Select **Single** or **Custom** for **Add-on Specifications**. **NOTE** Single-instance add-ons are used only for service verification. In commercial deployments, select **Custom** based on the cluster specifications. The specifications of cce-hpa-controller are decided by the total number of containers in the cluster and the number of scaling policies. You are advised to configure 500m CPU and 1,000 MiB memory for every 5,000 containers, and 100m CPU and 500 MiB memory for every 1,000 scaling policies.
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.

Parameter	Description
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select **Custom**, you can adjust the container specifications as required.

Step 3 Select **Single** or **Custom** for **Add-on Specifications**.

- **Pods**: Set the number of pods based on service requirements.
- **Containers**: Set a proper container quota based on service requirements.

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-412 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-413 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 3-414 Release history

Add-on Version	Supported Cluster Version	New Feature
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.
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 pods on nodes in different AZs.

Add-on Version	Supported Cluster Version	New Feature
1.3.3	v1.19 v1.21 v1.23 v1.25	• CCE clusters 1.25 are supported. • Allowed CronHPA to adjust the number of Deployments 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.
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	Supports ARM64 nodes.
1.2.2	v1.15 v1.17 v1.19	Enhances the health check function.
1.2.1	v1.15 v1.17 v1.19	• CCE clusters 1.19 are supported. • Updates the add-on to a stable version.

Add-on Version	Supported Cluster Version	New Feature
1.1.3	v1.15 v1.17	Supports periodic scaling rules.

3.14.10 CCE Cloud Bursting Engine for CCI

CCE Cloud Bursting Engine for CCI 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 scale Deployments, StatefulSets, Jobs, and CronJobs running in CCE clusters to **Cloud Container Instance (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 on the left, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Install**.
5. Configure the add-on parameters.

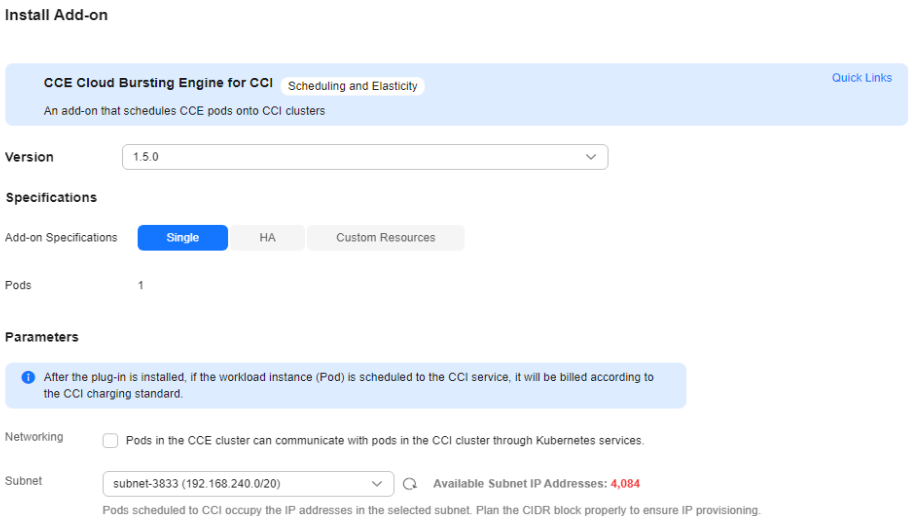
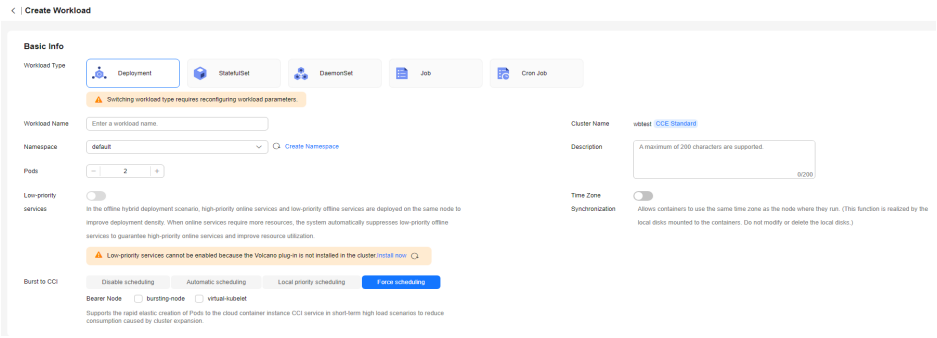


Table 3-415 Add-on parameters

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 a workload.
Networking	If this option is enabled, pods in a CCE cluster can communicate with the pods in CCI. 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 on the left, 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 **Scaling Pods to CCI**.



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 on the left, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

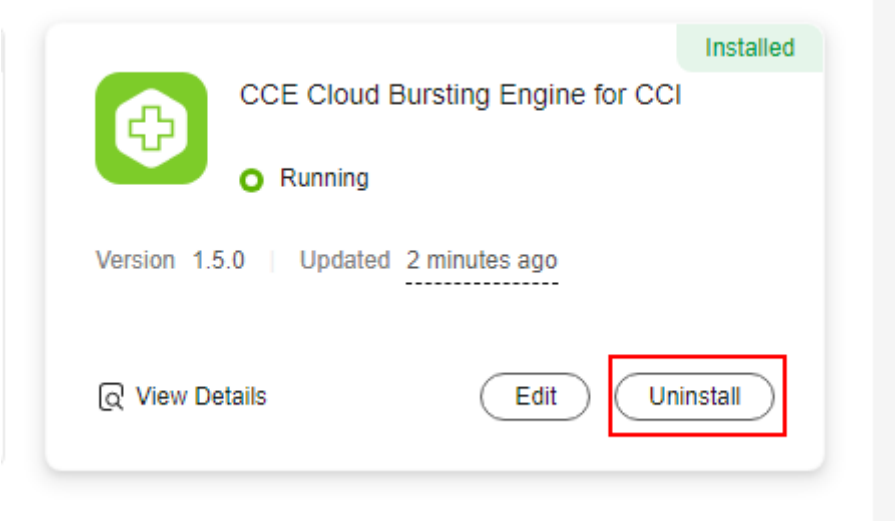


Table 3-416 Special scenarios for uninstalling the add-on

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 3-417 Release history

Add-on Version	Supported Cluster Version	New Feature
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.
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	Supports clusters of v1.21 and v1.23.
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.

Add-on Version	Supported Cluster Version	New Feature
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.
1.2.0	v1.13 v1.15 v1.17 v1.19	• Supported clusters 1.19. • 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.

3.14.11 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.

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. Choose **Add-ons** in the navigation pane, locate **CCE AI Suite (NVIDIA GPU)** on the right, and click **Install**.

Step 2 Configure the add-on parameters.

- **NVIDIA Driver**: Enter the link for downloading the NVIDIA driver. All GPU nodes in the cluster will use this 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.
  - After the driver version is changed, restart the node for the change to take effect.
  - 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.

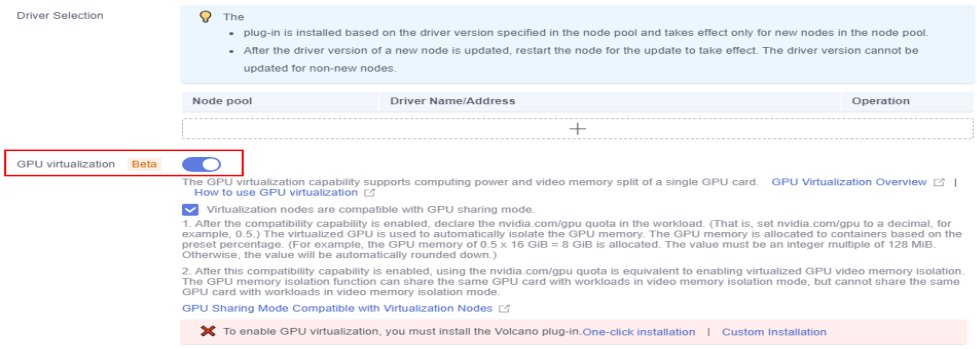
- **Driver Selection**: If you do not want all GPU nodes in a cluster to use the same driver, CCE allows you to install a different GPU driver for each node pool.

NOTE

- The add-on installs the driver with the version specified by the node pool. The driver takes effect only for new pool nodes.
- 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.

- **GPU virtualization** (supported in 2.0.5 and later versions): Enable GPU virtualization to support the segmentation and isolation for the compute power and GPU memory of a single GPU.

Figure 3-326 Enabling GPU Virtualization



If the Volcano add-on has not been installed in the cluster, GPU virtualization cannot be enabled. Click **One-click installation** to install it. To configure the Volcano add-on parameters during installation, click **Custom Installation**. For details, see **3.14.13 Volcano Scheduler**.

If the Volcano add-on has been installed in the cluster but its version does not support GPU virtualization, click **Upgrade** to upgrade it. To configure the Volcano add-on parameters during installation, click **Custom Upgrade**. For details, see **3.14.13 Volcano Scheduler**.

NOTE

After GPU virtualization is enabled, select **Virtualization nodes are compatible with GPU sharing mode**, that is, **default GPU scheduling in Kubernetes** is supported. This capability requires that the version of gpu-device-plugin is 2.0.10 or later and the version of Volcano is 1.10.5 or later.

- If you enable compatibility, the **nvidia.com/gpu** quota specified in workloads (the **nvidia.com/gpu** quota is set to a decimal fraction, for example, 0.5) is provided by GPU virtualization to implement GPU memory isolation. The GPU memory is allocated to containers based on the specified quota. For example, 8 GiB (0.5 x 16 GiB) GPU memory is allocated. The value of GPU memory must be an integer multiple of 128 MiB. Otherwise, the value is automatically rounded down to the nearest integer. If **nvidia.com/gpu** resources have been used in the workload before compatibility is enabled, the resources will not be provided by GPU virtualization but the entire GPU.
- After compatibility is enabled, if you use the **nvidia.com/gpu** quota, it is equivalent to enabling GPU memory isolation. The **nvidia.com/gpu** quota can share a GPU with workloads in GPU memory isolation mode, but cannot share a GPU with workloads in compute and GPU memory isolation mode. In addition, **Constraints** on GPU virtualization must be followed.
- If compatibility is disabled, the **nvidia.com/gpu** quota specified in the workload only affects the scheduling result. It does not require GPU memory isolation. That is, although the **nvidia.com/gpu** quota is set to 0.5, you can still view complete GPU memory in the container. In addition, workloads using **nvidia.com/gpu** resources and workloads using virtualized GPU memory cannot be scheduled to the same node.
- If you deselect **Virtualization nodes are compatible with GPU sharing mode**, running workloads will not be affected, but workloads may fail to be scheduled. For example, if compatibility is disabled, the workload using **nvidia.com/gpu** resources are still in the GPU memory isolation mode. As a result, the GPU cannot schedule workloads in compute and GPU memory isolation mode. You need to delete workloads using **nvidia.com/gpu** resources before rescheduling.

Step 3 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.


```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.118.02   Driver Version: 440.118.02   CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   31C    P0    23W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

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 3-418 Supported GPU Drivers

GPU Model	Supported Cluster Type	Specification	OS						
			Huawei Cloud EulerOS 2.0 (GPU virtualization supported)	Ubuntu 22.04	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	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	535.54.03 470.141.03	470.141.03	470.141.03

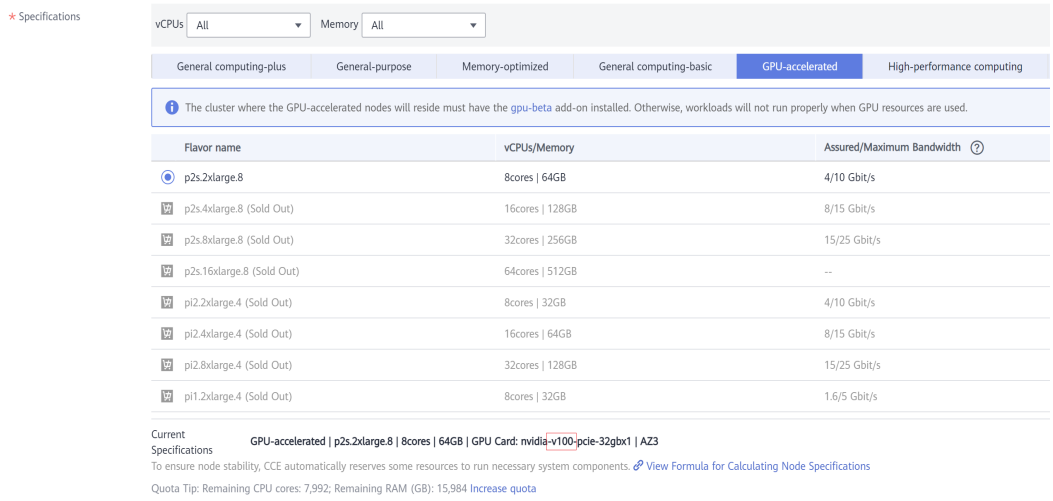
GPU Model	Supported Cluster Type	Specification	OS						
			Huawei Cloud EulerOS 2.0 (GPU virtualization supported)	Ubuntu 22.04	CentOS Linux release 7.6	EulerOS release 2.9	EulerOS release 2.5	Ubuntu 18.04 (end of maintenance)	EulerOS release 2.3 (end of maintenance)
Volta V100	CCE standard cluster	p2s p2vs p2v	535.54.03 510.47.03 470.57.02	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 Click **Create Node** and select the GPU node to be created in the **Specifications** area. The GPU card model of the node is displayed in the lower part of the page.

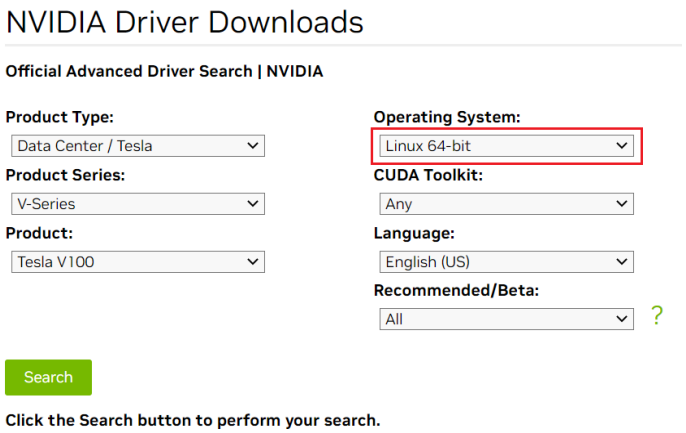
Figure 3-327 Viewing the GPU card model



Step 3 Visit *https://www.nvidia.com/Download/Find.aspx?lang=en*.

Step 4 Select the driver information on the **NVIDIA Driver Downloads** page, as shown in **Figure 3-328**. **Operating System** must be **Linux 64-bit**.

Figure 3-328 Setting parameters



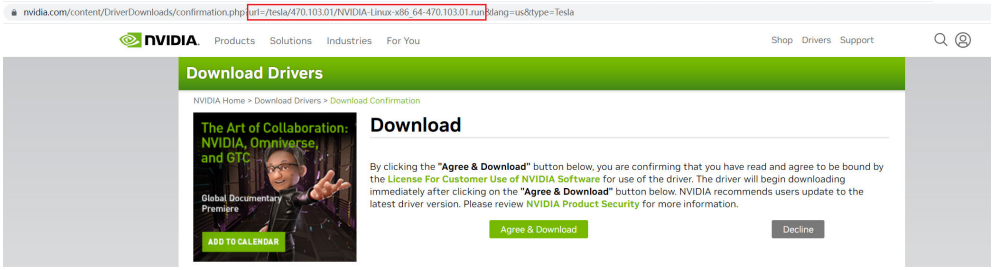
Step 5 After confirming the driver information, click **SEARCH**. A page is displayed, showing the driver information, as shown in **Figure 3-329**. Click **DOWNLOAD**.

Figure 3-329 Driver information

Step 6 Obtain the driver link in either of the following ways:

- Method 1: As shown in **Figure 3-330**, find *url=/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run* in the browser address box. Then, supplement it to obtain the driver link **https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run**. By using this method, you must bind an EIP to each GPU node.
- Method 2: As shown in **Figure 3-330**, click **AGREE & DOWNLOAD** to download the driver. Then, upload the driver to OBS and record the OBS URL. By using this method, you do not need to bind an EIP to GPU nodes.

Figure 3-330 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

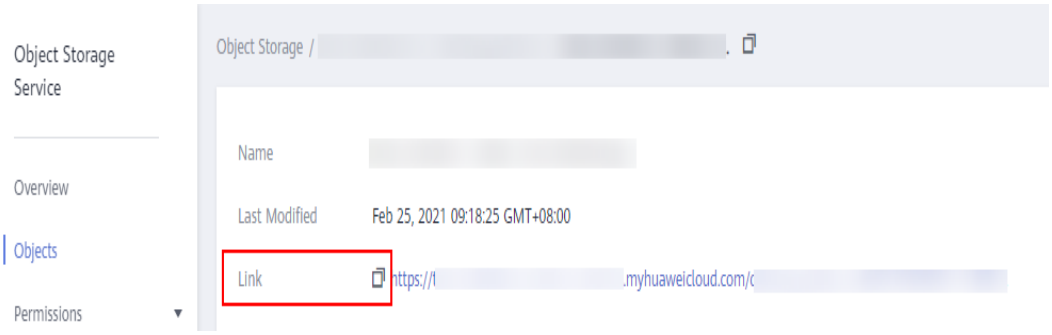
When the node is restarted, the driver will be downloaded and installed again. Ensure that the OBS bucket link of the driver is valid.

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 3-331 Copying an OBS link



----End

Components

Table 3-419 Add-on components

Component	Description	Resource Type
nvidia-driver-installer	Used for installing an NVIDIA driver on GPU nodes.	DaemonSet

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 3-420 Release history

Add-on Version	Supported Cluster Version	New Feature
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.
2.0.69	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.
1.2.28	v1.19 v1.21 v1.23 v1.25	• Adapts to Ubuntu 22.04. • Optimizes the automatic mounting of the GPU driver directory.

Add-on Version	Supported Cluster Version	New Feature
1.2.24	v1.19 v1.21 v1.23 v1.25	• Enables the node pool to configure GPU driver versions. • Supports GPU metric collection.
1.2.20	v1.19 v1.21 v1.23 v1.25	Sets the add-on alias to **gpu**.
1.2.17	v1.15 v1.17 v1.19 v1.21 v1.23	Adds 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	Supports 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.
1.2.2	v1.15 v1.17 v1.19	Supports the new EulerOS kernel.

Add-on Version	Supported Cluster Version	New Feature
1.2.1	v1.15 v1.17 v1.19	• CCE clusters 1.19 are supported. • Adds taint tolerance configuration.
1.1.13	v1.13 v1.15 v1.17	Supports kernel-3.10.0-1127.19.1.el7.x86_64 for CentOS 7.6.
1.1.11	v1.15 v1.17	• Allows users to customize driver addresses to download drivers. • Supports clusters v1.15 and v1.17.

3.14.12 CCE AI Suite (Ascend NPU)

Introduction

Ascend NPU is a device management add-on that supports Huawei NPUs in containers.

After this add-on is installed, you can create Ascend-accelerated nodes to quickly and efficiently process inference and image recognition.

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. Choose **Add-ons** in the navigation pane, locate **CCE AI Suite (Ascend NPU)** on the right, and click **Install**.

Step 2 Set NPU parameters. The add-on uses the following parameters by default. The default NPU settings provided by the add-on can satisfy most scenarios and require no changes.

```
{
    "check_frequency_failed_threshold": 100,
    "check_frequency_fall_times": 3,
    "check_frequency_gate": false,
    "check_frequency_recover_threshold": 100,
    "check_frequency_rise_times": 2,
    "container_path": "/usr/local/HiAI_unused",
```



```
    "host_path": "/usr/local/HiAI_unused"
}
```

Step 3 Click **Install**.

----End

Components

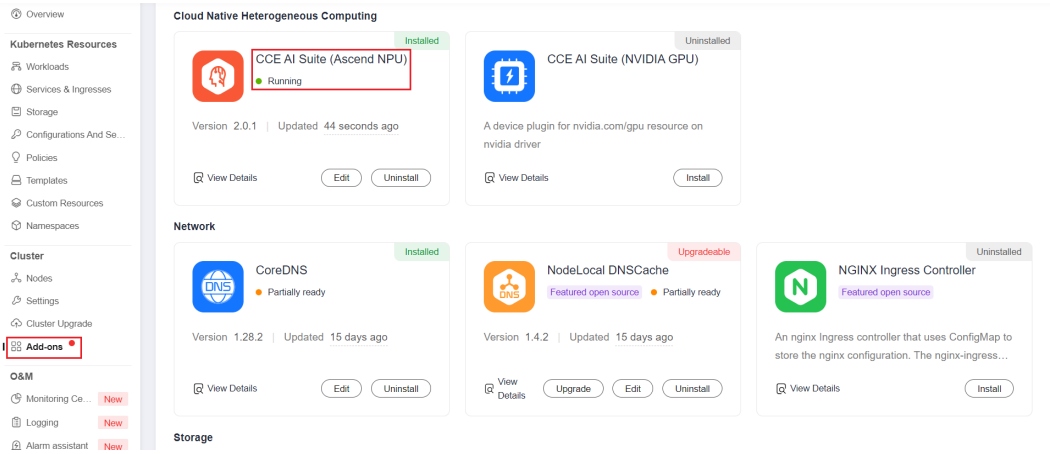
Table 3-421 Add-on components

Component	Description	Resource Type
npu-driver-installer	Used for installing an NPU driver on NPU nodes.	DaemonSet

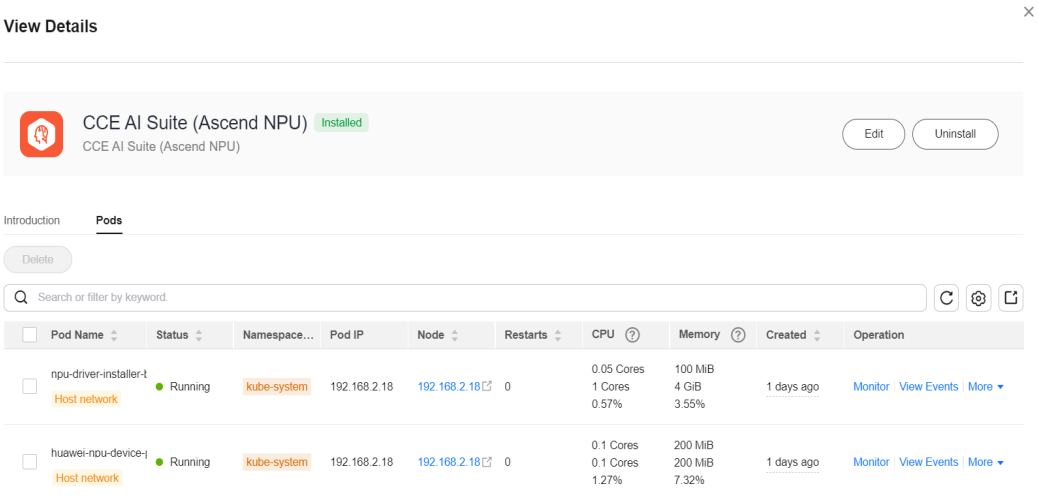
How to Check Whether the NPU Driver Has Been Installed on a Node

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 of the cluster indicating that the Ascend 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 3-422 Release history

Add-on Version	Supported Cluster Version	New Feature
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.
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	Supports automatical installation of NPU drivers.
1.2.5	v1.19 v1.21 v1.23 v1.25	Supports automatical installation of NPU drivers.
1.2.4	v1.19 v1.21 v1.23 v1.25	CCE clusters 1.25 are supported.

Add-on Version	Supported Cluster Version	New Feature
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	Adds 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	Adapts to the D310 C75 driver.
1.0.6	v1.13 v1.15 v1.17	Supports the Ascend C75 driver.
1.0.5	v1.13 v1.15 v1.17	Allows containers to use Huawei NPU add-ons.
1.0.3	v1.13 v1.15 v1.17	Allows containers to use Huawei NPU add-ons.

3.14.13 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 **3.6.5 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. Choose **Add-ons** in the navigation pane, locate **Volcano Scheduler** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-423 Volcano configuration

Parameter	Description
Add-on Specifications	Select **Standalone**, **Custom**, or **HA** for **Add-on Specifications**.

Parameter	Description
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select **Custom**, the recommended values for **volcano-controller** and **volcano-scheduler** 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 3-424**, 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 3-424**. 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 3-424 Recommended values for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	Requested vCPUs (m)	vCPU Limit (m)	Requested Memory (MiB)	Memory Limit (MiB)
50/5000	500	2000	500	2000
100/10,000	1000	2500	1500	2500
200/20,000	1500	3000	2500	3500
300/30,000	2000	3500	3500	4500
400/40,000	2500	4000	4500	5500
500/50,000	3000	4500	5500	6500
600/60,000	3500	5000	6500	7500
700/70,000	4000	5500	7500	8500

Step 3 Configure the add-on parameters.

Configure parameters of the default Volcano scheduler. For details, see **Table 3-426**.

```
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 3-425 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 3-426**.	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 3-426 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: 10 lifecycle.MaxScore: 200.0 lifecycle.SaturatedTresh: 1.0 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.	-	- 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**.
		• **selectorspread.weight**: Pods are evenly
			- 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 to different nodes. This parameter defaults to **0**. • **podtopologyspread.weight**: Pods are 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 **3.2.5.1 Cluster Configuration Management**.
		None	- plugins:
			  - name: 'cce-gpu-topology-predicate'
			  - name: 'cce-gpu-topology-priority'
			  - **name: 'cce-gpu'**

numa-aware	NUMA affinity scheduling. For details, see **3.6.5.6 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

networkresource	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'

Step 4 Configure scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-427 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 that AZ. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-428 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

Component	Description	Resource Type
volcano-agent	Cloud native hybrid agent, which is used for node QoS assurance, CPU burst, and dynamic resource oversubscription	DaemonSet
resource-exporter	Report the NUMA topology information of nodes.	DaemonSet

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.

Volcano allows you to configure the scheduler during installation, upgrade, and editing. The configuration will be synchronized to volcano-scheduler-configmap.

This section describes how to configure volcano-scheduler.

NOTE

Only Volcano of v1.7.1 and later support this function. On the new add-on page, options such as **resource_exporter_enable** are replaced by **default_scheduler_conf**.

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 right of the page, locate **Volcano Scheduler** and click **Install** or **Upgrade**. In the **Parameters** area, configure the Volcano parameters.

- Using **resource_exporter**:

```
{
    "ca_cert": "",
    "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"
            }
         ]
      }
   ]
  },
  "server_cert": "",
  "server_key": ""
}
```

After this function is enabled, you can use the functions of both numa-aware and resource_exporter.

Retaining the Original volcano-scheduler-configmap Configurations

If you want to use the original configuration after the plugin is upgraded, perform the following steps:

Step 1 Check and back up the original volcano-scheduler-configmap configuration.

Example:

```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
  default-scheduler.conf: |-
    actions: "enqueue, allocate, backfill"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
```



```
    - name: binpack
      arguments:
        binpack.cpu: 100
        binpack.weight: 10
        binpack.resources: nvidia.com/gpu
        binpack.resources.nvidia.com/gpu: 10000
  - plugins:
    - name: cce-gpu-topology-predicate
    - name: cce-gpu-topology-priority
    - name: cce-gpu
  - plugins:
    - name: nodelocalvolume
    - name: nodeemptydirvolume
    - name: nodeCSIscheduling
    - name: networkresource
```

Step 2 Enter the customized content in the **Parameters** area on the console.

```
{
  "ca_cert": "",
  "default_scheduler_conf": {
    "actions": "enqueue, allocate, backfill",
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          },
          {
            "name": "binpack",
            "arguments": {
              "binpack.cpu": 100,
              "binpack.weight": 10,
              "binpack.resources": "nvidia.com/gpu",
              "binpack.resources.nvidia.com/gpu": 10000
            }
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          }
        ]
```



```
        },
        {
          "plugins": [
            {
              "name": "nodelocalvolume"
            },
            {
              "name": "nodeemptydirvolume"
            },
            {
              "name": "nodeCSIscheduling"
            },
            {
              "name": "networkresource"
            }
          ]
        }
      ]
    },
    "server_cert": "",
    "server_key": ""
}
```

NOTE

When this function is used, the original content in volcano-scheduler-configmap will be overwritten. Therefore, you must check whether volcano-scheduler-configmap has been modified during the upgrade. If yes, synchronize the modification to the upgrade page.

----End

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 3-429 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"]

Metric	Type	Description	Label
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
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 3-430**) 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 3-430 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

Item	API Group	API Version	Resource Level
Queue	scheduling.volcano.sh	v1beta1	Cluster

Related Operations

- 3.6.6.1 Dynamic Resource Oversubscription
- 3.6.5.6 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 3-431 Release history

Add-on Version	Supported Cluster Version	New Feature
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.
1.11.21	v1.19.16 v1.21 v1.23 v1.25 v1.27 v1.28	• Kubernetes 1.28 are supported. • 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.

Add-on Version	Supported Cluster Version	New Feature
1.11.6	v1.19.16 v1.21 v1.23 v1.25 v1.27	• Supported Kubernetes v1.27. • 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.
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.

Add-on Version	Supported Cluster Version	New Feature
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.
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.

Add-on Version	Supported Cluster Version	New Feature
1.3.1	v1.15 v1.17 v1.19	• Upgrades the volcano framework to the latest version. • Supported Kubernetes v1.19. • 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.
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.

3.14.14 CCE Secrets Manager for DEW

Introduction

The dew-provider 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.

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.
- The dew-provider add-on can be installed only on clusters v1.19 or later.
- The dew-provider add-on can be installed in CCE standard clusters and CCE Turbo clusters.
- 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. Click **Add-ons** in the navigation pane, 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.

Parameter	Description
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 click **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 3-432 Add-on components

Component	Description	Resource Type
dew-provider	A component that obtains specified secrets from CSMS and mounts them to the pods	DaemonSet
secrets-store-csi-driver	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.

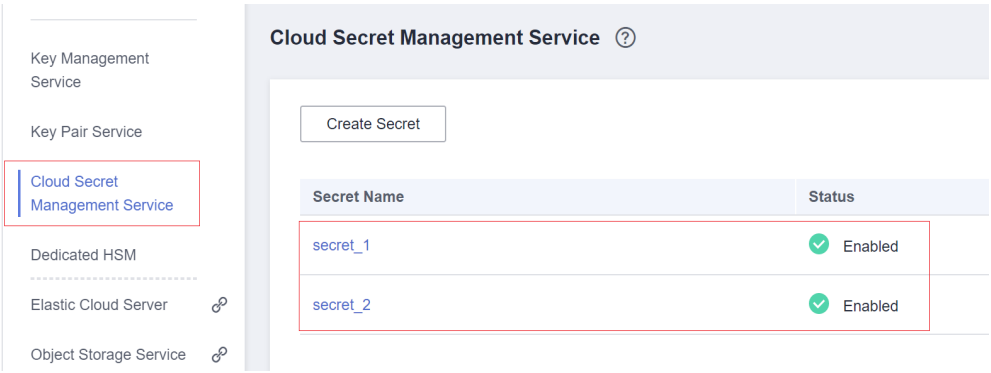
1. Create a ServiceAccount object, **which declares the secret names that can be used by services. If a user references a secret that is not declared here, the mounting will fail. As a result, the pod cannot run**.

   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.
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 subsequently.

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.

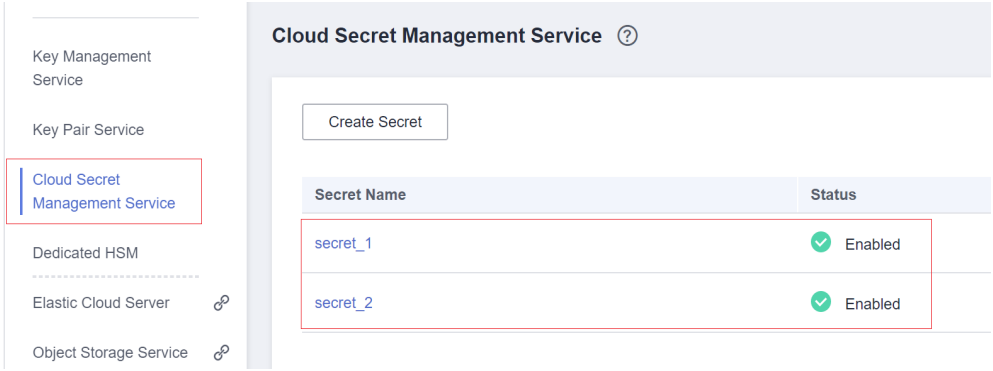
Step 1 Create a ServiceAccount.

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 3-433 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**).

Parameter	Type	Mandatory	Description
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. 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 3-434 secretObjects parameter

Parameter	Type	Mandatory	Description
secretName	String	Yes	Secret name
type	String	Yes	Secret type
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 3-435 Release history

Add-on Version	Supported Cluster Version	New Feature
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.

3.14.15 CCE Network Metrics Exporter

Introduction

Dolphin is an add-on for monitoring and managing container network traffic. The current version of dolphin can collect traffic statistics of containers that do not use the host network mode in CCE Turbo clusters and performs nodewide container connectivity check.

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 information has been adapted to the Prometheus format. You can call the Prometheus API to view monitoring data.

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.
- After the add-on is installed, traffic is by default not monitored. You need to create a MonitorPolicy to configure a monitoring task for traffic monitoring.
- Pods using the host network mode cannot be monitored.
- Ensure that there are sufficient 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. Click **Add-ons** in the navigation pane, locate **CCE Network Metrics Exporter** on the right, and click **Install**.

Step 2 On the Install Add-on page, view 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 click **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 3-436 Add-on components

Component	Description	Resource Type
dolphin	Used to monitor the container network traffic of 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 3-437 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

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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)

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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)
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

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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
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

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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
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

Monitoring Metric	Monitoring Item	Granularity	Supported Runtime	Supported Cluster Version	Supported Add-on Version	Supported OS
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
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 dolphin add-on, for example, labelSelector. By default, all containers 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].

Labels must comply with the following rules. The corresponding regular expression is (^[a-zA-Z_]$)|(^([a-zA-Z][a-zA-Z0-9_]|_[a-zA-Z0-9])([a-zA-Z0-9_]){0,254}$).

- 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 256 characters.
- The value cannot start with a digit or double underscores (_).
- The format of a single label must comply with A-Za-z_0-9.

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 MB kernel memory. A flow-based TCP monitoring task (one or more TCP monitoring items enabled in a MonitorPolicy) occupies 14 MB 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 CloudScope data fetches.
- If CloudScope 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 monitoring data collected by this add-on is exported in Prometheus exporter format, which can be obtained in the following ways:

- Directly access service port 10001 provided by the dolphin add-on, for example, http://{POD_IP}:10001/metrics.

  Note that if you access the dolphin 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 3-438 Release history

Add-on Version	Supported Cluster Version	New Feature
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 for 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.
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	• Adds the description that only EulerOS is supported.

Add-on Version	Supported Cluster Version	New Feature
1.2.2	v1.19 v1.21 v1.23 v1.25	• Supports health check for pod VPCs.
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.
1.0.1	v1.19 v1.21	• Supports traffic statistics persistence and local socket communications.

3.14.16 NodeLocal DNSCache

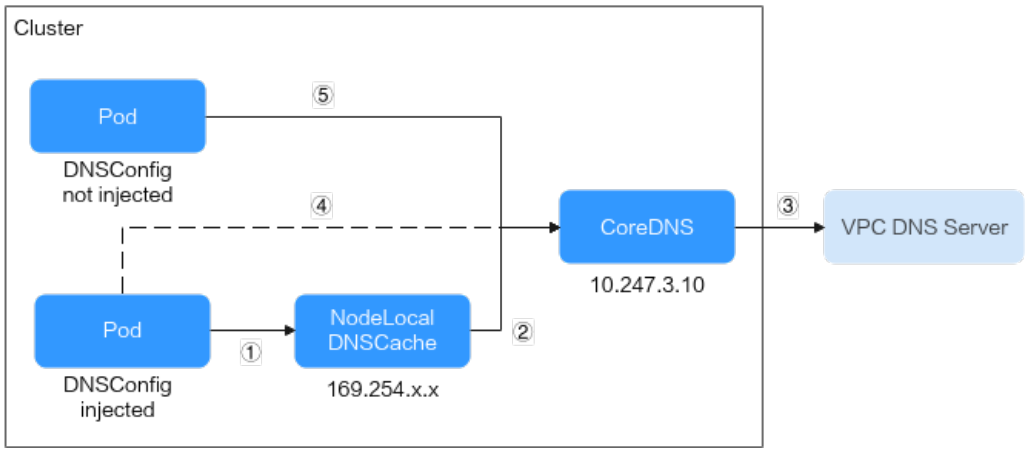
Introduction

NodeLocal DNSCache is an add-on developed based on the community **NodeLocal DNSCache**. This add-on functions as a DaemonSet to run the DNS cache proxy on cluster nodes 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 3-332 NodeLocal DNSCache query path



The resolution lines are described as follows:

- 1. By default, pods that have been injected into the DNS local cache will use the NodeLocal DNSCache to resolve requested domain names.
- 2. If the NodeLocal DNSCache's cache cannot resolve a request, it will ask the cluster's CoreDNS for resolution.
- 3. CoreDNS resolves domain names outside of the cluster by using the DNS server in the VPC.
- 4. If a pod injected into the local DNS cache cannot access the NodeLocal DNSCache, the domain name will be resolved through CoreDNS.
- 5. By default, CoreDNS resolves domain names for pods that are not injected into the local DNS cache.

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. Choose **Add-ons** in the navigation pane, locate **NodeLocal DNSCache** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-439 NodeLocal DNSCache configuration

Parameter	Description
Add-on Specifications	Select **Standalone**, **HA**, or **Custom** for **Add-on Specifications**.
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.

Parameter	Description
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. If you select **Custom**, you can adjust the container specifications as required.

Step 3 Configure the add-on parameters.

- **enable_dnsconfig_admission**: After this function is enabled, a DNSConfig dynamic injection controller will be created. The controller intercepts pod creation requests in the namespace labeled with **node-localdns-injection=enabled** based on Admission Webhook, and automatically configures **Pod dnsConfig** that uses the DNS cache. If this function is disabled or the pod belongs to a non-target namespace, you must manually configure DNSConfig for the pod.
- **Target Namespace**: This parameter is available after **DNSConfig Automatic Injection** is enabled. Only NodeLocal DNSCache of v1.3.0 or later supports this function.
  - **All Enabled**: 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 configuration**: 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 scheduling policies for the add-on.

NOTE

- Scheduling policies do not take effect on add-on instances 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 3-440 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 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. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

----End

Components

Table 3-441 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 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. (Pods cannot be automatically injected into system namespaces such as kube-system.)
- Manual configuration: Manually configure the **dnsConfig** field of the pod.

Auto injection

The following conditions must be met:

- **Automatic DNSConfig injection** has been enabled during the add-on installation.
- The **node-local-dns-injection=enabled** label has been added 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**
- The new pod does not run in system namespaces such as kube-system and kube-public namespace.
- 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**.

After auto injection is enabled, the following **dnsConfig** settings are automatically added to the created pod. In addition to the NodeLocal DNSCache address 169.254.20.10, the CoreDNS 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
...
```

Manual configuration

Manually add the **dnsConfig** settings to the pod.

Create a pod and add the NodeLocal DNSCache IP address 169.254.20.10 to the DNSConfig nameservers configuration.

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

```
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
```

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 this label.

Step 1 Check the add-on.

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 **NodeLocal DNSCache** on the right, and click **Edit**.
2. In the **Parameters** area, check whether **DNSConfig Automatic Injection** is enabled.

If **DNSConfig Automatic Injection** 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 Automatic Injection** 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

3.7.5.4 Using NodeLocal DNSCache to Improve DNS Performance

Change History

Table 3-442 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
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

Add-on Version	Supported Cluster Version	New Feature	Community Version
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 basic image OS of the add-on pods to HCE OS 2.0.	1.22.20
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 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

3.14.17 Cloud Native Cluster Monitoring

Introduction

kube-prometheus-stack uses Prometheus-operator and Prometheus to provide easy-to-use, end-to-end Kubernetes cluster monitoring.

This add-on allows the monitoring data to be interconnected with Container Intelligent Analysis (CIA) so that you can view monitoring data and configure alarms on the CIA console.

Open source community: **https://github.com/prometheus/prometheus**

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 whitelist 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: []
  ```

Permissions

The node-exporter component of the kube-prometheus-stack 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

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, locate **Cloud Native Cluster Monitoring** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

- **Deployment Mode**: This parameter is available for the kube-prometheus-stack version 3.7.1 or later.
  - **Agent mode**: Data is not stored locally, requiring fewer resources than the server mode. However, this mode does not support HPA.

NOTE

In the agent mode, monitoring data is no longer stored locally. Therefore, AOM or a third-party monitoring system must be accessed.

  - **Server mode**: Data is stored locally, requiring more resources than the agent mode. In this mode, all kube-prometheus-stack functions are supported.
- **Containers**: component instance created by the add-on. For details, see **Components**. You can select or customize a specification as required.

Step 3 Configure related parameters.

- **Connect to Third Party**: To report Prometheus data to a third-party monitoring system, enter the address and token of the third-party monitoring system and determine whether to skip certificate authentication.
- **User-defined indicator 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 **3.9.6.1 Monitoring Custom Metrics Using Cloud Native Cluster Monitoring**.
- **Prometheus HA**: The Prometheus-server, Prometheus-operator, thanos-query, custom-metrics-apiserver and alertmanager components are deployed in multi-instance mode in the cluster.
- **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.
- **Number of collected shards**: Collected targets are distributed among different Prometheus shards. This increases the upper limit of the metrics collection throughput but will consume more resources. Therefore, this parameter is recommended for large-scale clusters.
- **Collection Interval**: Configure the collection interval.
- **Storage**: Select the type and size of the disk for storing monitoring data. Uninstalling the add-on will not delete this volume.

  NOTE

  An available PVC named **pvc-prometheus-server** exists in namespace **monitoring** and will be used as the storage source.

Step 4 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 the add-on is in the server mode and then perform the following steps:
  a. Collect custom metrics reported by applications to Prometheus. For details, see **3.9.6.1 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 use this add-on to provide system resource metrics (such as CPU and memory usage) for workload auto scaling, enable the Metric API. For details, see **Providing Resource Metrics Through the Metrics API**. After the configuration, you can use Prometheus to collect system resource metrics. (This configuration is not recommended).

----End

Components

All Kubernetes resources created during kube-prometheus-stack add-on installation are created in the namespace named **monitoring**.

Table 3-443 Add-on components

Component	Description	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.	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.	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.	StatefulSet
thanosSidecar	Available only in HA mode. Runs with prometheus-server in the same pod to implement persistent storage of Prometheus metric data.	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.	Deployment
adapter (workload name: custom-metrics-apiserver)	Aggregates custom metrics to the native Kubernetes API Server.	Deployment

Component	Description	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.	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	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.	Deployment
clusterProblemDetector (workload name: cluster-problem-detector)	Monitors cluster exceptions.	Deployment

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 the following information 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
```

Creating an HPA Policy Using Custom Metrics

HPA policies can only be used when Cloud Native Cluster Monitoring is deployed in the server mode. You can configure custom metrics required by HPA policies in the **user-adapter-config** ConfigMap.

NOTICE

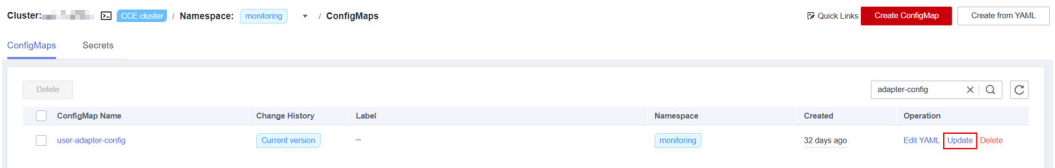
To use Prometheus to monitor custom metrics, the application needs to provide a metric monitoring API. For details, see **Prometheus Monitoring Data Collection**.

In this section, the nginx metric (nginx_connections_accepted) in **3.9.6.1 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 3-333 Updating a ConfigMap



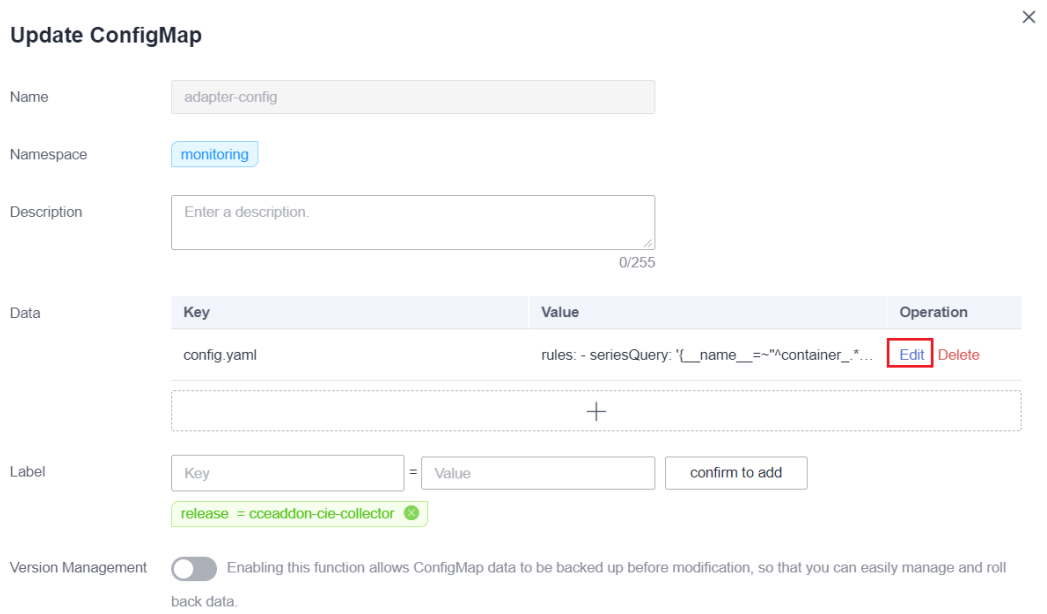
Step 3 In **Data**, click **Edit** for the **config.yaml** file to add a custom metric collection rule under the **rules** field. Click **OK**.

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:

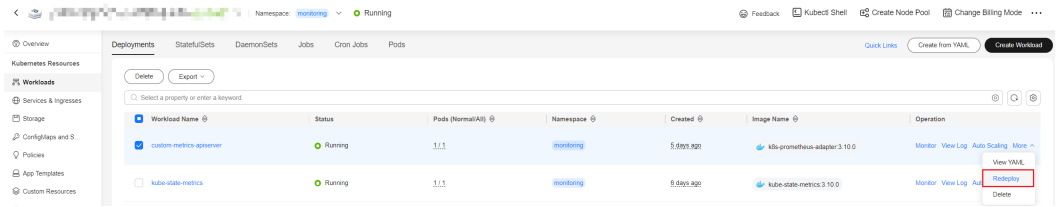
```
rules:
# Match the metric whose name is nginx_connections_accepted. The metric name must be confirmed. Otherwise, the HPA controller cannot get the metric.
- seriesQuery: '{__name__=~"nginx_connections_accepted",container!="POD",namespace!="",pod!=""}'
  resources:
    # Specify pod and namespace 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 3-334 Modifying ConfigMap data



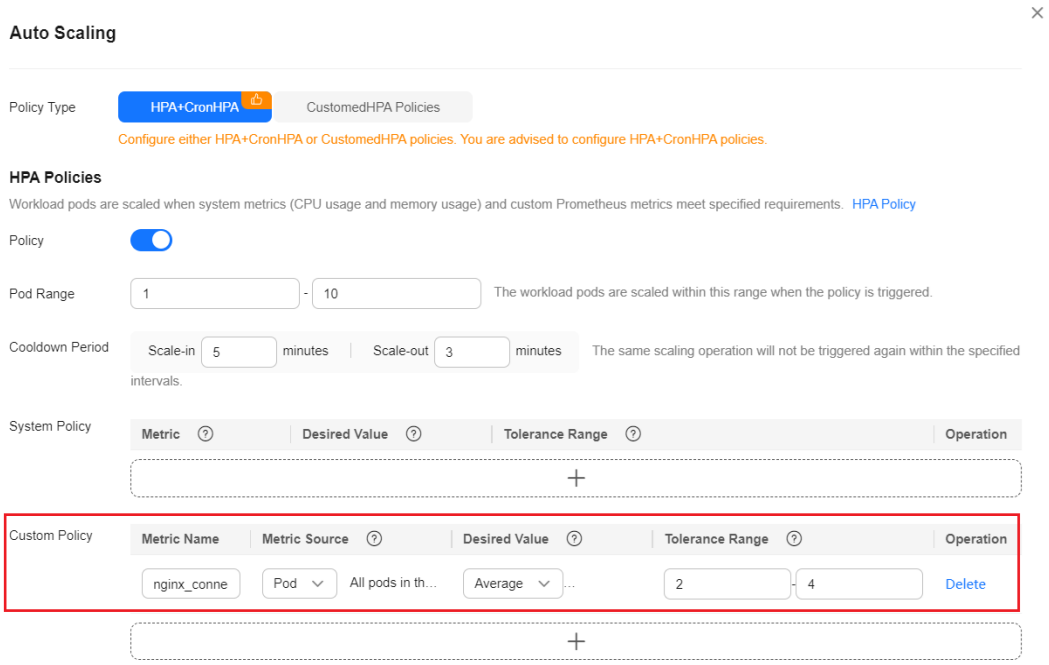
Step 4 Redeploy the **custom-metrics-apiserver** workload in the **monitoring** namespace.

Figure 3-335 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 3-336 Creating an HPA policy



----End

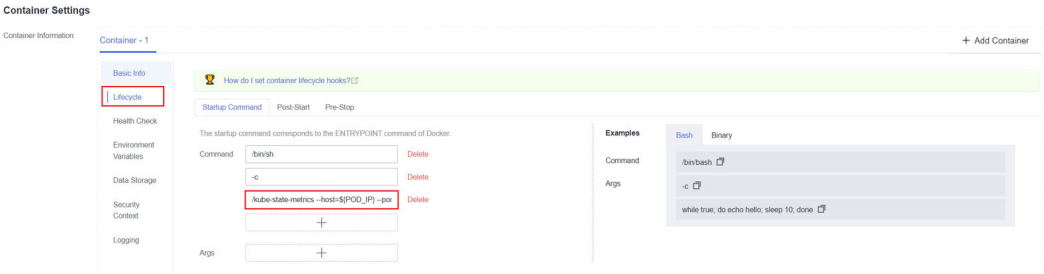
Collecting All Labels and Annotations of a Pod

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, find the **kube-state-metrics** workload on the **Deployments** tab page, and click **Upgrade** in the **Operation** column.

Step 3 In the **Lifecycle** area of the container settings, edit the startup command.

Figure 3-337 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 CloudScope.

```
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 3-444 Release history

Add-on Version	Supported Cluster Version	New Feature	Community Version
3.9.5	v1.21 v1.23 v1.25 v1.27 v1.28	• Added the collecting custom metrics function. This function is enabled by default. • Removed support for clusters of v1.17 and v1.19. • Removed Grafana from the original add-on. 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
3.8.2	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27	Fixed some issues.	2.35.0

Add-on Version	Supported Cluster Version	New Feature	Community Version
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	Supports collection of pod metrics on Virtual Kubelet.	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. • Supports 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

3.14.18 Cloud Native Logging

Introduction

log-agent is built based on Fluent Bit and OpenTelemetry. It collects 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 Logging 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 Logging is installed, events are reported to AOM by this add-on instead of the control plane component. After Cloud Native Logging is uninstalled, events will not be reported to AOM.

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
Logging 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 2,000 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 log-agent 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. Choose **Add-ons** in the navigation pane, locate **Cloud Native Logging** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-445 Cloud Native Logging Configuration

Parameter	Description
Add-on Specifications	The add-on specifications can be of the **Low**, **High**, or **custom-resources** type.
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.
Containers	The log-agent add-on contains the following containers, whose specifications can be adjusted as required: • **fluent-bit**: log collector, which is installed on each node as a DaemonSet. • **cop-logs**: generates and updates configuration files on the collection side. • **log-operator**: parses and updates log rules. • **otel-collector**: forwards logs collected by **fluent-bit** to LTS.

Step 3 Configure scheduling policies for the add-on.

NOTE

Scheduling policies do not take effect on add-on instances of the DaemonSet type.

Table 3-446 Configurations for add-on scheduling

Parameter	Description
Multi AZ	• **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 that AZ. • **Required**: Deployment pods of the add-on will be forcibly scheduled to nodes in different AZs. If there are fewer AZs than pods, the extra pods will fail to run.

Step 4 Click **Install**.

----End

Components

Table 3-447 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 3-448**.

Table 3-448 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 **3.9.4.2.1 Collecting Container Logs Using Cloud Native Logging**.
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 3-449 Release history

Add-on Version	Supported Cluster Version	New Feature
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.

Add-on Version	Supported Cluster Version	New Feature
1.3.0	v1.17 v1.19 v1.21 v1.23 v1.25	Supported clusters 1.25.
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.

3.14.19 Container Image Signature Verification

Introduction

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.

Constraints

- The CCE cluster version can only be V1.23 or later.
- 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. Choose **Add-ons** in the navigation pane, locate **swr-cosign** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-450 swr-cosign specifications

Parameter	Description
Add-on Specifications	Select **Standalone**, **HA**, or **Custom** for **Add-on Specifications**.

Parameter	Description
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.
Containers	If you select **Custom**, you can adjust the container specifications as required.

Step 3 Configure the add-on parameters.

Table 3-451 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 click **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 3-452 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 **3.5.2.1 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 signature is successful, 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 Sign an image.

1. Use the **Linux pgp** signature tool to generate a key pair.
   ```
   gpg --quick-generate-key {uid}
   ```

   CAUTION

   Keep the key pair and your password secure.

2. Print the information about the image to be signed.
   ```
   echo "{\"critical\":{\"identity\":{\"docker-reference\":\"{namespace}/{repo}\"},\"image\":{\"docker-manifest-digest\":\"{sha256}\"},\"type\":\"atomic container signature\"}}" > payload.txt
   ```

   NOTE

   - {namespace}/{repo}: organization of the image to be signed/name of the image to be signed
   - {sha256}": SHA256 value of the image, for example, sha256:aba27e8e**********1d566e218cb3ecaa424

3. Sign the image and encrypt it using Base64.

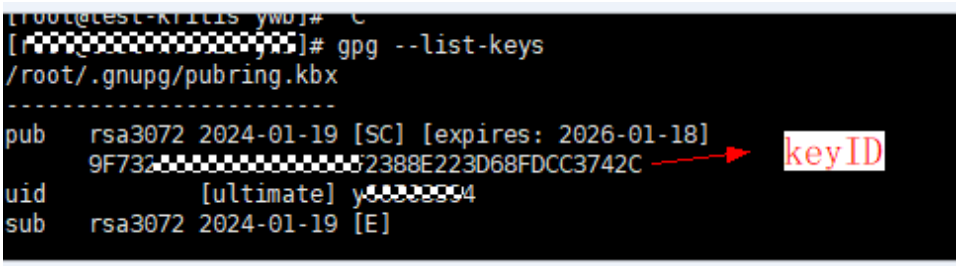
   a. Sign the image.
      ```
      gpg --output signature.txt --armor -u {uid} --sign payload.txt
      ```

   b. Encrypt it using Base64.
      ```
      cat signature.txt | base64 -w0
      ```

4. Generate a signature file.
   ```
   echo "{\"PublicKeyID\": \"{KeyID}\", \"Signature\": \"{Sign}\", \"SerializedPayload\": null}" > {sha256}.asc
   ```

   NOTE

   - {KeyID}: Enter the key pair fingerprint, which can be obtained by running **gpg --list-keys**.

   - {Sign}: Enter the signature generated in **Step 5.3**.
   - {sha256}: Enter the SHA256 value of the image.

5. Upload the signature file generated in **Step 5.4** to a public OBS bucket. The path in the bucket is {namespace}/{repo}/{sha256}.asc.
   ```
   curl -X PUT https:// Bucket name.OBS Domain name /{namespace}/{repo}/{sha256}.asc -T {sha256}.asc
   ```

Step 6 Go back to the CCE console, and check whether the signed image can be used to create a workload successfully.

----End

3.14.20 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. Choose **Add-ons** in the navigation pane, locate **Grafana** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 3-453 Grafana configuration

Parameter	Description
Add-on Specifications	The specifications can be **Custom**.
Containers	If you select **Custom**, you can adjust the container specifications as required.

Step 3 Configure the add-on parameters.

Table 3-454 Grafana parameters

Parameter	Description
Disk Type	To install Grafana, you need to create EVS disks to store the local data. When Grafana is uninstalled, the EVS disks created for Grafana will not be deleted. EVS disks are charged by storage capacity and occupy your EVS disk quotas. You can select the EVS disk type on the management console. Different sites may support different disk types.
Capacity (GiB)	The EVS disk size is 5 GiB by default. You can expand the disk capacity after the disk is created. For details, see **Related Operations**.
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.

Step 4 Configure scheduling policies for the add-on.

Table 3-455 Configurations for add-on scheduling

Parameter	Description
Node Affinity	• **Not configured**: Node affinity is disabled for the add-on. • **Node Affinity**: 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. • **Specified Node Pool Scheduling**: Specify the node pool where the add-on is deployed. If you do not specify the node pool, the add-on will be randomly scheduled based on the default cluster scheduling policy. • **Custom Policies**: 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 **3.5.3.12 Taints and Tolerations**.

Step 5 Click **Install**.

After the add-on is installed, select the cluster and click **Add-ons** in the navigation pane. On the displayed page, view the add-on in the **Add-ons Installed** area.

----End

Components

Table 3-456 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 **Networking** 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 3-338 Grafana panel

----End

Change History

Table 3-457 Release history

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	CCE 1.29 clusters are supported.
1.1.0	v1.17 v1.19 v1.21 v1.23 v1.25 v1.27 v1.28	Supported the open source Grafana.

3.14.21 e-backup (EOM)

Introduction

The e-backup add-on offers cluster backup and restoration. It backs up application data and service data to OBS and provides local and remote data backup.

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
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

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: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHWOBS
```

- 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.

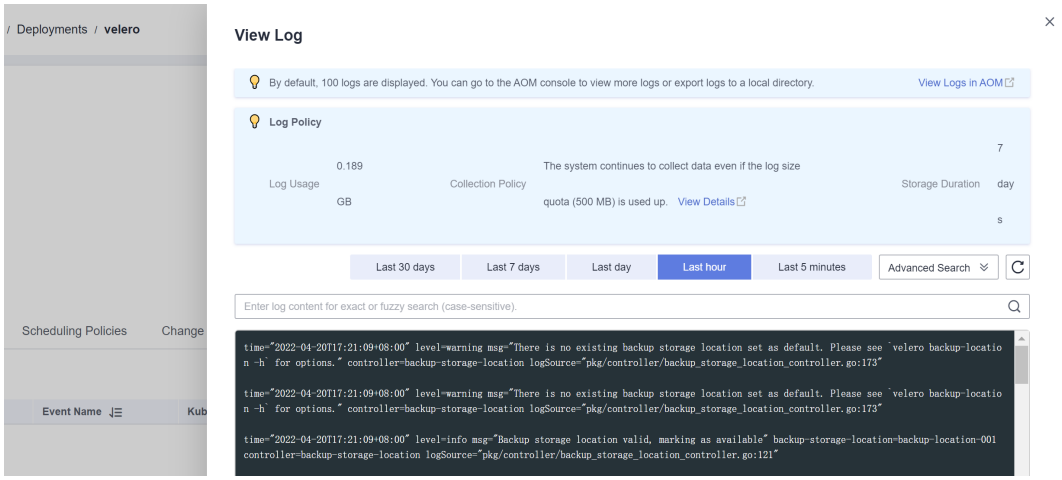
```
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  name: backup-location-001
  namespace: velero            # The object must be in the same namespace as e-backup.
spec:
  config:
    endpoint: obs.{regionname}.myhuaweicloud.com   # OBS endpoint
  credential:
    name: secret-secure-opaque   # 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
```

Parameter description:

- 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**):

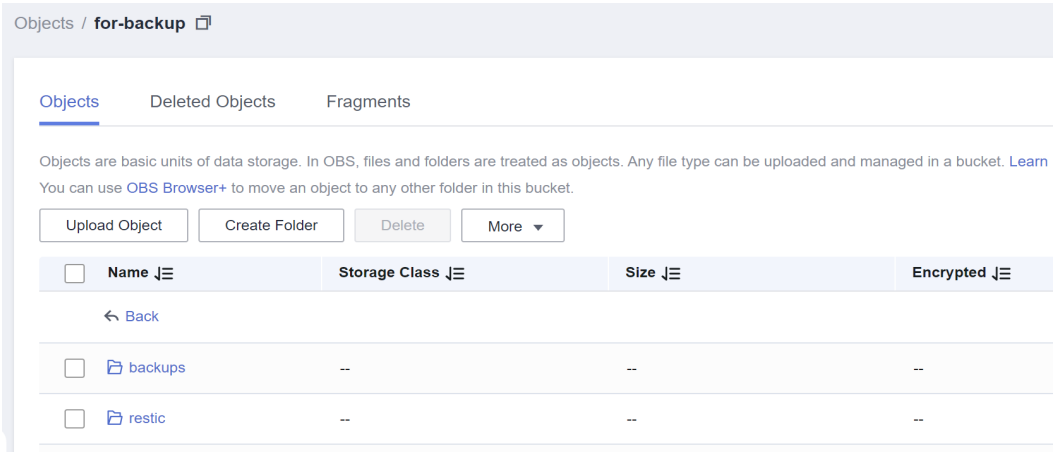
```
$ kubectl -n velero get backups backup-01 -o yaml | grep  "phase"
  phase: Completed

$ 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
```

Parameter description:

- **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
```

Parameter description:

- 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 3-458 Release history

Add-on Version	Supported Cluster Version	New Feature
1.2.0	v1.15 v1.17 v1.19 v1.21	• Supports EulerOS 2.0 (SP5, SP9). • Supports security hardening. • Optimizes functions.

3.14.22 web-terminal (EOM)

The web-terminal add-on is a lightweight terminal server that allows you to use kubectl on the web UI. It provides a remote command-line interface (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**

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. Choose **Add-ons** in the navigation pane, 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. Choose **Add-ons** in the navigation pane.

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 **3.16.3 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 3-459 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	• Adds 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**

3.14.23 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**

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. Choose **Add-ons** in the navigation pane, locate **Prometheus** on the right, and click **Install**.

Step 2 In the **Configuration** step, set the following parameters:

Table 3-460 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 the following information 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 3-461 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**

3.14.24 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.

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. Choose **Add-ons** in the navigation pane, 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

3.15 Helm Chart

3.15.1 Overview

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 **3.15.2 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 **3.15.4 Deploying an Application Through the Helm v2 Client** and **3.15.5 Deploying an Application Through the Helm v3 Client**.

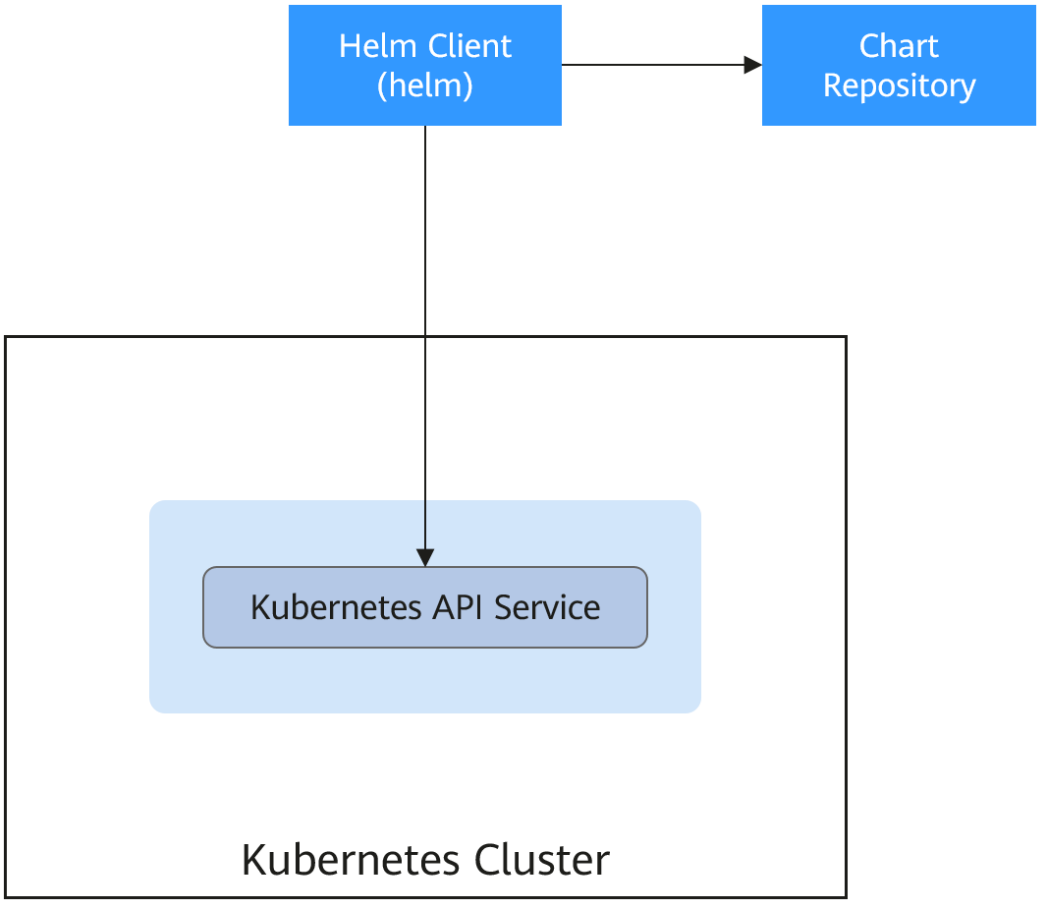
Helm

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.

3.15.2 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.

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 3-462**, the parameters marked with * are mandatory.

  Table 3-462 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 3-463**.

Table 3-463 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 Click **More** > **Roll Back** for the workload to be rolled back, select the workload version, and click **Roll back to this version**.

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

3.15.3 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 old manifest, its live state, and the new manifest when generating a patch. Helm compares the current live state with the live state of the old 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**

3.15.4 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 on this **website**, 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 run:
  ```
  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 3-464 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 from 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**.

3.15.5 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 run:

  ```
  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 3-465 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**.

3.15.6 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

3.16 Permissions

3.16.1 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.

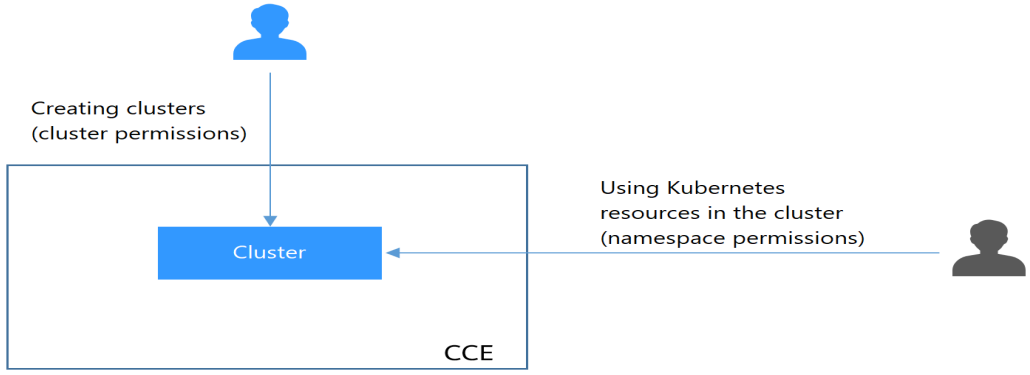
  Cluster-level permissions involve non-Kubernetes APIs in CCE clusters and support fine-grained IAM policies and enterprise project management.

- **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 3-339 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 3-466** lists the namespace permissions of different users.

Table 3-466 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 3-466**.

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 3-467 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	√	√

Permission	API	Action	IAM Project	Enterprise Project
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 3-468 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	√	√

Permission	API	Action	IAM Project	Enterprise Project
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 3-469 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	√	√
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 3-470 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 3-471 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
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 3-472 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 3-473 Storage management actions

Permission	API	Action	IAM Project	Enterprise Project
Creating a PersistentVolumeClaim	POST /api/v1/namespaces/{namespace}/cloudpersistentvolume-claims	cce:storage:create	√	√
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 3-474 Add-on management actions

Permission	API	Action	IAM Project	Enterprise Project
Creating an add-on instance	POST /api/v3/addons	cce:addonInstance:create	√	√

Permission	API	Action	IAM Project	Enterprise Project
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 3-475 Quota management actions

Permission	API	Action	IAM Project	Enterprise Project
Obtaining quota details	GET /api/v3/projects/{project_id}/quotas	cce:quota:get	√	√

3.16.2 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** at the same time.
- 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 **3.16.5 Permission Dependency of the CCE Console**.

Prerequisites

- Before granting permissions to user groups, get familiar with the system policies listed in **Permissions** for CCE. For the system policies of other services, see **System Permissions**.

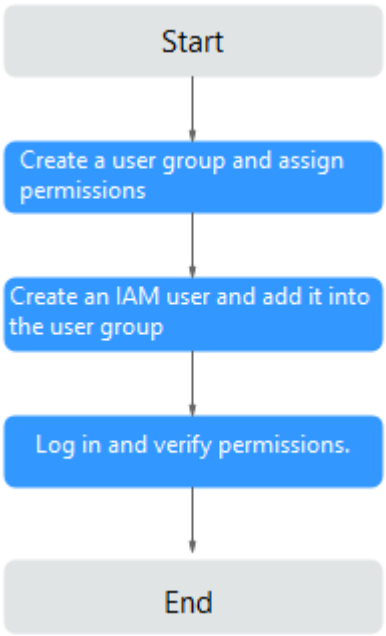
- A user with the Security Administrator role (for example, your account) has all IAM permissions except role switching. Only these users can view user groups and their permissions on the **Permissions** page on the CCE console.

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 3-340 Process of assigning CCE permissions



1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console, and assign CCE permissions, for example, the **CCE ReadOnlyAccess** policy to the group.

   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**.

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.

   - Log in to the management console, switch to the CCE console, and buy a cluster. If you fail to do so (assuming that only the **CCE ReadOnlyAccess**

permission is assigned), the **CCE ReadOnlyAccess** policy has already taken effect.

- Switch to the console of any other service. If a message appears indicating that you do not have the required permissions for accessing the service, the **CCE ReadOnlyAccess** policy has already taken 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**, **SWR Admin**, and **APM FullAccess**. For more information about dependencies, see **System 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 3-476 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.

Action	Specific Action	Description
	cce:cluster:get	Obtain cluster details.
	cce:node:create	Add a node.
	cce:node:delete	Delete one or more nodes.
	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.

Action	Specific Action	Description
	cce:addonInstance:list	List all add-on instances.
	cce:addonTemplate:list	List all add-on templates.
	cce:chart:list	List all charts.
	cce:chart:delete	Delete a chart.
	cce:chart:update	Update a chart.
	cce:chart:upload	Upload a chart.
	cce:chart:get	Obtain information about a chart.
	cce:release:get	Obtain information about 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:*:*	-	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.
sfs:*:get*	None	View SFS resource details.
sfs:shares:ShareAction	None	Share SFS resources for scaling.
aom:*:get	None	View Application Operations Management (AOM) resource details.
aom:*:list	None	List AOM resources.
aom:autoScalingRule:*	None	Perform all operations on AOM auto scaling rules.

Action	Specific Action	Description
apm:icmgr:*	None	Perform operations on the ICAgent in Application Performance Management (APM).
lts:*:*	None	Perform all operations on Log Tank Service (LTS).

Table 3-477 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.
	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**.

Action	Specific Action	Description
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.
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 details of all VPC resources (including enhanced 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.
vpc:*:list	None	List all VPC resources (including enhanced load balancers).
sfs:*:get*	None	View SFS resource details.
sfs:shares:ShareAction	None	Share SFS resources for scaling.
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.

Custom Policies

Custom policies can be created as a supplement to the system-defined policies of CCE. For the actions that can be added to custom policies, see **Permissions Policies 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: Edit JSON policies from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. This section provides examples of common custom CCE policies.

Example Custom Policies:

- 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.

  The following method can be used if you need to assign permissions of the **CCEFullAccess** policy to a user but you want to prevent the user from deleting nodes (**cce:node:delete**). Create a custom policy for denying node deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on CCE except deleting nodes. The following is an example of a deny policy:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "cce:node:delete"
            ]
        }
    ]
}
```

- Example 3: Defining permissions for multiple services in a policy

  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 actions of multiple services:

```
{
    "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 Name	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

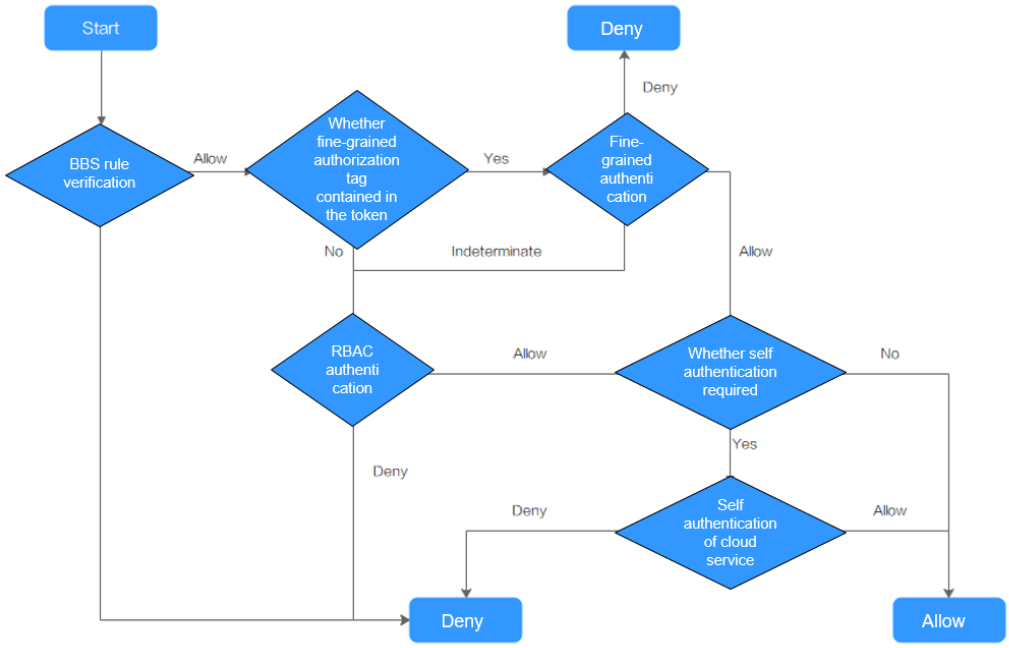
NOTE

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 Management** > **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.



3.16.3 Namespace Permissions (Kubernetes RBAC-based)

Namespace Permissions (Kubernetes RBAC-based)

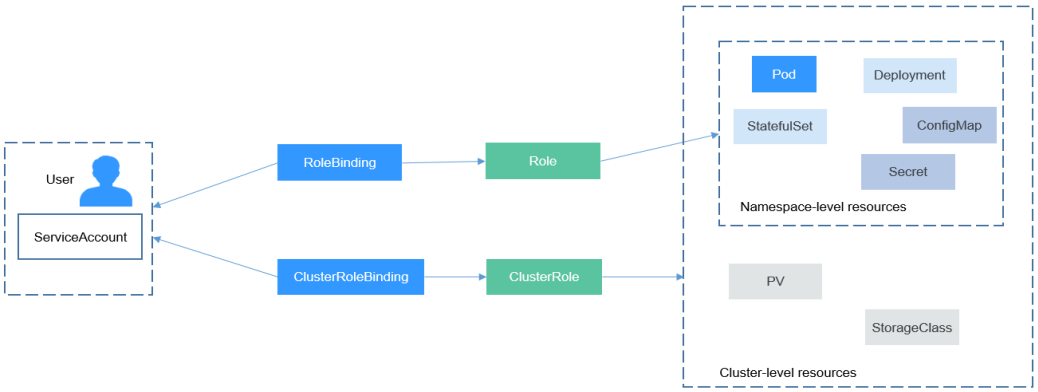
You can regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- 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 3-341 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.

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 3-478** lists the namespace permissions of different users.

Table 3-478 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.

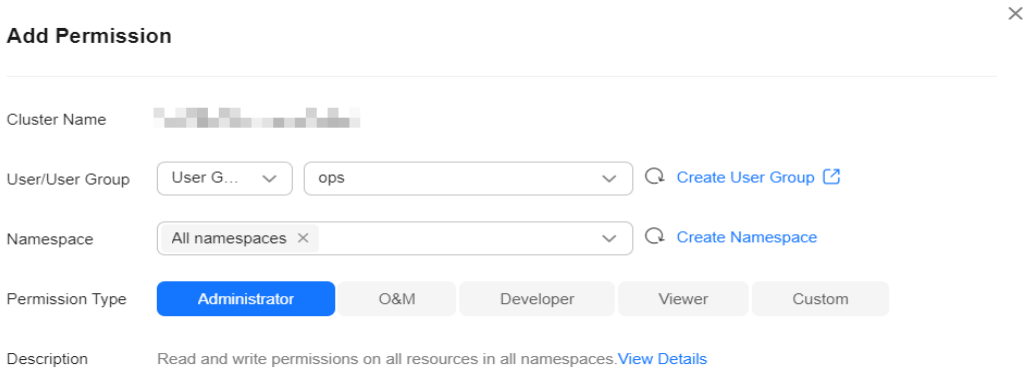
Step 3 Click **Add Permissions** in the upper right corner.

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 3-342 Configuring namespace permissions

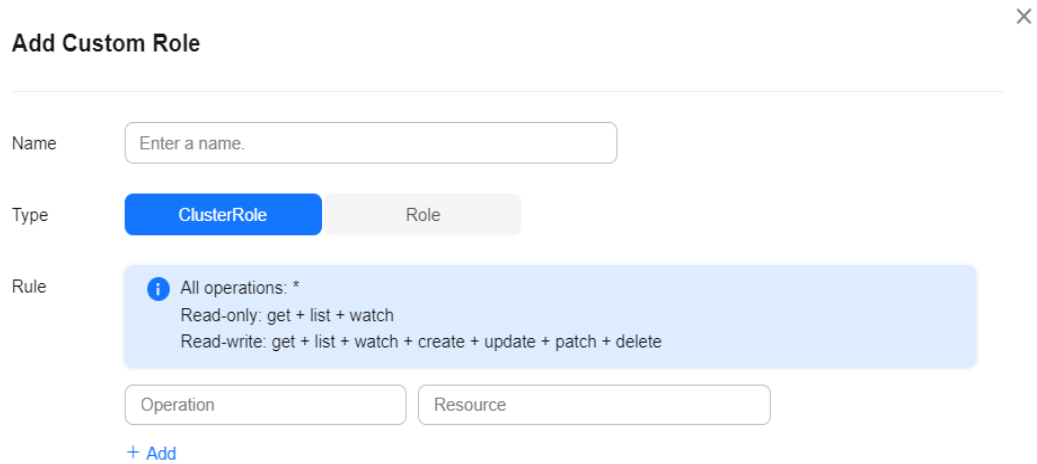


Permissions can be customized as required. After selecting **Custom** for **Permission Type**, click **Add Custom Role** on the right of the **Custom** parameter. In the dialog box displayed, enter a name and select a rule. After the custom rule is created, you can select a value from the **Custom** 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 3-343 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**.

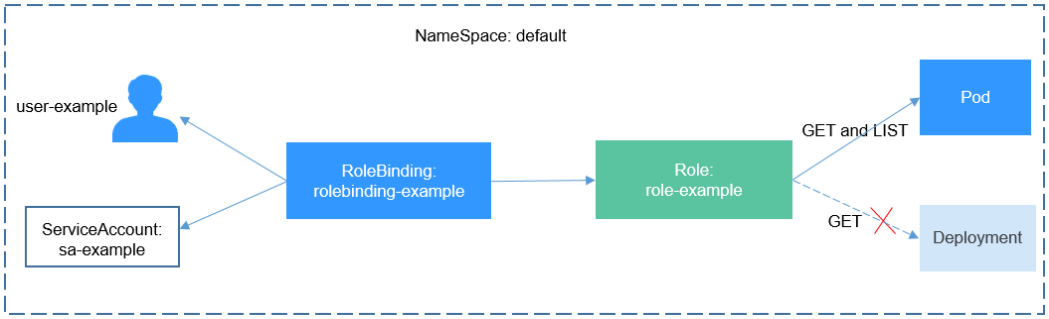
For details, see **Using RBAC Authorization**.

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 3-344 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
```

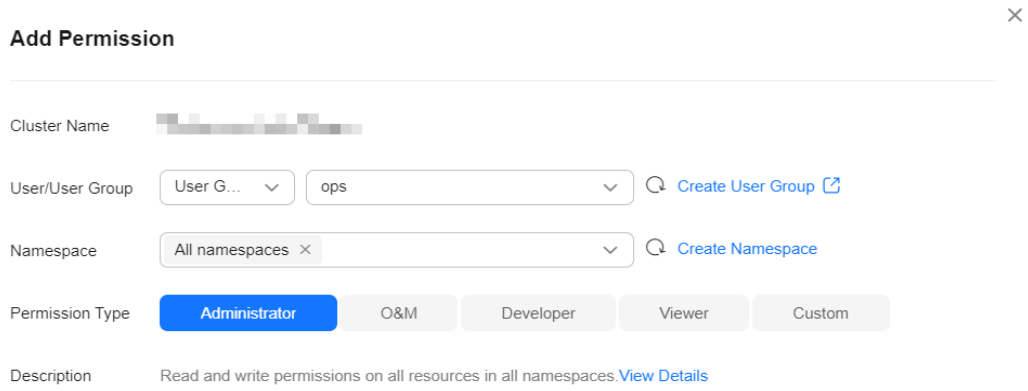
Try querying Deployments and Services in the namespace. The output shows that **user-example** does not have the required permissions. Try querying the pods in namespace kube-system. The output shows that **user-example** does not have the required permissions. This indicates that the IAM user **user-example** 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 3-345 Assigning cluster administrator permissions (cluster-admin)



In the following example kubectl output, a ClusterRoleBinding has been created and binds the cluster-admin role 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
```

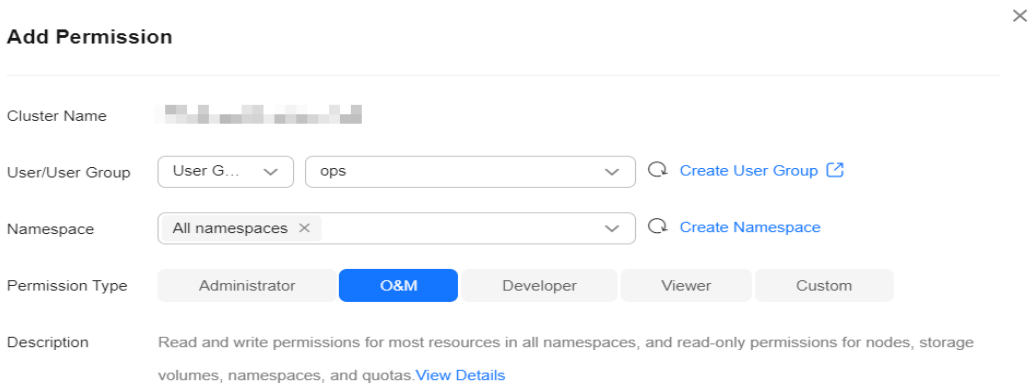
Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried, the permission configuration takes effect.

```
# 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 3-346 Assigning O&M permissions on all namespaces (admin)



In the following example kubectl output, a RoleBinding has been created and binds the admin role 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
```

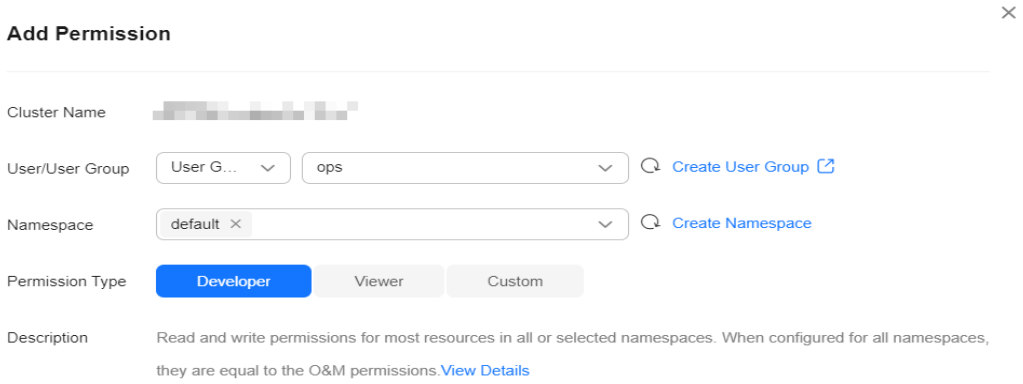
Connect to the cluster as an authorized user. If the PVs and StorageClasses can be queried but a namespace cannot be created, the permission configuration takes effect.

```
# 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 3-347 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
```

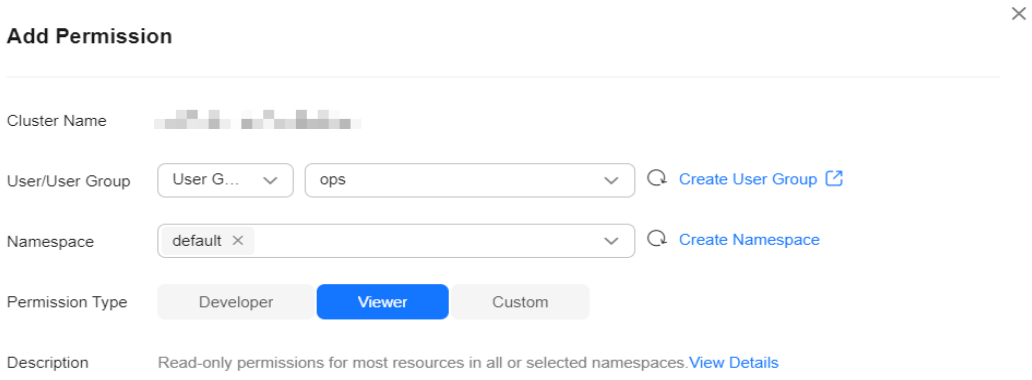
Connect to the cluster as an authorized user. In this example, you can create and obtain resources in the default namespace, but cannot query resources in the kube-system namespace or cluster resources.

```
# 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 3-348 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**.

3.16.4 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, node pools, 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, networking, storage, and namespaces. These permissions are assigned based on Kubernetes RBAC.

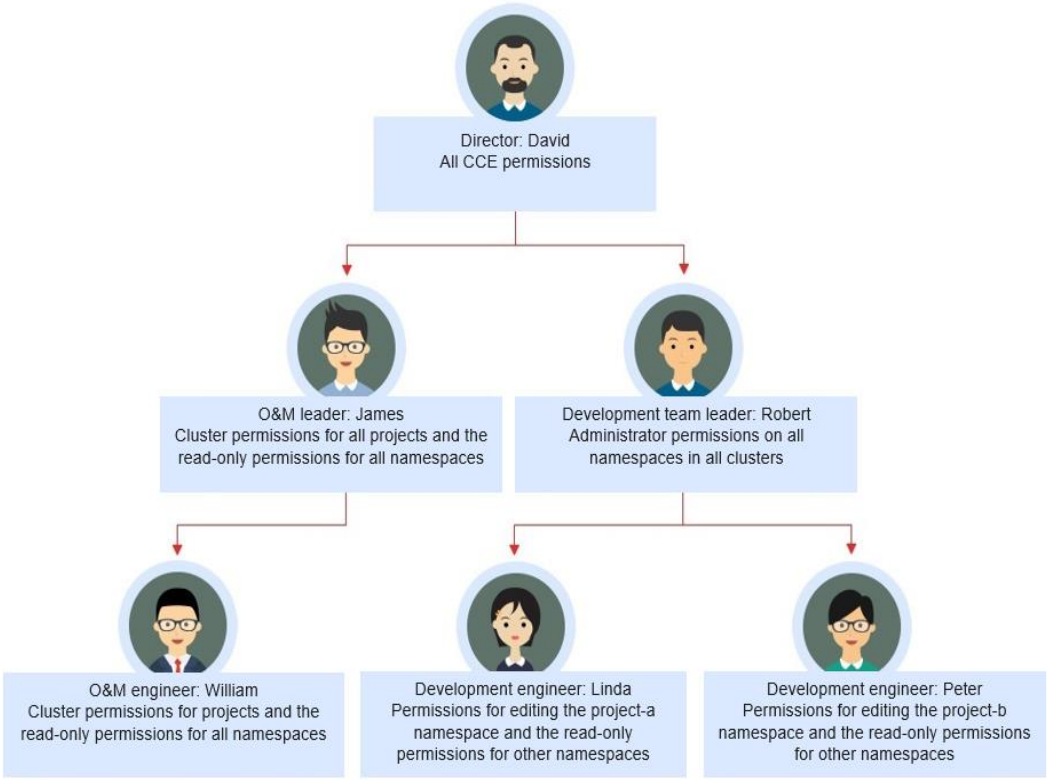
Cluster permissions and namespace permissions are independent of each other but must be used together. The permissions set for a user group apply to all users in the user group. When multiple permissions are added to a user or user group, they take effect at the same time (the union set is used).

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

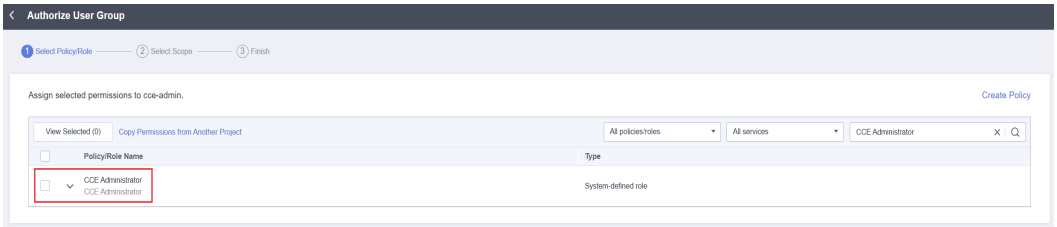
David is a department director of company X. To assign him all CCE permissions (both cluster and namespace permissions), create the **cce-admin** user group for David on the IAM console and assign the CCE Administrator role.

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 3-349 Assigning permissions to the user group to which David belongs

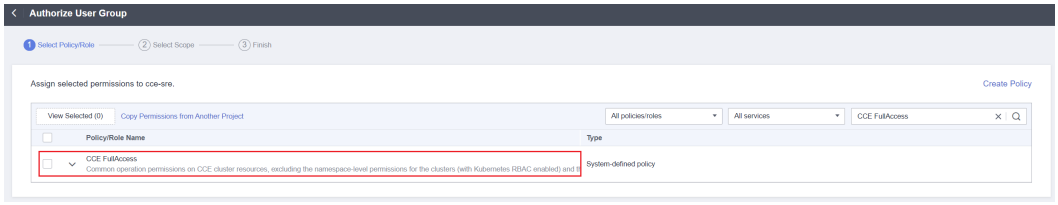


O&M Leader: James

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 3-350 Assigning permissions to the user group to which James belongs



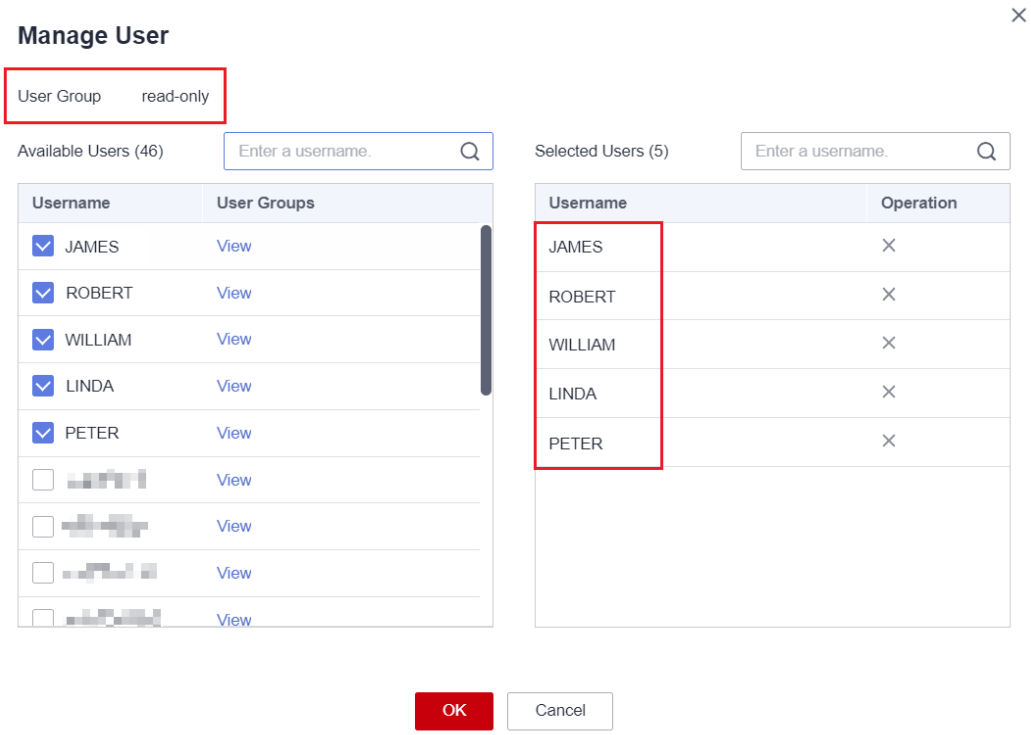
Assigning Read-only Permissions on All Clusters and Namespaces to All Team Leaders and Engineers

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.

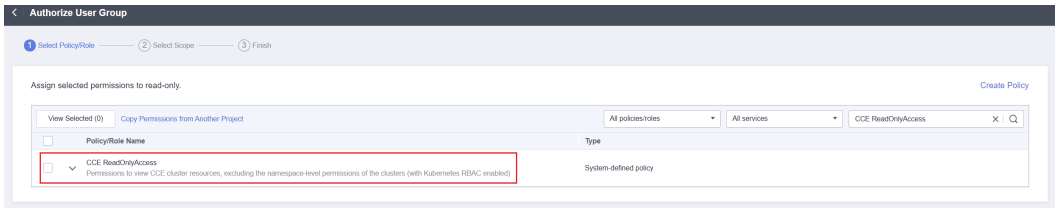
Users James, Robert, William, Linda, and Peter are added to the **read_only** user group.

Figure 3-351 Adding users to the read_only user group



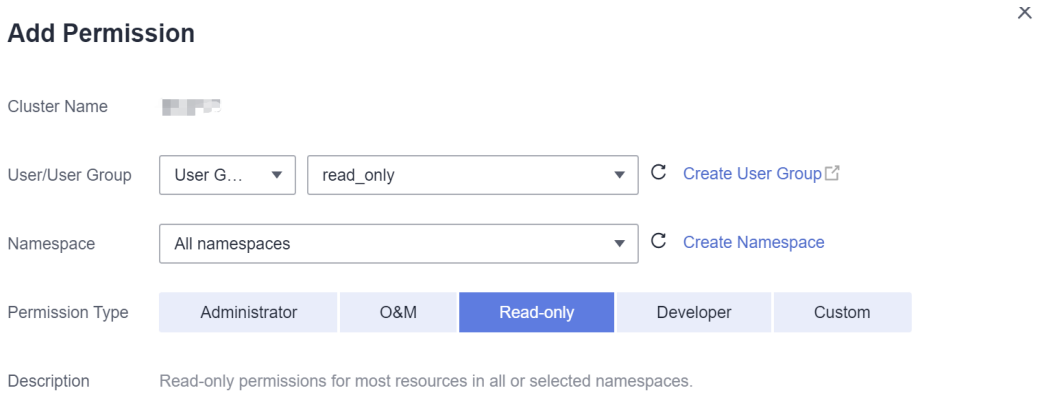
Assign the read-only permission on clusters to the user group **read_only**.

Figure 3-352 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 3-353 Assigning the read-only permission on namespaces to the user group



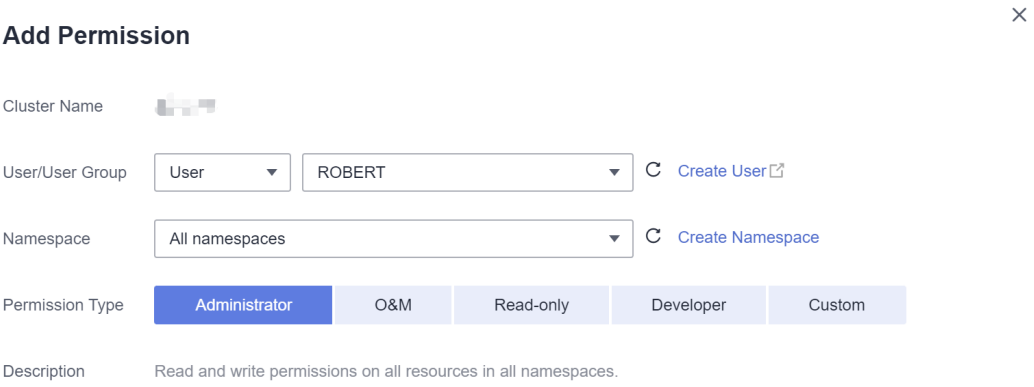
After the setting is complete, James has the cluster management permissions for all projects and the read-only permissions on all namespaces, and the Robert, William, Linda, and Peter have the read-only permission on all clusters and namespaces.

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.

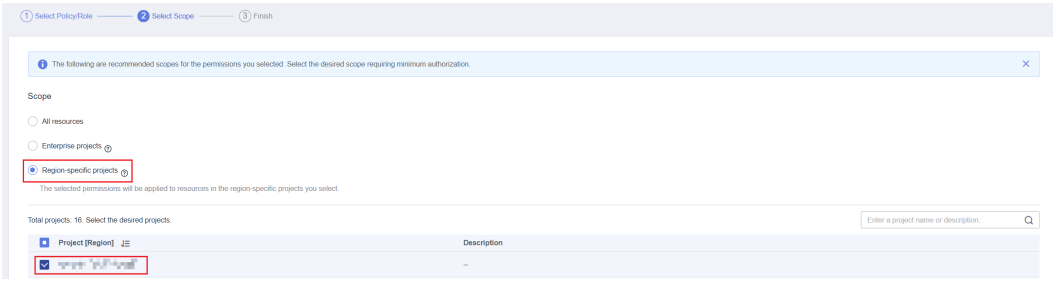
Figure 3-354 Assigning the administrator permissions on namespaces to Robert



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 3-355 Assigning the cluster management permissions for Beijing4 region to the user group to which WILLIAM belongs

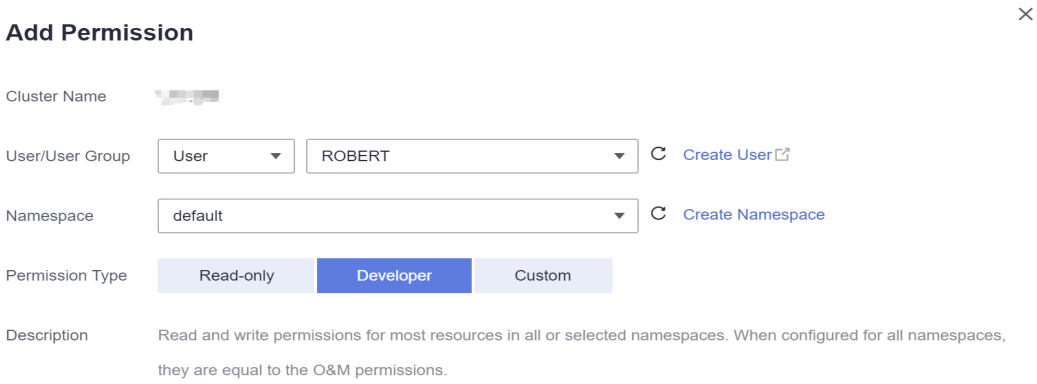


Now, William has the cluster management permissions for his region and the read-only permission on all namespaces.

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 3-356 Assigning the edit policy on namespaces



By now, all the required permissions are assigned to the department members.

3.16.5 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 **3.16.2 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 3-479** 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 3-479 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	/	/
Chart management	/	Cloud accounts and the IAM users with CCE Administrator assigned can use this function.
Add-ons	/	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	/	• 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	/	• 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	/	/
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.

3.16.6 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

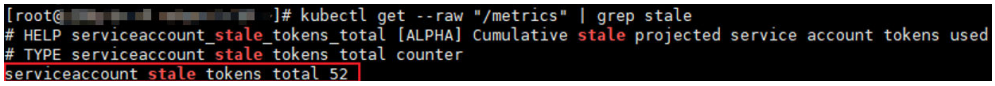
If you need a token that never expires, you can also **manually manage secrets for service accounts**. Although a permanent service account token can be manually created, you are advised to use a short-lived token by calling the **TokenRequest** API for higher security.

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 connect to 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.



3.16.7 System Entrustment Description

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**
- **cce_cluster_agency**

cce_admin_trust

The cce_admin_trust agency has the Tenant Administrator permissions. Tenant Administrator has the permissions on all cloud services except IAM, which are used to call the cloud services that CCE depends on. The delegation takes effect only in the current region.

To use CCE in multiple regions, request for cloud resource permissions in each region. You can go to the IAM console > and click **cce_admin_trust** to view the permissions of each region.

NOTE

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.

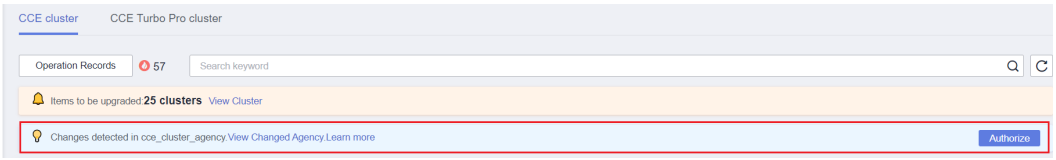
cce_cluster_agency

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.

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.

3.17 Storage Management: FlexVolume (Deprecated)

3.17.1 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 **3.8.1 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.

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 **3.14.3 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 3-480 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**.

3.17.2 How Do I Change 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 3-481 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 3-482 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 3-483 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 3-484 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 3-485 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 3-486 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 3-487 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 3-488 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.

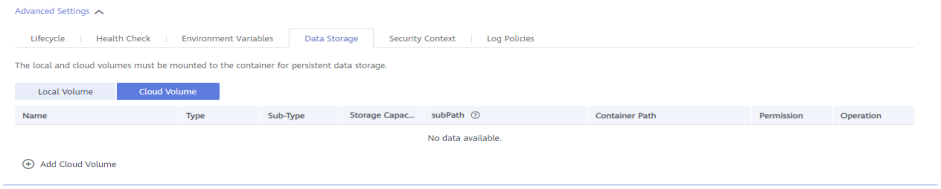
   kubectl create -f pv-example.yaml

   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

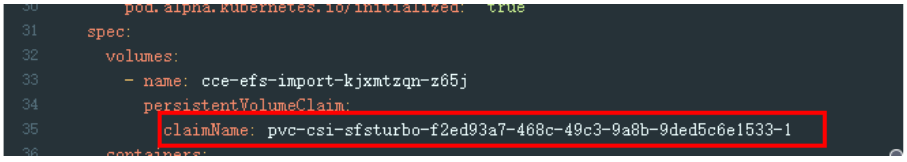
   kubectl create -f pvc-example.yaml

   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}}'

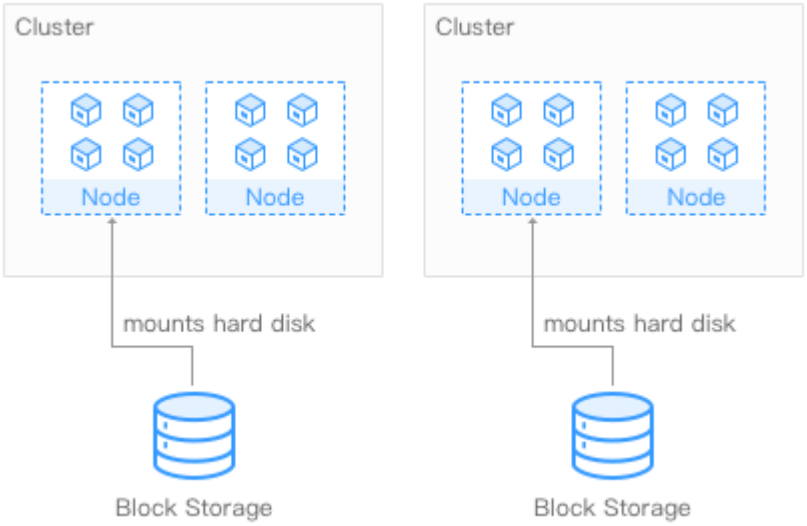
----End

3.17.3 Using EVS Disks as Storage Volumes

3.17.3.1 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 3-357 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.

3.17.3.2 (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 **3.2.3.1 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 3-489 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

3.17.3.3 (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 **3.2.3.1 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 3-490 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 3-491 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 3-492 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 3-493 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 3-494 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 3-495 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 3-496 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 3-358**.
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 3-359**.
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 3-358 Obtaining the disk ID

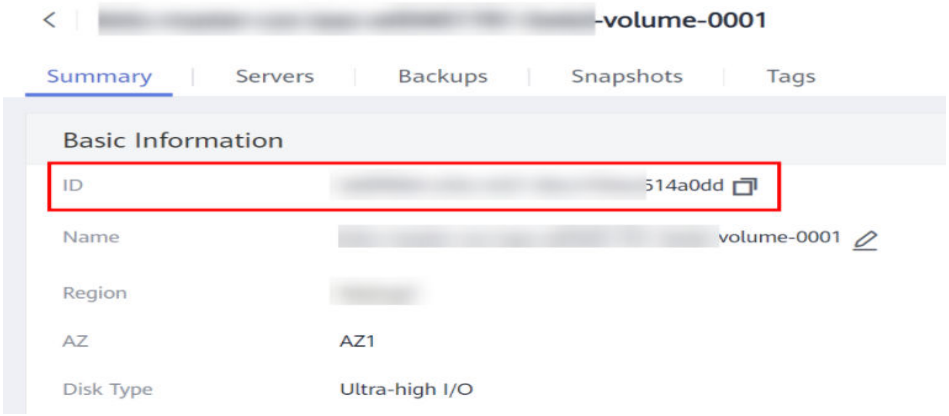
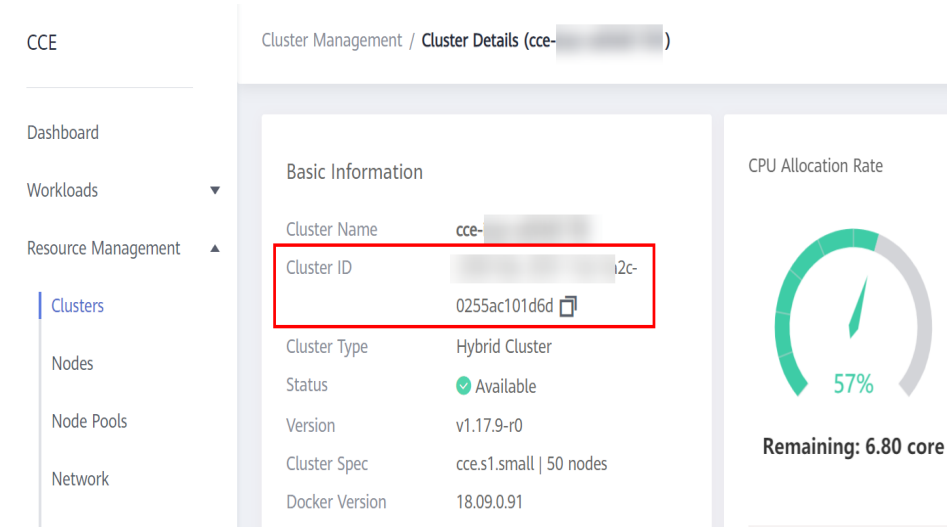


Figure 3-359 Obtaining the cluster ID



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

3.17.3.4 (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 connect to the cluster. For details, see **3.2.3.1 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 3-497 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 3-498 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 **3.5.2.2 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.

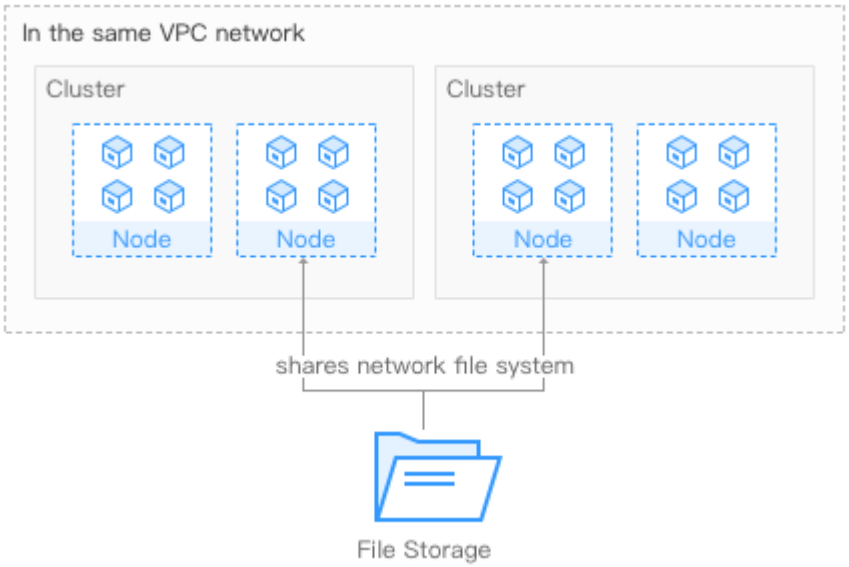
----End

3.17.4 Using SFS Turbo File Systems as Storage Volumes

3.17.4.1 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 3-360 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

3.17.4.2 (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 **3.2.3.1 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 3-499 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.
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**.

Parameter	Description
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 3-500 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

3.17.4.3 (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 **3.2.3.1 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 3-501 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

3.17.4.4 (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 **3.2.3.1 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 3-502 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.

Parameter	Description
serviceName	Service corresponding to the workload. For details about how to create a Service, see **3.5.2.2 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

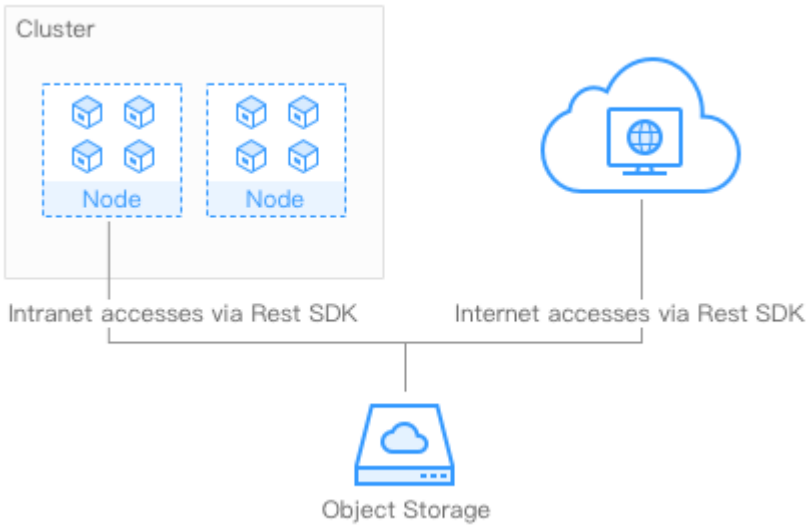
----End

3.17.5 Using OBS Buckets as Storage Volumes

3.17.5.1 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 3-361 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**.

3.17.5.2 (kubectl) Automatically Creating an OBS Volume

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 connect to the cluster. For details, see **3.2.3.1 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 3-503 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. **ReadWriteOnly** is not supported.

Parameter	Description
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

3.17.5.3 (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 **3.2.3.1 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 3-504 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).
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.

Parameter	Description
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 3-505 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 3-506 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 3-507 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

3.17.5.4 (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 **3.2.3.1 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 3-508 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 3-509 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 **3.5.2.2 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

3.17.5.5 (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 **3.17.5.2 (kubectl) Automatically Creating an OBS Volume** and obtain the PVC name.

Step 2 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 3-510 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 **3.5.2.2 Creating a StatefulSet**.
claimName	Name of an existing PVC.

Step 4 Create the StatefulSet.

kubectl create -f obs-statefulset-example.yaml

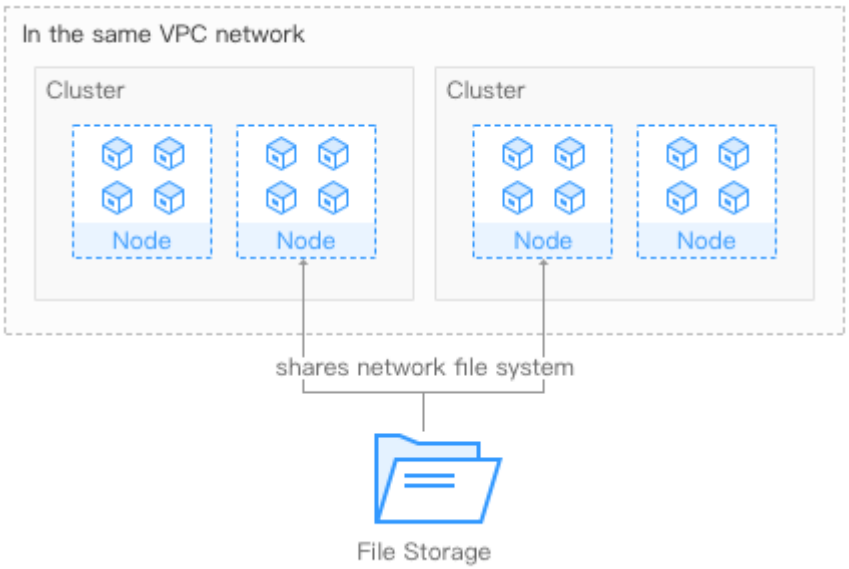
----End

3.17.6 Using SFS File Systems as Storage Volumes

3.17.6.1 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 3-362 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**.

3.17.6.2 (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 connect to the cluster. For details, see **3.2.3.1 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 3-511 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. **ReadWriteOnly** is not supported.
storage	Storage capacity in the unit of 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

3.17.6.3 (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 **3.2.3.1 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 3-512 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 3-363**.
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 3-513 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 3-514 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 3-363**.
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:**

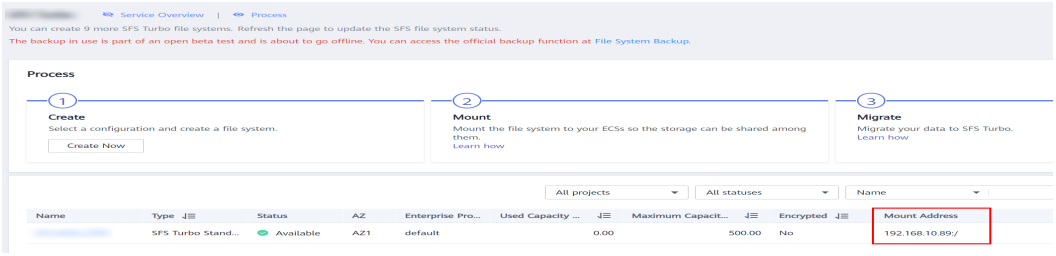
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
  volumeNamespace: default
```

Table 3-515 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.

Parameter	Description
volumeName	Name of the PV.

Figure 3-363 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

3.17.6.4 (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 **3.2.3.1 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 3-516 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 3-517 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 **3.5.2.2 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

3.17.6.5 (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 **3.17.6.2 (kubectl) Automatically Creating an SFS Volume** and record the volume name.

Step 2 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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 3-518 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 **3.5.2.2 Creating a StatefulSet**.
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

4 User Guide for Autopilot Clusters

4.1 What Is a CCE Autopilot Cluster?

Introduction

CCE Autopilot allows you to create serverless clusters that offer optimized Kubernetes compatibility and free you from O&M. After a CCE Autopilot cluster is created, you can deploy applications without purchasing nodes or maintaining the deployment, management, and security of nodes. You only need to focus on the implementation of application service logic, which greatly reduces your O&M costs and improves the reliability and scalability of applications.

Advantages

- Out-of-the-box usability: Applications can be directly deployed after Kubernetes clusters are created.
- Focus on applications: You can focus on building and deploying applications without managing the underlying infrastructure of the clusters.
- Security assurance: If there are available security patches, the clusters will be automatically upgraded as scheduled.
- Lower costs: Applications are created and charged on demand, and no idle resources are produced, reducing the costs.
- Node hosting: Worker nodes are hosted on CCE. You do not need to manually manage and upgrade these nodes.
- Elastic pre-warming: When an HPA policy is configured for a workload, CCE automatically pre-warms new nodes for new pods to easily handle traffic spikes.
- Appropriate scheduling: CCE Autopilot manages resource bin packing to improve resource utilization. You do not need to consider the number of pods running on each node.
- Resource allocation: If resource requests (such as CPU request and memory request) are not specified for workload deployment, CCE Autopilot automatically allocates the required resources for the pods.

- Simplified O&M: CCE Autopilot frees you from paying attention to node status. Cluster nodes can be automatically scaled in or out based on the load scale, which reduces the platform O&M burden.

Comparison Between CCE Autopilot, CCE Standard, and CCE Turbo

Table 4-1 Comparison between different types of CCE clusters

Dimension	CCE Autopilot	CCE Standard	CCE Turbo
Node management	Worker nodes are fully managed. CCE Autopilot takes care of node scaling and pre-warming.	You need to take care of the management and O&M of worker nodes.	You need to take care of the management and O&M of worker nodes.
Node OSs	There are dedicated OSs that use containerd as the container engine.	You can select an OS and container engine.	You can select an OS and container engine.
Node flavors	Node flavors are adaptive to the workload scale.	You can select the node flavors as needed.	You can select the node flavors as needed.
Node upgrade and maintenance	Nodes are upgraded and recovered automatically.	Nodes need to be reset for upgrade.	Nodes need to be reset for upgrade.
Container network model	Cloud native 2.0 network	• VPC network • Tunnel network	Cloud native 2.0 network
Network performance	The VPC network and container network are flattened into one for zero performance loss.	The container network is overlaid with the VPC network, causing performance loss.	The VPC network and container network are flattened into one for zero performance loss.

Dimension	CCE Autopilot	CCE Standard	CCE Turbo
Network isolation	Pods can be associated with security groups for isolation.	• Tunnel network model: network policies for communications within a cluster • VPC network model: isolation not supported	Pods can be associated with security groups for isolation.
Services	• ClusterIP • LoadBalancer	• ClusterIP • NodePort • LoadBalancer • DNAT	• ClusterIP • NodePort • LoadBalancer • DNAT
Ingresses	• ELB Ingress • Nginx Ingress	• ELB Ingress • Nginx Ingress	• ELB Ingress • Nginx Ingress

Dimension	CCE Autopilot	CCE Standard	CCE Turbo
Add-ons	The following configurable add-ons are available: • CoreDNS • Kubernetes Metrics Server • CCE Advanced HPA • Cloud Native Cluster Monitoring • Cloud Native Logging • NGINX Ingress Controller	The following configurable add-ons are available: • CoreDNS • CCE Container Storage (Everest) • CCE Node Problem Detector • Kubernetes Dashboard • CCE Cluster Autoscaler • Kubernetes Metrics Server • CCE Advanced HPA • CCE Cloud Bursting Engine for CCI • CCE AI Suite (NVIDIA GPU) • CCE AI Suite (Ascend NPU) • Volcano Scheduler • Nginx Ingress Controller • CCE Secrets Manager for DEW • CCE Network Metrics Exporter • NodeLocal DNSCache • Cloud Native Cluster Monitoring • Cloud Native Logging	The following configurable add-ons are available: • CoreDNS • CCE Container Storage (Everest) • CCE Node Problem Detector • Kubernetes Dashboard • CCE Cluster Autoscaler • Kubernetes Metrics Server • CCE Advanced HPA • CCE Cloud Bursting Engine for CCI • CCE AI Suite (NVIDIA GPU) • CCE AI Suite (Ascend NPU) • Volcano Scheduler • Nginx Ingress Controller • CCE Secrets Manager for DEW • CCE Network Metrics Exporter • NodeLocal DNSCache • Cloud Native Cluster Monitoring • Cloud Native Logging

Dimension	CCE Autopilot	CCE Standard	CCE Turbo
Local storage	emptyDir	• Local PV • hostPath • emptyDir	• Local PV • hostPath • emptyDir
Cloud storage	• SFS Turbo	• EVS • SFS • SFS Turbo • OBS	• EVS • SFS • SFS Turbo • OBS

Constraints

- Worker nodes in CCE Autopilot clusters are fully hosted. For this reason, some features that depend on the node features, such as hostPath and hostNetwork, are not supported.

Unavailable Feature	Description	Recommended Alternative Solution
DaemonSets	Deploys pods on each node.	Deploy multiple images in a pod using sidecars.
Setting hostPath in a pod	Mounts local files of a node to a container.	Use emptyDir or cloud storage of any type.
Setting hostNetwork in a pod	Maps a node port to a container port.	Use Services of the LoadBalancer type.
Use Services of the NodePort type.	Opens a node port to access containers.	Use Services of the LoadBalancer type.

- If a CCE Autopilot cluster is used, an emptyDir volume can have up to 20 GB of space.
- If a CCE Autopilot cluster is used, a maximum of 500 pods can be created. Add-on pods may occupy the pod quota. Plan the pod quota properly.
- If a CCE Autopilot cluster is used, workloads that use Arm images are not supported.

4.2 CCE Autopilot Cluster Billing

The price of a CCE Autopilot cluster consists of the following parts: cluster management, pods, and cloud resources (VPC endpoints and other cloud resources).

NOTE

The billed items marked with asterisks (*) are mandatory.

Table 4-2 Price of a CCE Autopilot cluster

Billed Item	Description	Billing Mode	Formula
*Cluster management	The expenses for managing the cluster **NOTE** If a cluster is frozen, workloads in the cluster will be in the pending state and will not be rescheduled until the cluster is unfrozen.	Pay-per-use	**Unit price of the cluster flavor x Required duration** For details about the unit price of cluster flavor, see **Unit Prices in Pay-per-Use Billing**.
*Pods	Pods are billed by flavor. **NOTICE** If a flavor is not supported, it will be automatically upgraded to a higher one. For example, if all containers in a pod require 2 vCPUs and 3 GiB of memory, the flavor is automatically upgraded to 2 vCPUs and 4 GiB of memory. **Flavor Description** lists the flavors supported by CCE Autopilot.	Pay-per-use	**Unit price of the pod flavor x Required duration** For details about the unit price of pod flavors, see **Unit Prices in Pay-per-Use Billing**.
*VPC endpoints	CCE Autopilot clusters connect to other cloud services such as SWR through VPC endpoints, which are billed separately based on the number of VPC endpoints you use.	Pay-per-use	**Unit price of the VPC endpoint x Required duration** **NOTE** • If a VPC endpoint connects to a VPC endpoint service other than DNS or OBS, you will be billed for how long you use this VPC endpoint. • If a VPC endpoint connects to DNS or OBS, you will not be billed for this VPC endpoint. See the pricing on the VPC Endpoint console.

Billed Item	Description	Billing Mode	Formula
Other cloud resources	Resources of cloud services used by a cluster such as Elastic Load Balance (ELB) are billed based on their pricing rules, no matter whether these resources are automatically created or manually added during cluster creation and use. Although cloud resources can be created on the CCE console, their billed items and bills are independent of those of CCE clusters.	Billing mode of each cloud service	For details, see **Price Calculator**.

Unit Prices in Pay-per-Use Billing

Table 4-3 Unit prices in pay-per-use billing

Region	Cluster Management	Pod
AP-Singapore	Cluster management is free of charge during the OBT.	• CPU: USD0.0000126/s per core • Memory: USD0.00000138/s per GiB

Flavor Description

CCE Autopilot automatically upgrades the flavors that are not supported to higher ones to ensure that the pods always have the resources required for running.

Table 4-4 Combinations of vCPUs and memory supported by CCE Autopilot

vCPU	Memory (GiB)
0.25 vCPUs	0.5, 1, and 2
0.5 vCPUs	1, 2, 3, and 4
1 vCPU	2, 3, 4, 5, 6, 7, and 8
2 vCPUs	4 to 16 (increment: 1 GiB)

vCPU	Memory (GiB)
4 vCPUs	8 to 32 (increment: 1 GiB)
8 vCPUs	8 to 64 (increment: 4 GiB)
16 vCPUs	16 to 128 (increment: 8 GiB)
32 vCPUs	32, 64, 128, and 256
48 vCPUs	96, 192, and 384
64 vCPUs	128, 256, and 512

4.3 Clusters

4.3.1 Kubernetes Version Release Notes

4.3.1.1 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 topic 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**
- **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 the instructions in **GitHub**.
- The Ceph FS in-tree volume 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 volume 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 the alpha stage are disabled by default, those in the beta stage are enabled by default, and those in the General Availability (GA) stage are always

enabled and cannot be disabled. The function of enabling or disabling the features in the 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 StorageClass is assigned to PersistentVolumeClaims (PVCs).

  The PersistentVolume (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.

  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 so 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 is shut down and that shutdown is 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. In this case, both pods are running concurrently.

    In Kubernetes 1.28, you can enable this feature by turning on the JobPodReplacementPolicy feature gate. With this feature enabled, you can set the **podReplacementPolicy** field under **spec** of a job to **Failed**. In this way, pods would only be replaced when they reach 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 that are owned by the job and are currently terminating.

  – Backoff limit per index

    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. In this case, the backoff limit per index configuration is useful.

    In Kubernetes 1.28, this feature can be enabled using the JobBackoffLimitPerIndex feature gate of a cluster. With this feature enabled, you can specify **.spec.backoffLimitPerIndex** 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 Common Expression Language (CEL) related features are improved.

  CEL related capabilities are enhanced.

  – CEL used to validate CustomResourceDefinitions (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 the following new functions:

    ▪ 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 when a job is created.
- 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 for jobs. 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 through 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 so 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 blocks 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 following feature gates are removed: DelegateFSGroupToCSIDriver, DevicePlugins, KubeletCredentialProviders, MixedProtocolLBService, ServiceInternalTrafficPolicy, ServiceIPStaticSubrange, and EndpointSliceTerminatingCondition.

References

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

4.3.1.2 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 topic describes the changes made in Kubernetes 1.27.

Indexes

New Features

- 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 toleration) 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 retrained 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**.

Deprecations and Removals

- In Kubernetes 1.27, the feature gates that are used for volume extension and in the General Availability (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. The 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 the **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.

References

For more details about the performance comparison and function evolution between Kubernetes 1.27 and other versions, see **Kubernetes v1.27 Release Notes**.

4.3.2 CCE Autopilot Cluster Version Release Notes

Indexes

v1.28

Table 4-5 Release notes for the v1.28 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Update	Enhancement	Vulnerability Fixing
v1.28.2-r0	v1.28.3	• Supported CronHPA policies. • Supported the security context configuration for pods.	-	Fixed some security issues.
v1.28.1-r10	v1.28.3	Supported cluster v1.28. For more information, see **Kubernetes 1.28 Release Notes**.	-	-

v1.27

Table 4-6 Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Update	Enhancement	Vulnerability Fixing
v1.27.4-r0	v1.27.4	• Supported CronHPA policies. • Supported the security context configuration for pods.	-	Fixed some security issues.
v1.27.3-r30	v1.27.4	-	Supported one-click configuration of monitoring alarms.	Fixed some security issues.
v1.27.3-r20	v1.27.4	• Supported the Nginx Ingress Controller add-on. • Supported the Cloud Native Cluster Monitoring and Cloud Native Logging add-ons to monitor application metrics and collect application logs. • Launched the application template market. • Supported CustomResourceDefinitions (CRDs). • Interconnected with CloudShell.	Optimized the function of creating a NAT gateway by default during cluster creation so that applications can access the public network.	Fixed some security issues.
v1.27.3-r10	v1.27.4	Supported cluster v1.27. For more information, see **Kubernetes 1.27 Release Notes**.	-	-

4.3.3 Buying a CCE Autopilot Cluster

A CCE Autopilot cluster runs on Cloud Container Instance (CCI) and provides native Kubernetes extended APIs, allowing you to run containers without creating or managing servers. You pay only for the resources used by your applications.

Constraints

- During the node creation, software packages are downloaded from OBS using the domain name. Use a private DNS server to resolve the OBS domain name, and configure the DNS server address of the subnet where the node resides with a **private DNS server address**. When you create a subnet, the private DNS server is used by default. If you change the subnet DNS settings, ensure that the DNS server in use can resolve the OBS domain name.
- After a cluster is created, the following items cannot be changed:
  - Cluster type
  - Network configuration of the cluster, such as the VPC, pod subnet, Service CIDR block, and kube-proxy (request forwarding) settings.
- When using a CCE Autopilot cluster, pay attention to the quotas of related resources. The following table lists the resources required by each cluster.

Service	Quota Item	Minimum Usage	Description	Quota Increase
CCE	Cluster	1	Maximum number of clusters that can be created by each account in a region: 50	Increase the quota on the **My Quotas** page.
VPC	VPC	1 per cluster	Maximum number of VPCs that can be created by each account in a region: 5	
	Subnet	1 per cluster	Maximum number of subnets that can be created by each account in a region: 50	
	Security group	2 per cluster	Maximum number of security groups that can be created by each account in a region: 100	
	Security group rules	10 per cluster	Maximum number of security groups rules that can be added by each account in a region: 1,000	
VPC Endpoint	Endpoint	3 per cluster	Maximum number of VPC endpoints that can be created by each account in a region: 50	
Domain Name Service (DNS)	Private zone	2 per cluster	Maximum number of private zones that can be created by each account in a region: 50	

Service	Quota Item	Minimum Usage	Description	Quota Increase
	Record set	6 per cluster	Maximum number of record sets that can be added by each account in a region: 500	

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** in the upper right corner.

----End

Step 2: Configure the Cluster

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

Basic Settings

Parameter	Description
Type	Select **CCE Autopilot cluster**. For details about the differences between three types of clusters, see **Comparison Between CCE Autopilot, CCE Standard, and CCE Turbo**.
Cluster Name	Enter a cluster name. Cluster names in the same account must be unique.
Cluster Version	Select the Kubernetes version used by the cluster.

Network Settings

Parameter	Description
VPC	Select the VPC that the cluster belongs to. If no VPC is available, create one first. Once a cluster is created, the VPC cannot be changed.
Pod Subnet	Select the subnet that the pods belong to. If no subnet is available, create one first. This subnet determines how many pods you can create in a cluster. After the cluster is created, you can add more subnets.

Parameter	Description
Service CIDR Block	Configure the Service CIDR block for containers in the same cluster to access each other. The CIDR block determines how many Services you can create. Once a cluster is created, the Service CIDR block cannot be changed.
Image Access	To ensure that the nodes in a cluster can pull images from SoftWare Repository for Container (SWR), existing endpoints in the selected VPC are used by default. If there are no endpoints in the VPC, new endpoints will be created for you to access SWR and OBS. VPC endpoints are not free. For details, see **VPC Endpoint Pricing**.
SNAT	If this option is enabled, a cluster can access the Internet through a NAT gateway. By default, an existing NAT gateway in the selected VPC is used. If there is no NAT gateway in the VPC, a new NAT gateway of the default specifications will be created, with an SNAT rule added and an EIP specified in the rule. NAT gateways are not free. For details, see **NAT Gateway Pricing**.

(Optional) Advanced Settings

Parameter	Description
Resource Tag	You can add resource tags to classify resources. You can create **predefined tags** on the Tag Management Service (TMS) console. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency. For details, see **Creating Predefined Tags**. Tag key requirements • A tag key can contain 1 to 128 single-byte characters and cannot be left empty. • A tag key cannot start with **_sys_**. Tags starting with **_sys_** are system tags. • A tag key can contain UTF-8 letters, digits, spaces, and the following special characters: _ . : / = + - @ Recommended regular expression: ^((?!_sys_)[\\p{L}\\p{Z}\\p{N}_.:\\/=+\\-@]*)$ Tag value requirements • A tag value can contain up to 255 characters. • A tag value can contain UTF-8 letters, digits, spaces, and the following special characters: _. : / = + - @ Recommended regular expression: ^([\p{L}\p{Z}\p{N}_.:\/=+\-@]*)$ • A tag value can be left empty or set to **null**. • The value of a predefined tag cannot left empty or set to **null**.
Description	You can enter up to 200 characters.

Step 3: Select Add-ons

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

Basic capabilities

Add-on	Description
CoreDNS	This add-on (**4.11.1 CoreDNS**) is installed by default. It provides DNS resolution for your cluster and can be used to access the cloud DNS servers.

Observability

Add-on	Description
Kubernetes Metrics Server	This add-on (**4.11.2 Kubernetes Metrics Server**) is installed by default. It collects resource usage metrics, such as the container CPU and memory usage, for the cluster.
Cloud Native Cluster Monitoring	(Optional) If selected, this add-on (**4.11.3 Cloud Native Cluster Monitoring**) will be automatically installed. It collects monitoring metrics for your cluster and reports the metrics to Application Operations Management (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.
Cloud Native Logging	(Optional) If selected, this add-on (**4.11.4 Cloud Native Logging**) 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 log collection every month. You pay only for log volume that exceeds the quota. For details, see **Price Calculator**. For details, see **4.7.2.1 Collecting Logs**.

Step 4: Configure Add-ons

Click **Next: Add-on Configuration**. The add-ons that are installed by default cannot be configured. After the cluster is created, you can go to the **Add-ons** page to modify their settings.

The table describes how to configure the optional add-ons.

Observability

Add-on	Description
Cloud Native Cluster Monitoring	Select an AOM instance for the add-on to report metrics. If no AOM instance is available, create one first. Basic metrics are free of charge. Custom metrics are billed based on the standard pricing of AOM. For details, see **AOM Pricing Details**.

Add-on	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}*. If log collection is disabled, choose **Logging** in the navigation pane of the cluster console after the cluster is created and enable this option. 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**. For details, see **4.7.2.1 Collecting Logs**.

Step 5: Confirm the Configuration

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 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 **4.3.4.1 Connecting to a Cluster Using kubectl**.
- Connect to multiple clusters using kubectl. For details, see **Connecting to Multiple Clusters Using kubectl**.

4.3.4 Connecting to a Cluster

4.3.4.1 Connecting to a Cluster Using kubectl

Scenario

You can use kubectl to connect a cluster.

Permissions

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

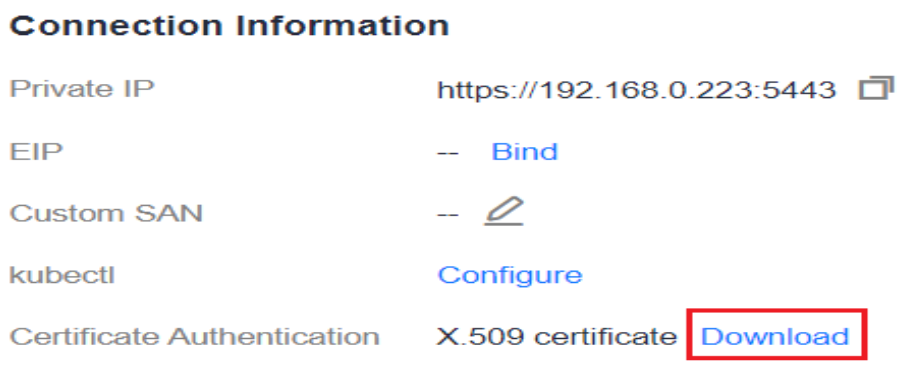
CCE determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a **kubeconfig.json** file vary from user to user.

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

Using kubectl

kubectl is a Kubernetes command line tool for you to connect to a Kubernetes cluster from a client. You can log in to the CCE console, click the name of the cluster to be connected, and view the access address and connection steps on the cluster details page.

Figure 4-1 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 steps are as follows:

Step 1 Prepare the environment.

You need to prepare a VM that is in the same VPC as the cluster and bind an EIP to the VM for downloading kubectl.

Step 2 Download kubectl.

You can run the **kubectl version** command to check whether kubectl has been installed. If kubectl has been installed, skip this step.

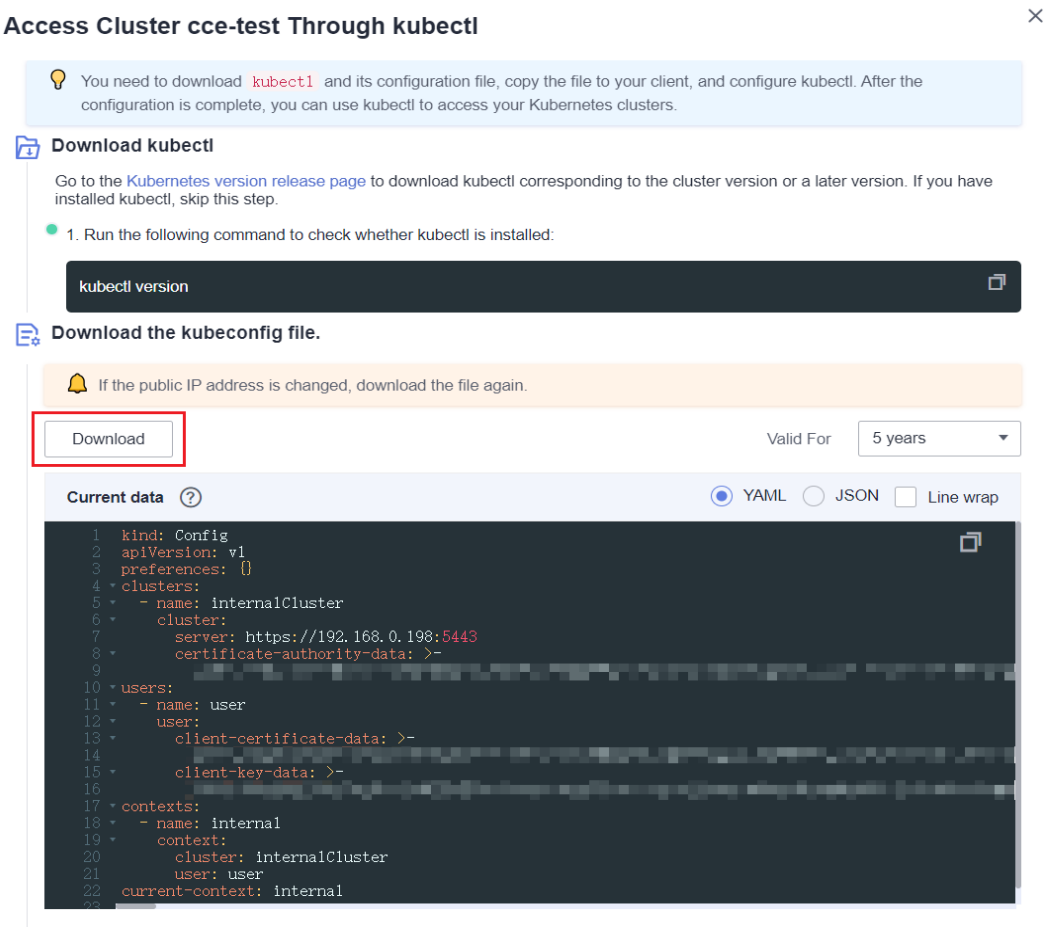
The Linux environment is used as an example to describe how to install and configure kubectl. For details, see **Installing kubectl**.

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 number. Replace it as required.
2. Install kubectl.
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

Step 3 Obtain the kubectl configuration file.

In the **Connection Information** pane on the cluster details page, click **Configure** next to **kubectl**. On the window displayed, download the configuration file.

Figure 4-2 Downloading the configuration file



NOTE

- The kubectl configuration file **kubeconfig.json** is used for cluster authentication. If the file is leaked, your cluster may be attacked.
- For IAM users, the Kubernetes permissions specified in the configuration file are the same as those assigned 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 4 Configure kubectl.

A Linux OS is used as an example to describe how to configure kubectl.

1. Log in to your client and copy **kubeconfig.json** downloaded in **Step 3** 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. Use kubectl for access.
   ```
   kubectl config use-context internal
   ```

----End

Troubleshooting

- **Error from server Forbidden**

  When you use kubectl to create or query Kubernetes resources, the following information is displayed:

  ```
  # 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"
  ```

  This is because the user does not have permission to operate the Kubernetes resources. For details about how to grant permissions to the user, 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 information is displayed:

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

  This is because cluster authentication is not configured for the kubectl client. For details, see **Step 4**.

Related Operations

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

4.3.4.2 Connecting to a Cluster Using CloudShell

Scenario

You can use CloudShell to connect a cluster.

Permissions

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

Using CloudShell

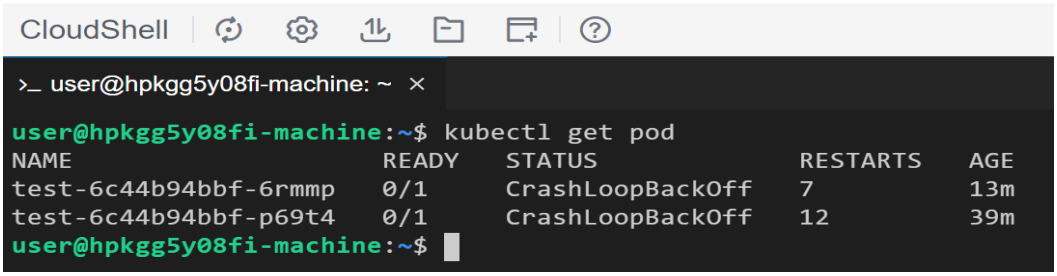
CloudShell is a web shell used to manage and maintain cloud resources. CCE allows you to use CloudShell to connect to clusters and use kubectl in CloudShell to access clusters (by clicking the command line tool icon in **Figure 4-3**).

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 VPC Endpoint. 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 corresponding 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 using 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.
- CloudShell is only available in the following regions: CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and CN North-Ulanqab1.
- CloudShell cannot be used across accounts or in or sub-projects.

Figure 4-3 CloudShell

Figure 4-4 Using kubectl in CloudShell



4.3.4.3 Connecting to a Cluster Using an X.509 Certificate

Scenario

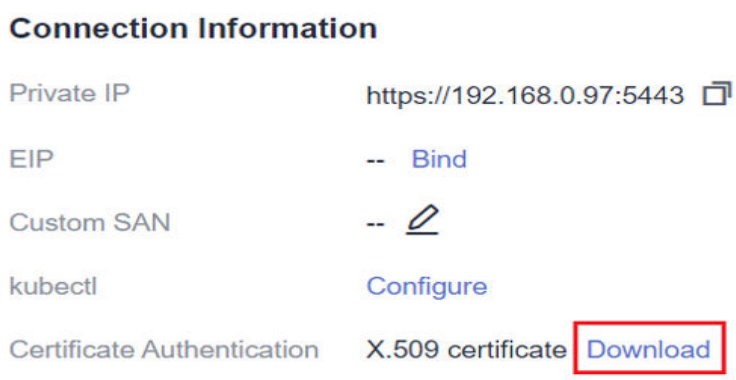
This section describes how to obtain the cluster certificate from the console and use it to access Kubernetes clusters.

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 4-5 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. In the following information, *192.168.0.18:5443* indicates the IP address of the API server in the cluster.

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

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

----End

4.3.5 Managing a Cluster

4.3.5.1 Deleting a Cluster

Precautions

- 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 data has been backed up or migrated. Deleted data cannot be restored.

  The following resources will not be deleted:

  – ELB load balancers associated with Services and ingresses (Only the automatically created load balancers will be deleted.)

  - Manually created cloud storage resources associated with PVs or imported cloud storage resources (Only the cloud storage resources automatically created by PVCs will be deleted.)
  - Resources created in the VPC, such as VPC endpoints, NAT gateways, and EIPs specified in SNAT rules
- If you delete a cluster that is not in the **Running** state (for example, **Froze**n or **Unavailable**), associated resources, such as storage and networking resources, will remain.

Deleting a Cluster

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

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

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

- 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.
- Delete network resources such as load balancers in a cluster. (Only automatically created load balancers will be deleted).

By default, the resources (such as VPC endpoints, NAT gateways, and EIPs specified in the SNAT rules) created in the VPC of the cluster are retained. Ensure that the resources are not reused by other clusters or Services and delete them on the network console.

Step 4 Click **Yes** to start deleting the cluster.

The delete operation takes 1 to 3 minutes to complete.

----End

4.4 Workloads

4.4.1 Creating a Workload

4.4.1.1 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

- A cluster is available. For details about how to create a cluster, see **4.3.3 Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS**.
- A Service of the LoadBalancer type has been created if the workload needs to be accessed by external networks.

  NOTE

  If a pod has multiple containers, the ports used by the containers cannot conflict with each other. If there is a conflict, the Deployment will fail to be created.

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 on the left, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **Deployment**.
- **Workload Name**: Enter a name for 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 a namespace. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **4.8.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.

Container Settings

- Container Information

  A pod can have more than one container. You can click **Add Container** on the right to configure multiple containers.

  - **Basic Info**: Configure basic information about each container.

Parameter	Description
Container Name	Enter a name for 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. To use a third-party image, see **4.4.2.2 Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	CPU limit, which is the maximum CPU available for the container to prevent excessive resource usage.
Memory Quota	Memory limit, which is the maximum memory available for the container. When the container's memory usage exceeds the memory limit, the container will be terminated.
(Optional) Init Container	Whether the container will be used 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 **4.4.2.3 Configuring the Container Lifecycle**.
  - (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **4.4.2.4 Setting Health Check for a Container**.
  - (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 **4.4.2.5 Configuring Environment Variables**.
  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and the ways for mounting the volumes vary with the storage type. For details, see **4.6 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.
- **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. For details about **default-secret**, see **default-secret**.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards the 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 **4.5.1 Service**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and upgrade parameters of the workload. **Rolling upgrade** and **Replace upgrade** are supported. For details, see **4.4.2.6 Configuring the Workload Upgrade Policy**.
- **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 **4.4.2.7 Labels and Annotations**.
- **DNS**: Configure a DNS policy for the workload. For details, see **DNS Configuration**.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

Nginx is used as an example here to describe how to create a workload using kubectl.

NOTICE

Node affinity and anti-affinity are not available for CCE Autopilot clusters. When you use kubectl to create a workload, do not configure the **affinity** field to prevent pod creation failures.

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name. 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 these parameters, see **Table 4-7**.

Table 4-7 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
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

Parameter	Description	Mandatory/ 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
imagePullSecrets	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 "nginx" created
```

Step 4 Query 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, add the corresponding Service. For details, see **4.5.1 Service**.

----End

Documentation

- **Upgrading Pods Without Interrupting Services**
- **Configuring Domain Name Resolution for CCE Containers**

4.4.1.2 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, mount HA storage volumes provided by CCE to the container.

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 **4.5.1.3 Headless Service**.

Prerequisites

- A cluster is available. For details about how to create a cluster, see **4.3.3 Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS**.
- A Service of the LoadBalancer type has been created if the workload needs to be accessed by external networks.

  NOTE

  If a pod has multiple containers, the ports used by the containers cannot conflict with each other. If there is a conflict, the Deployment will fail to be created.

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 on the left, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **StatefulSet**.
- **Workload Name**: Enter a name for 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 a namespace. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **4.8.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.

Container Settings

- Container Information

  A pod can have more than one container. You can click **Add Container** on the right to configure multiple containers.

  – **Basic Info**: Configure basic information about each container.

Parameter	Description
Container Name	Enter a name for 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. To use a third-party image, see **4.4.2.2 Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	CPU limit, which is the maximum CPU available for the container to prevent excessive resource usage.
Memory Quota	Memory limit, which is the maximum memory available for the container. When the container's memory usage exceeds the memory limit, the container will be terminated.

Parameter	Description
(Optional) Init Container	Whether the container will be used 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 **4.4.2.3 Configuring the Container Lifecycle**.
  - (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **4.4.2.4 Setting Health Check for a Container**.
  - (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 **4.4.2.5 Configuring Environment Variables**.
  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and the ways for mounting the volumes vary with the storage type.

    NOTE

    - StatefulSets allow you to mount storage volumes dynamically.

      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.

- **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. For details about **default-secret**, see **default-secret**.

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 **4.5.1.3 Headless Service**.

(Optional) Service Settings

A Service provides external access for pods. With a static IP address, a Service forwards the 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 **4.5.1 Service**.

(Optional) Advanced Settings

- **Upgrade**: Specify the upgrade mode and upgrade parameters of the workload. **Rolling upgrade** and **Replace upgrade** are supported. For details, see **4.4.2.6 Configuring the Workload Upgrade Policy**.
- **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.
- **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 **4.4.2.7 Labels and Annotations**.
- **DNS**: Configure a DNS policy for the workload. For details, see **DNS Configuration**.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

NOTICE

Node affinity and anti-affinity are not available for CCE Autopilot clusters. When you use kubectl to create a workload, do not configure the **affinity** field to prevent pod creation failures.

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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
      imagePullSecrets:
        - name: default-secret
      dnsPolicy: ClusterFirst
  serviceName: nginx-svc
  replicas: 2
  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, set the corresponding workload access type. For details, see **4.5.1 Service**.

----End

4.4.1.3 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 the job is completed based on user configurations. The success flag varies depending on 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.
- Parallel jobs in a work queue: Jobs are considered completed based on the global success confirmed by the application.

Prerequisites

- A cluster is available. For details about how to create a cluster, see **4.3.3 Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS**.
- A Service of the LoadBalancer type has been created if the workload needs to be accessed by external networks.

  NOTE

  If a pod has multiple containers, the ports used by the containers cannot conflict with each other. If there is a conflict, the Deployment will fail to be created.

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 on the left, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **Job**.
- **Workload Name**: Enter a name for 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 a namespace. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **4.8.1 Creating a Namespace**.
- **Pods**: Enter the number of pods of the workload.

Container Settings

- Container Information

  A pod can have more than one container. You can click **Add Container** on the right to configure multiple containers.

  - **Basic Info**: Configure basic information about each container.

Parameter	Description
Container Name	Enter a name for 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. To use a third-party image, see **4.4.2.2 Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	CPU limit, which is the maximum CPU available for the container to prevent excessive resource usage.
Memory Quota	Memory limit, which is the maximum memory available for the container. When the container's memory usage exceeds the memory limit, the container will be terminated.
(Optional) Init Container	Whether the container will be used 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 **4.4.2.3 Configuring the Container Lifecycle**.

  - (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 **4.4.2.5 Configuring Environment Variables**.
  - (Optional) **Data Storage**: Mount local storage or cloud storage to the container. The application scenarios and the ways for mounting the volumes vary with the storage type. For details, see **4.6 Storage**.
- **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. For details about **default-secret**, see **default-secret**.

(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 **4.4.2.7 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.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

NOTICE

Node affinity and anti-affinity are not available for CCE Autopilot clusters. When you use kubectl to create a workload, do not configure the **affinity** field to prevent pod creation failures.

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, all 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 4-8 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 completed 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 completed when the number of successful pods reaches the value of **.spec.completions**.	>1	>1
Parallel jobs in 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**.	Left 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       # 50 pods need to be run to finish a job. In this example, π is printed for 50 times.
  parallelism: 5        # 5 pods are run in parallel.
```



```
backoffLimit: 5        # The maximum number of retry times is 5.
template:
  spec:
    containers:
    - name: pi
      image: perl
      command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
    restartPolicy: Never
    imagePullSecrets:
      - name: default-secret
```

Description

- **apiVersion: batch/v1** indicates the version of the current job.
- **kind: Job** indicates that the current resource is a job.
- **restartPolicy: Never** indicates the current restart policy. For jobs, this parameter can only be set to **Never** or **OnFailure**. For other controllers (for example, Deployments), you can set this parameter to **Always**.

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 Query 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

```
# kubectl logs myjob-29qlw
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892352421995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019893809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151557485724245415069595082953311686172785588907509838175463746493931925506040092770167113900984882401285836160356370766010471018194295559619894676783744944825537977472684710404753464620804668425906949129331367702898915210
```


475216205696602405803815019351125338243003558764024749647326391419927260426992279678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295532116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321721477235014144197356854816136115735255213347574184946843852332390739414333454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438382796797668145410095388378636095068006422512520511739298489608412848862694560424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759015

----End

Related Operations

After a one-off job is created, you can perform operations listed in **Table 4-9**.

Table 4-9 Other operations

Operation	Description
Deleting a job	1. Select the job to be deleted and click **More** > **Delete** in the **Operation** column. 2. Click **Yes**. Deleted jobs cannot be recovered.

4.4.1.4 Creating a CronJob

Scenario

A CronJob runs on a repeating schedule. You can perform time synchronization for all active nodes at a fixed time point.

Similar to Linux crontab, a CronJob runs periodically at the specified time and has the following characteristics:

- A CronJob runs only once at the specified time.
- A CronJob runs periodically at the specified time.

A CronJob is typically used to:

- Schedule jobs at the specified time.
- Create jobs to run periodically, for example, database backup and email sending.

Prerequisites

- A cluster is available. For details about how to create a cluster, see **4.3.3 Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS**.
- A Service of the LoadBalancer type has been created if the workload needs to be accessed by external networks.

NOTE

If a pod has multiple containers, the ports used by the containers cannot conflict with each other. If there is a conflict, the Deployment will fail to be created.

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 on the left, and click the **Create Workload** in the upper right corner.

Step 3 Set basic information about the workload.

Basic Info

- **Workload Type**: Select **CronJob**.
- **Workload Name**: Enter a name for 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 a namespace. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **4.8.1 Creating a Namespace**.

Container Settings

- Container Information

  A pod can have more than one container. You can click **Add Container** on the right to configure multiple containers.

  – **Basic Info**: Configure basic information about each container.

Parameter	Description
Container Name	Enter a name for 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. To use a third-party image, see **4.4.2.2 Using Third-Party Images**.
Image Tag	Select the image tag to be deployed.
CPU Quota	CPU limit, which is the maximum CPU available for the container to prevent excessive resource usage.

Parameter	Description
Memory Quota	Memory limit, which is the maximum memory available for the container. When the container's memory usage exceeds the memory limit, the container will be terminated.
(Optional) Init Container	Whether the container will be used 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 **4.4.2.3 Configuring the Container Lifecycle**.
  - (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 **4.4.2.5 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. For details about **default-secret**, see **default-secret**.

Schedule

- **Concurrency Policy**: There are three options:
  - **Forbid**: A new job cannot be created before the previous job is completed.
  - **Allow**: The CronJob 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**: Specify when a new job is executed. Policy settings in YAML are implemented using cron expressions.
  - A job is executed at a fixed interval. The unit can be minute, hour, day, or month. For example, if a 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 **...**.
  - A job is executed at by month. For example, if a 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 **...**.

  - A job is executed by week. For example, if a 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 job is executed at a fixed time (by month) and the date in a month does not exist, the 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 job that is executed by hour as an example. As **/2, /3, /4, /6, /8, and /12** can exactly divide 24 hours, an accurate period can be represented. If another period is used, the last period will be reset at the beginning of a new day. For example, if the cron expression is *** */12 * * ***, the execution time is **00:00:00** and **12:00:00** every day. If the cron expression is *** */13 * * ***, the execution time is **00:00:00** and **13:00:00** every day. At 00:00 on the next day, the execution time is updated even if the period does not reach 13 hours.

- **Job Records**: You can set the number of jobs that are successfully executed or fail to be executed. If the values are set to **0**, none of the jobs will be kept 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 **4.4.2.7 Labels and Annotations**.

Step 4 Click **Create Workload** in the lower right corner.

----End

Using kubectl

NOTICE

Node affinity and anti-affinity are not available for CCE Autopilot clusters. When you use kubectl to create a workload, do not configure the **affinity** field to prevent pod creation failures.

A CronJob 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 CronJob, which is saved in the **cronjob.yaml** file.

NOTE

In clusters of v1.21 or later, the **apiVersion** value of CronJobs is **batch/v1**.

In clusters earlier than v1.21, the **apiVersion** value of CronJobs 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 a CronJob.

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

If a CronJob is deleted, the related jobs and pods are deleted accordingly.

----End

Related Operations

After a CronJob is created, you can perform operations described in **Table 4-10**.

Table 4-10 Other operations

Operation	Description
Editing a YAML file	Click **More** > **Edit YAML** next to the CronJob name to edit the YAML file of the current job.
Stopping a CronJob	1. Select the CronJob 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 CronJobs cannot be recovered.

4.4.2 Configuring a Container

4.4.2.1 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.

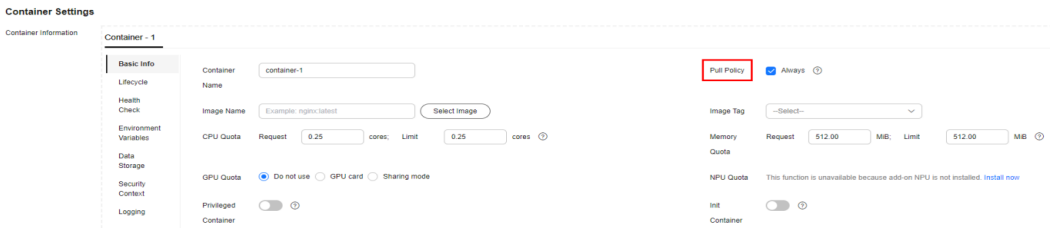
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
```



```
- 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 4-6 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.

4.4.2.2 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 only be accessed after authentication (using your username and password). CCE uses the secret-based authentication to pull images. You need to create a secret for an image repository before pulling images from the repository.

Prerequisites

CCE Autopilot can access the network where the private repository is located. There are two options for this:

- Access over a private network: Ensure that the VPC of the private repository is the same as the VPC where the cluster resides.
- Access over Direct Connect or VPN: Connect the private repository network to the VPC where the cluster resides through **Direct Connect** or **VPN**.

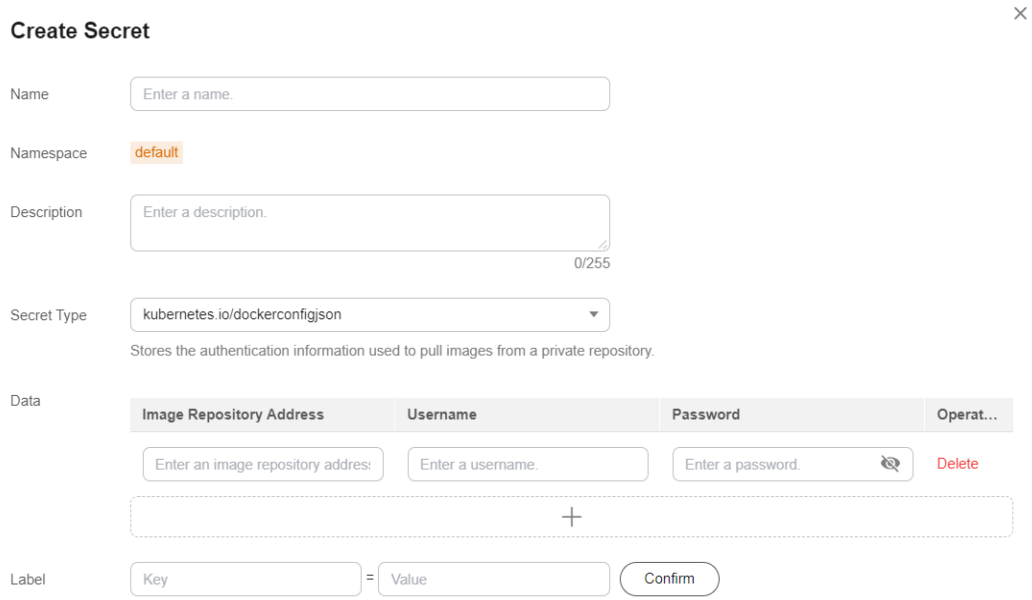
Using the Console

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 on the left, choose **ConfigMaps and Secrets**. On the **Secrets** tab, click **Create Secret** in the upper right corner. Set **Secret Type** to **kubernetes.io/dockerconfigjson**. For details, see **4.9.3 Creating a Secret**.

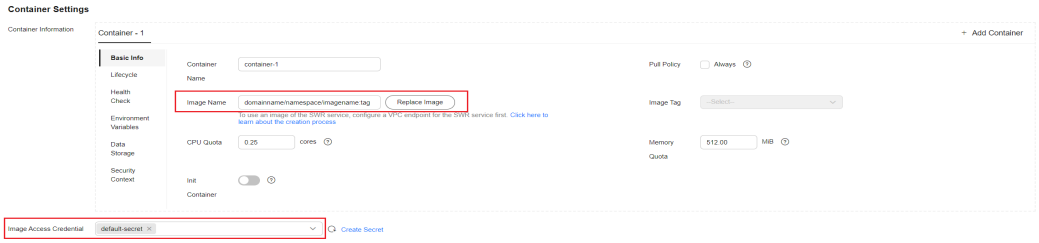
Enter the username and password used to access the third-party image repository.

Figure 4-7 Creating a secret



Step 2 When creating a workload, you can 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 4-8 Private image path



Step 3 Set other parameters and click **Create Workload**.

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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

4.4.2.3 Configuring the Container Lifecycle

Scenario

CCE provides hooks for container lifecycle management. For example, you can use a hook to make a container perform a specific operation before stopping.

CCE provides the following hooks:

- **Startup Command**: executed to start a container. For details, see **Startup Command**.
- **Post-Start**: executed immediately after a container is started. For details, see **Post-Start**.
- **Pre-Stop**: executed before a container is stopped. This hook helps you ensure that the services running on the pods can be completed in advance in case of pod upgrade or deletion. For details, see **Pre-Stop**.

Startup Command

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 (Docker instructions **ENTRYPOINT** and **CMD**) provided during image creation.

If the commands and arguments used to run a container are set during workload creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows.

Table 4-11 Commands and arguments used to run a container

Image ENTRYPOINT	Image CMD	Command to Run a Container	Arguments 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 On the **Startup Command** tab, enter a command and arguments.

Table 4-12 Container startup command

Parameter	Description
Command	Enter an executable command, for example, **/run/server**. If there are multiple executable commands, write them on different lines. **NOTE** If there are multiple commands, it is recommended that you run **/bin/sh** or other **shell** commands and use other commands as parameters.
Args	Enter an argument for the command, for example, **--port=8080**. If there are multiple arguments, write them on different lines.

----End

Post-Start

Step 1 Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

Step 2 On the **Post-Start** tab, configure the parameters.

Table 4-13 Post-Start parameters

Parameter	Description

CLI	The tool for running the commands 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 them into a script for execution. **Commands that are executed in the backend 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.

----End

Pre-Stop

Step 1 Log in to the CCE console. When creating a workload, configure container information and select **Lifecycle**.

Step 2 On the **Pre-Stop** tab, configure the parameters.

Table 4-14 Pre-Stop parameters

Parameter	Description

CLI	The tool for running the commands 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 them into a script for execution.
	Example command:
	`exec:`
	`  command:`
	`  - /uninstall.sh`
	`  - uninstall_agent`
	Enter **/uninstall uninstall_agent** in the script. This command indicates that the **uninstall.sh** script will be executed before the container completes its execution and stops running.

----End

YAML Example

Nginx is used as an example to describe how to set the container lifecycle.

In the following configuration file, there is a Post-Start command (**install.sh**) in the **/bin/bash** directory and a PreStop command (**uninstall.sh**).

```
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
```


4.4.2.4 Setting Health Check for a Container

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, the application process is running, but 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.

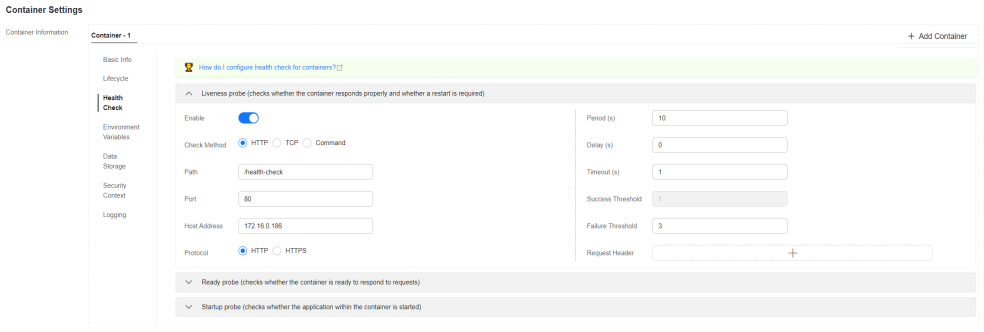
Check Method

- **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 4-9 HTTP request-based check

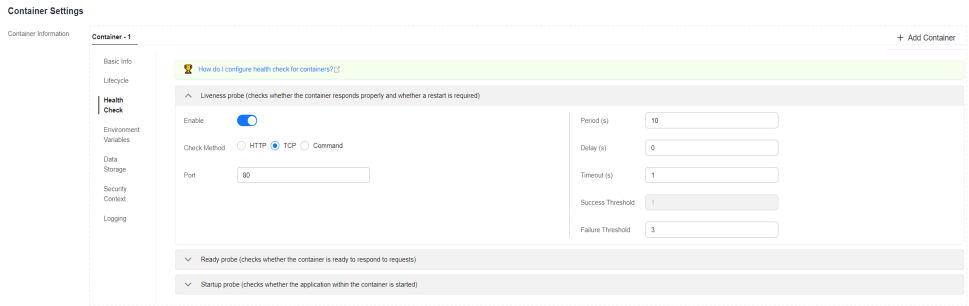


- **TCP port**

  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 4-10 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.

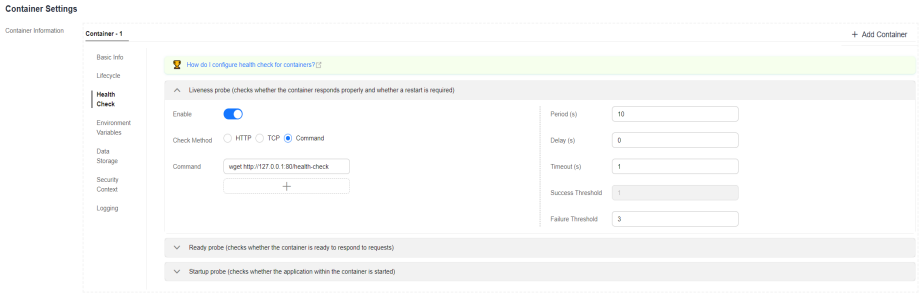
  The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

  - 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 4-11 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. The reason is that the cluster is not in the terminal environment when executing programs in a container.

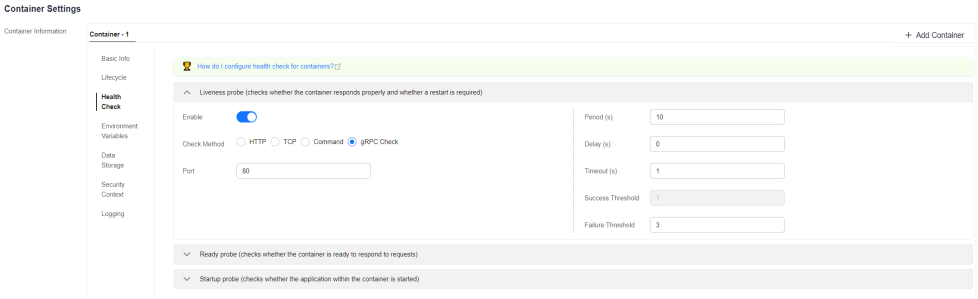
- **gRPC Check**

  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

- 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 4-12 gRPC check



Common Parameters

Table 4-15 Common parameter description

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**. The minimum value is **1**.

YAML Example

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
```



```
livenessProbe:
  httpGet:
    path: /healthz
    port: 80
    httpHeaders:
    - name: Custom-Header
      value: Awesome
  initialDelaySeconds: 3
  periodSeconds: 3
readinessProbe:
  exec:
    command:
      - cat
      - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
startupProbe:
  httpGet:
    path: /healthz
    port: 80
  failureThreshold: 30
  periodSeconds: 10
```

4.4.2.5 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 key**: Import all keys in a ConfigMap as environment variables.
- **Added from ConfigMap**: Import a key in a ConfigMap as the value of an environment variable. As shown in **Figure 4-13**, 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 keys 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 4-13**, 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 4-13**, 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 4-13**, 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

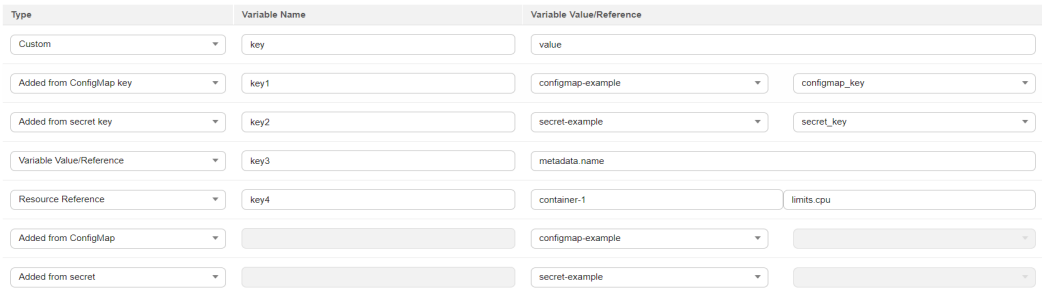
Step 1 Log in to the CCE console.

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.

Figure 4-13 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
ey1=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.
```

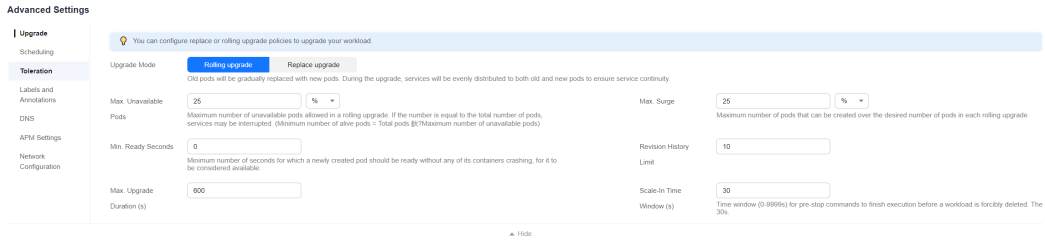
4.4.2.6 Configuring the Workload Upgrade Policy

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 pods are deleted, and then new pods are created.

Figure 4-14 Configuring a workload upgrade policy



Upgrade Parameters

Parameter	Description	Constraint
Max. Surge (maxSurge)	Specifies the percentage of pods that are allowed based on the value of **spec.replicas**. The default value is **25%**. For example, if **spec.replicas** is set to **4**, there should be a maximum of five pods during the upgrade, and one pod can be added each time (at a step of 1). During the upgrade, the value is converted into a number and rounded up. The value can also be set to an absolute number.	This parameter is only supported by Deployments and DaemonSets.

Parameter	Description	Constraint
Max. Unavailable Pods (maxUnavailable)	Specifies the percentage of pods that can be deleted based on the value of **spec.replicas**. The default value is **25%** For example, if **spec.replicas** is set to **4**, there should be at least three pods during the upgrade, and one pod can be deleted (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
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

Parameter	Description	Constraint
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 the earlier version when a fault occurs during the 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**.

4.4.2.7 Labels and Annotations

Pod Annotations

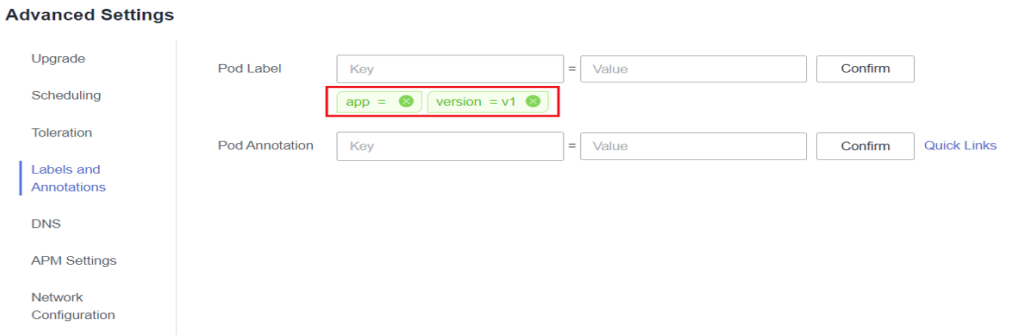
CCE allows you to add annotations to a YAML file to realize some advanced pod functions. The following table describes the annotations you can add.

Table 4-16 Pod annotations

Annotation	Description	Default Value
kubernetes.io/ingress-bandwidth	Ingress bandwidth of a pod. For details, see **Configuring QoS for a Pod**.	None
kubernetes.io/egress-bandwidth	Egress bandwidth of a pod. For details, see **Configuring QoS for a Pod**.	None
node.cce.io/node-az-list	A list of AZs for pod affinity. For details, see **4.4.2.8 Setting AZ Affinity**.	-

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:
    ...
```

4.4.2.8 Setting AZ Affinity

Scenario

An availability zone (AZ) is a physical region in a data center where resources use independent power supplies and networks. Nodes in the same AZ can quickly communicate with each other through a high-speed network, while nodes in different AZs need to communicate with each other across physical distances, which may cause latency and risks.

Scheduling pods to different AZs improves the availability and fault tolerance of applications.

Procedure

For a CCE Autopilot cluster, you can set workload annotations to implement AZ affinity and schedule pods to specified AZs.

Step 1 Log in to the CCE console.

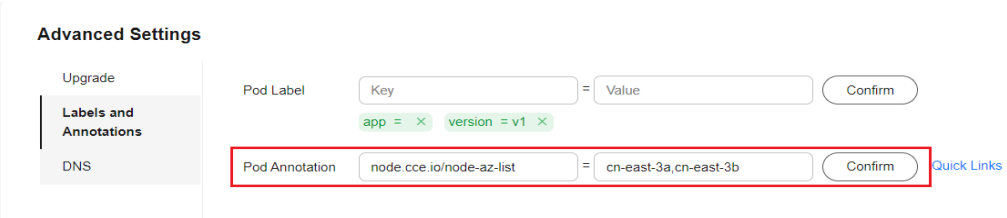
Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane on the left, and click the **Create Workload** in the upper right corner.

Step 3 In the **Advanced Settings** area, select **Labels and Annotations** and set the following annotation:

- **Key**: **node.cce.io/node-az-list**
- **Value**: AZ name. Use commas (,) to separate multiple AZs.

  For details about the AZs in different regions, see **Regions and Endpoints**.

Figure 4-15 Setting AZ affinity



Step 4 Configure other workload parameters and click **Create Workload**.

----End

4.4.3 Accessing a Container

Scenario

If you encounter unexpected problems when using a container, you can log in to the container to debug it.

Logging In to a Container Using CloudShell

NOTICE

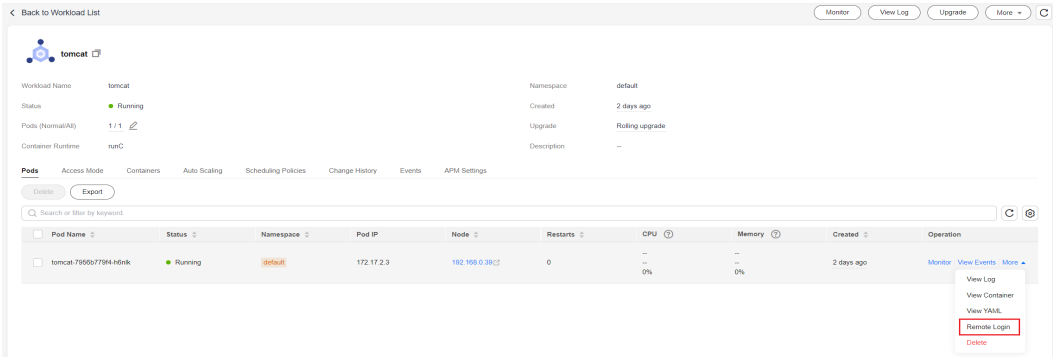
- CloudShell is implemented based on VPC Endpoint (VPCEP). To use kubectl to access a cluster, configure the security group (*Cluster name*-cce-control-*Random number*) on the master node of the cluster to allow access to port 5443. By default, port 5443 allows access from all CIDR blocks. If you have hardened security groups and any cluster cannot be accessed in CloudShell, check whether port 5443 allows access from 198.19.0.0/16.
- Currently, you can use CloudShell to log in to containers only in CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN South-Guangzhou, CN Southwest-Guiyang1, and CN North-Ulanqab1 regions.

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.

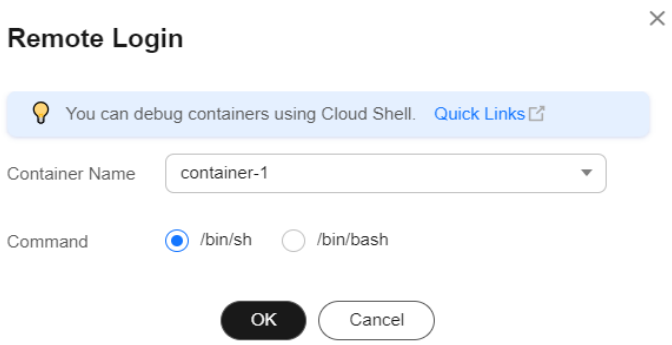
Step 3 Locate the target pod and choose **More** > **Remote Login** in the **Operation** column.

Figure 4-16 Accessing a container



Step 4 In the displayed dialog box, select the container you want to access and the command, and click **OK**.

Figure 4-17 Selecting a container and login command

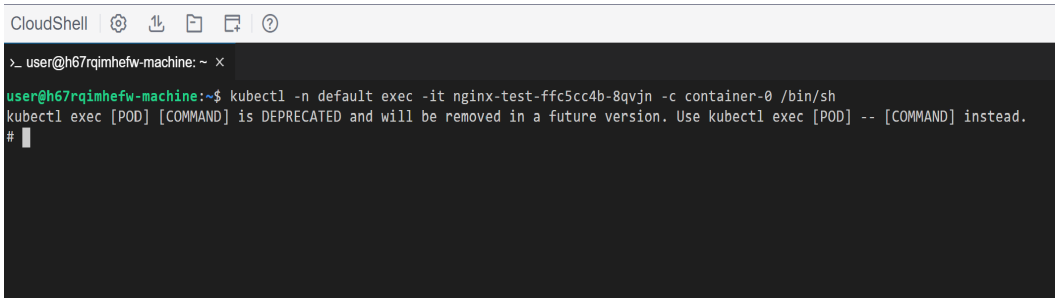


Step 5 You will be automatically redirected to CloudShell. Then, the system initializes kubectl and runs the **kubectl exec** command to log in to the container.

NOTE

Wait for 5 to 10 seconds until the **kubectl exec** command is automatically executed.

Figure 4-18 CloudShell page



----End

Logging In to a Container Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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

4.4.4 Managing Workloads and Jobs

Scenario

After a workload is created, you can upgrade, monitor, roll back, or delete the workload, as well as edit its YAML file.

Table 4-17 Workload/Job management

Operation	Description
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 the YAML files of Deployments, StatefulSets, 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 will only be applied for the new pods, and 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.
Enable/Disable 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.

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 the **View Log** of the target workload.

On the displayed **View Log** window, you can view logs.

Figure 4-19 Viewing logs of a workload

----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 the YAML files of Deployments, StatefulSets, 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)

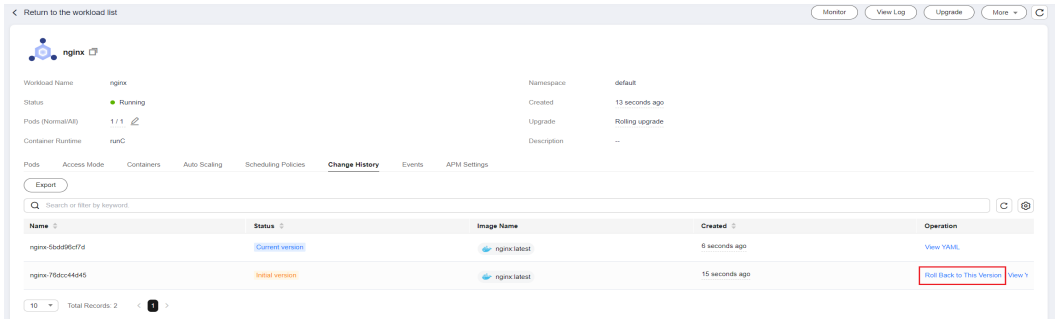
CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

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 4-20 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.
- If a Deployment is being upgraded, it can be upgraded or rolled back. Its pods will inherit the latest updates of the Deployment. If they are inconsistent, the pods are upgraded automatically according to the latest information of the Deployment.

NOTICE

The workload cannot be rolled back when the upgrade is disabled.

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

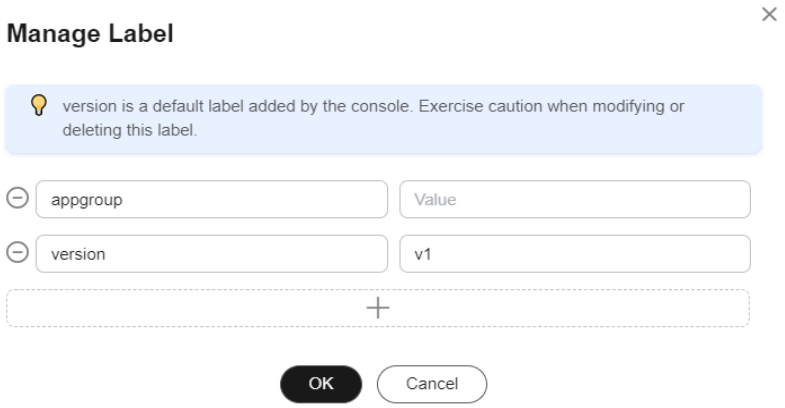
Labels are key-value pairs and can be attached to workloads. You can manage and select workloads by labels. You can add labels to multiple workloads or a specified workload.

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 **Add**, enter a key and a value, and click **OK**.

Figure 4-21 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 Deployments as an example to illustrate how to view events of a workload. To view the event of a job or cron jon, 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

4.4.5 Managing kernel Options

CCE Autopilot is a serverless cluster and isolated from the kernel of physical machines. kernel tuning is a common practice in advanced service deployment scenarios. In a safe situation, CCE Autopilot allows you to configure kernel parameters through a security context of a pod based on the solution recommended by the Kubernetes community, greatly improving the flexibility of service deployment. For details of security contexts, see **Configure a Security Context for a Pod or Container**.

In Linux, sysctl is the most common method of modifying kernel parameters. In Kubernetes, kernel parameters are configured through the sysctl security context of the pod. If you are not familiar with the sysctl concept, see **Using sysctls in a Kubernetes Cluster**. A security context applies to all containers in the same pod.

CCE Autopilot allows you to modify the following non-secure sysctl parameters:

```
kernel.shm*,
kernel.msg*,
kernel.sem,
fs.mqueue.*,
net.*
```

NOTICE

To avoid affecting the stability of the OS, modify the sysctl parameters after understanding the consequences of the modification.

The sysctl parameters with a namespace may change in future Linux kernel versions.

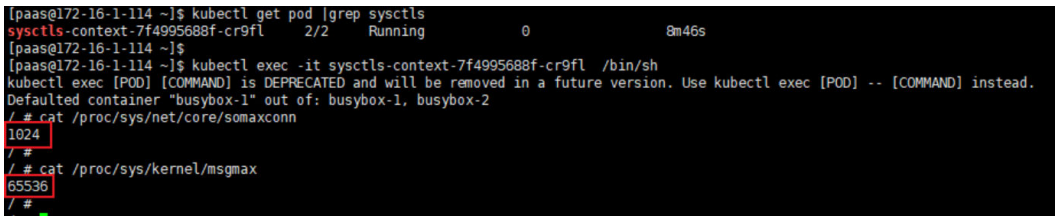
Non-secure sysctl parameters are unstable. Using non-secure sysctl parameters may cause some serious problems, such as container errors. You need to take care of the risks.

In the following example, the pod's security context is used to set two sysctl parameters: **kernel.msgmax** and **net.core.somaxconn**.

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctls-context-example
spec:
  securityContext:
    sysctls:
    - name: kernel.msgmax
      value: "65536"
    - name: net.core.somaxconn
      value: "1024"
  ...
```


Go to the container to check whether the configuration takes effect.

```
kubectl exec -it podname -c container-1 -- /bin/sh
```



4.4.6 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.

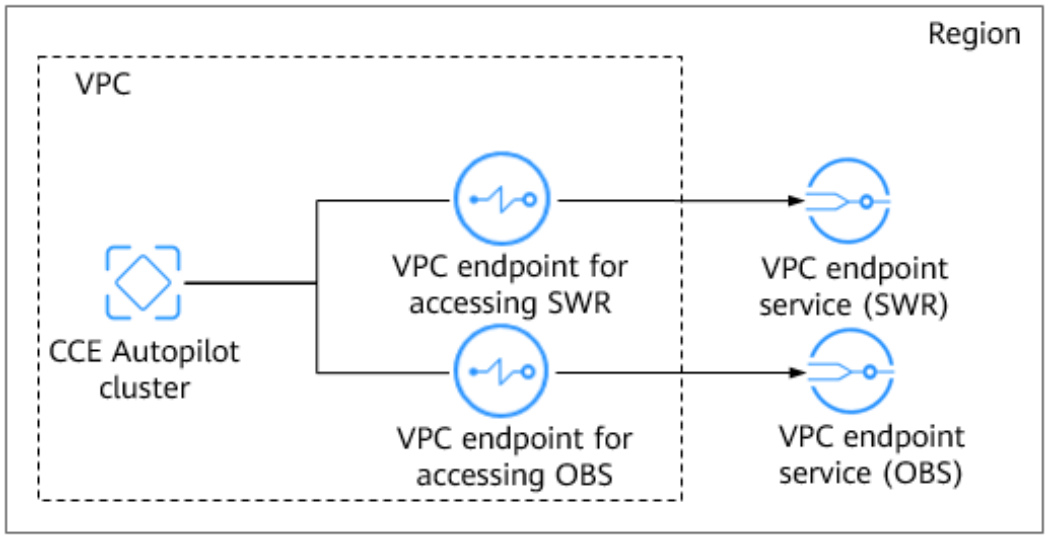
----End

4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS

When deploying a workload in a CCE Autopilot cluster, you need to use the VPC endpoints of SWR and OBS to pull images.

For details about VPC Endpoint, see **What Is VPC Endpoint?**

Figure 4-22 A CCE Autopilot cluster accessing SWR and OBS



Configuring the VPC Endpoint for Accessing SWR

Step 1 Log in to the **VPC Endpoint** console.

Step 2 On the displayed page, click **Buy VPC Endpoint**.

Step 3 Configure the parameters.

Table 4-18 Parameters for creating a VPC endpoint

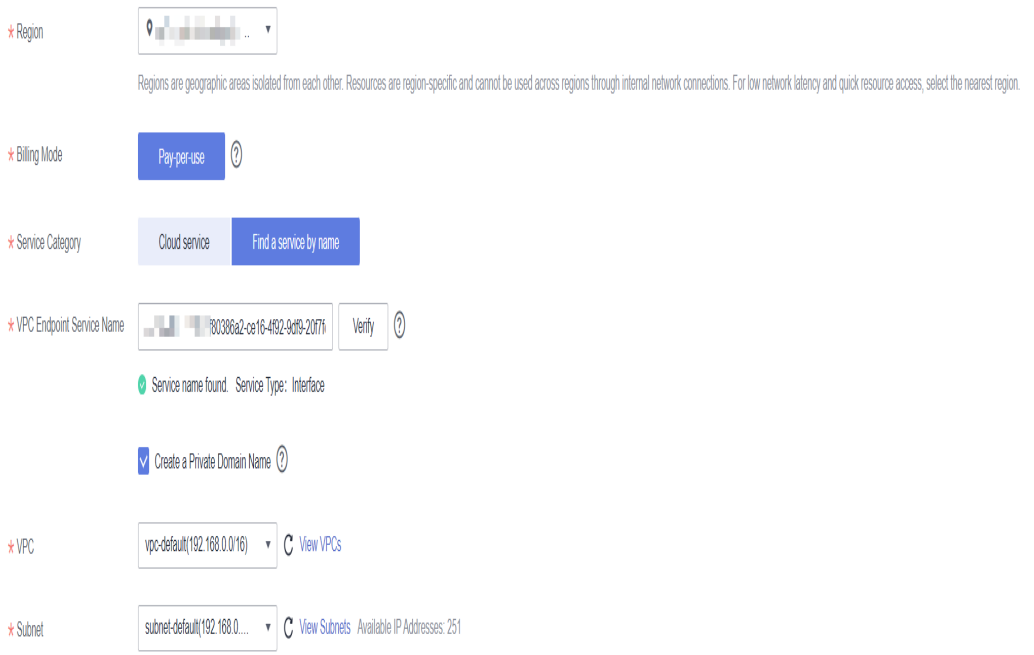
Parameter	Description
Region	Select the region where the VPC endpoint is located. The VPC endpoint must be in the same region as the cluster.
Billing Mode	Select **Pay-per-use**.
Service Category	Select **Find a service by name**.
VPC Endpoint Service Name	Enter a service name based on **Table 4-19** and click **Verify**.
VPC	Select the VPC where the cluster is located.
Subnet	Select a subnet.

Parameter	Description
IPv4 Address	By default, **Automatically assign IPv4 address** is selected. You can also select **Manually specify an IP address** as required.

Table 4-19 VPC endpoint service names for SWR

Region	Name
CN South-Guangzhou-InvitationOnly	cn-south-4.SWR.f80386a2-ce16-4f92-9df9-20f7fc01e7a2
CN Southwest-Guiyang1	com.myhuaweicloud.cn-southwest-2.swr
CN South-Guangzhou	swr.cn-south-1.myhuaweicloud.com
CN East-Shanghai1	com.myhuaweicloud.cn-east-3.swr

Figure 4-23 Creating a VPC endpoint for SWR



Step 4 Click **Next** to confirm the configuration.

- If the configuration is correct, click **Submit**.
- If any parameter is incorrect, click **Previous** to modify it as needed, and then click **Submit**.

Step 5 Go back to the VPC endpoint list.

If the status of the VPC endpoint changes to **Accepted**, the VPC endpoint is connected to the VPC endpoint service.

----End

Configuring the VPC Endpoint for Accessing OBS

Step 1 Log in to the **VPC Endpoint** console.

Step 2 On the displayed page, click **Buy VPC Endpoint**.

Step 3 Configure the parameters.

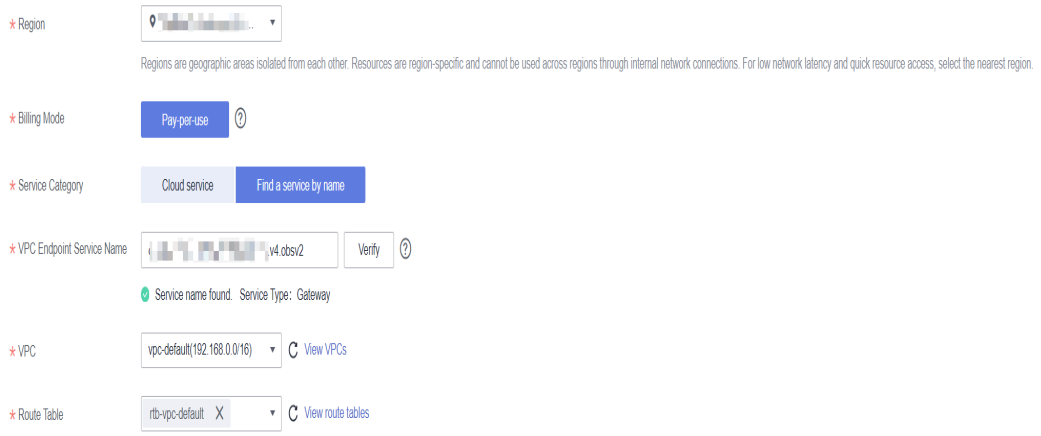
Table 4-20 Parameters for creating a VPC endpoint

Parameter	Description
Region	Select the region where the VPC endpoint is located. The VPC endpoint must be in the same region as the cluster.
Billing Mode	Select **Pay-per-use**.
Service Category	Select **Find a service by name**.
VPC Endpoint Service Name	Enter a service name based on **Table 4-21** and click **Verify**.
VPC	Select the VPC where the cluster is located.
Route Table	Select a route table.

Table 4-21 VPC endpoint service names for OBS

Region	Name
CN South-Guangzhou-InvitationOnly	cn-south-4.com.myhuaweicloud.v4.obsv2
CN Southwest-Guiyang1	cn-southwest-2.com.myhuaweicloud.v4.obsv2
CN South-Guangzhou	cn-southcom.myhuaweicloud.v4.obsv2
CN East-Shanghai1	cn-east-3.com.myhuaweicloud.v4.obsv2

Figure 4-24 Creating a VPC endpoint for OBS



Step 4 Click **Next** to confirm the configuration.

- If the configuration is correct, click **Submit**.
- If any parameter is incorrect, click **Previous** to modify it as needed, and then click **Submit**.

Step 5 Go back to the VPC endpoint list.

If the status of the VPC endpoint changes to **Accepted**, the VPC endpoint is connected to the VPC endpoint service.

----End

4.4.8 Workload FAQs

This topic describes the FAQs related to CCE Autopilot clusters.

What Do I Do If an Image Can't Be Pulled from SWR During Workload Creation?

The following information is displayed when you deploy a workload in a CCE Autopilot cluster:

```
Failed to pull image "swr.cn-north-xx.myhuaweicloud.com/xx/nginx:latest": rpc error: code = Unknown desc = failed to pull and unpack image "swr.cn-north-7.myhuaweicloud.com/xx/nginx:latest": failed to resolve reference "swr.cn-north-7.myhuaweicloud.com/xx/nginx/latest": failed to do request: Head "https://swr.cn-north-xx.myhuaweicloud.com/v2/xx/nginx/manifests/latest": dial tcp 100.79.xx.xx:443: i/o timeout
```

The error information indicates that the SWR image cannot be pulled during workload creation. Check whether the VPC endpoints for accessing OBS and SWR are normal. If no VPC endpoints are available for accessing OBS and SWR, create them by referring to **4.4.7 Configuring VPC Endpoints for Accessing SWR and OBS**.

What Do I Do If "Cluster pod max limit exceeded" Is Displayed for a Workload?

When a workload is being created, the following error may occur:

```
Cluster pod max limit exceeded(x)
```


This indicates that the maximum number of pods in the cluster is reached and no more pods can be created. *x* indicates the maximum of pods in the cluster. The default value is **500**.

Plan the number of pods in the cluster appropriately.

NOTE

The add-ons installed in the cluster occupy the pod quota. Plan the pod quota properly.

4.5 Services and Ingresses

4.5.1 Service

4.5.1.1 ClusterIP

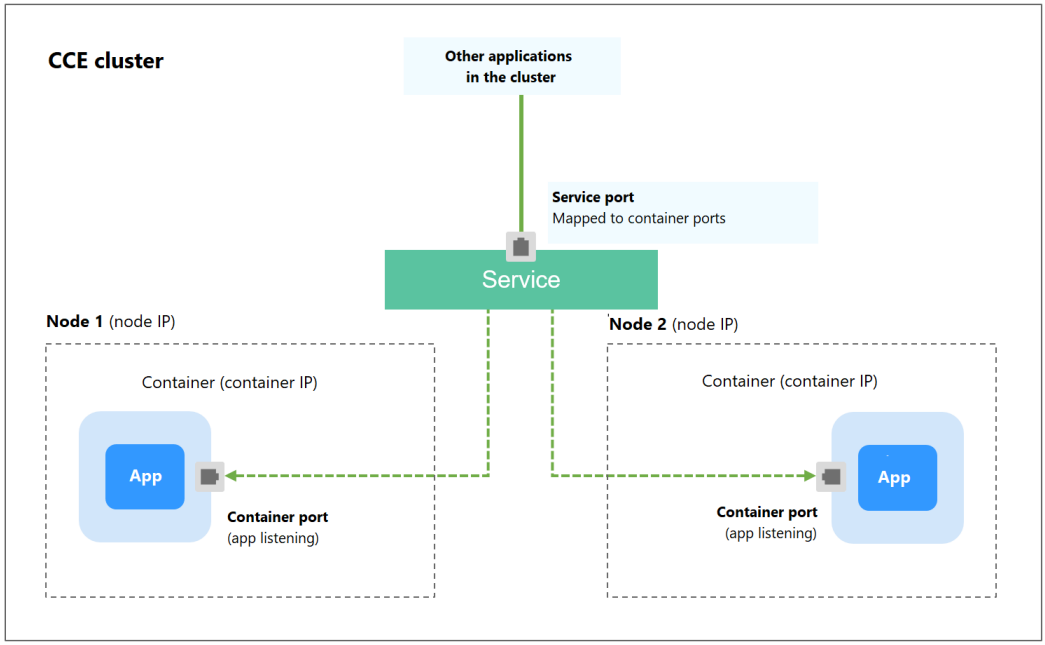
Scenario

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 4-25 shows the mapping relationships between access channels, container ports, and access ports.

Figure 4-25 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 to which the workload belongs.
- **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.

Step 4 Click **OK**.

----End

Setting the Access Type Using kubectl

You can run kubectl commands to configure Service access. This section uses an Nginx workload as an example to describe how to implement intra-cluster access using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **3.2.3.1 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

4.5.1.2 LoadBalancer

4.5.1.2.1 Creating a LoadBalancer Service

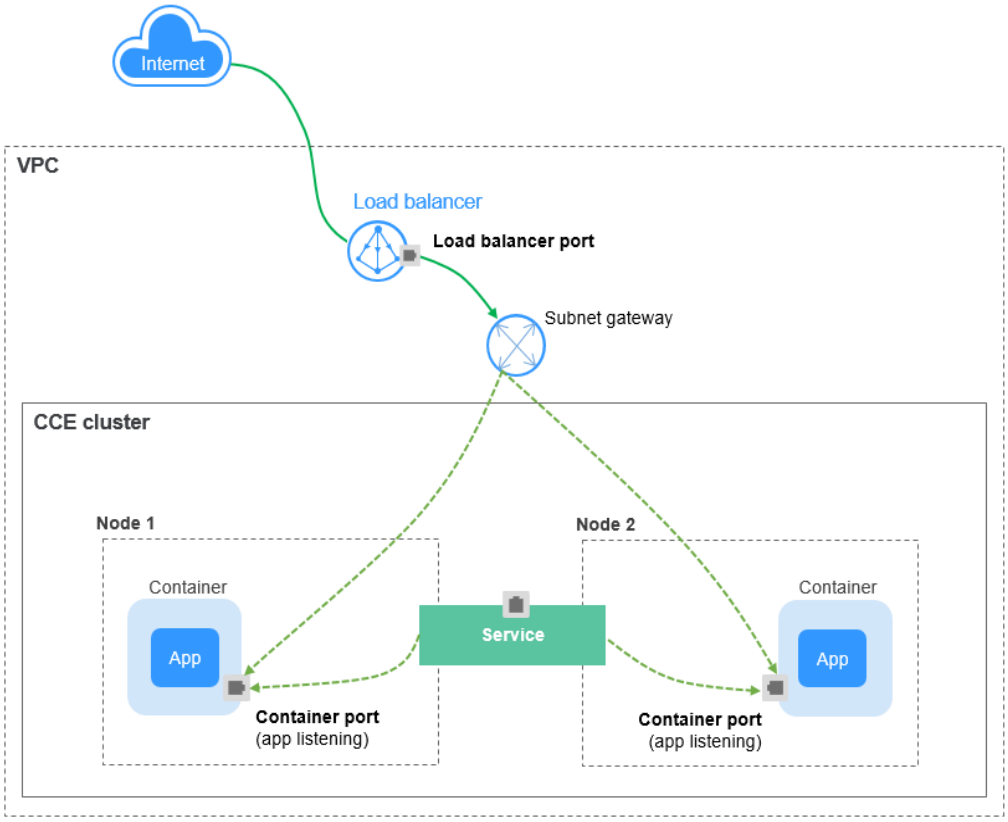
Scenario

LoadBalancer Services can access workloads from the public network through ELB, 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**.

If dedicated load balancers are deployed for CCE Autopilot clusters, passthrough networking is supported to reduce the network 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 4-26 Passthrough networking



Constraints

- Automatically created load balancers should not be used by other resources. If they are used by other resources, they cannot be deleted completely.
- Dedicated load balancers with private IP addresses bound and used for network load balancing (load balancing over TCP or UDP) should be selected. If a Service needs to support HTTP, dedicated load balancers must also support application load balancing (load balancing over HTTP or HTTPS).

Creating a LoadBalancer 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 on the left, 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 to which the workload belongs.

- **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.

- **Selector**: Add a label and click **Confirm**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Load Balancer**: Select the load balancer type and whether to use an existing load balancer or create a new one.

  Select **Dedicated**.

  Select either **Use existing** or **Auto create**. For more information, see **Table 4-22**.

  Table 4-22 Load balancer configurations

Option	Description
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 only available for enterprise users who have enabled an enterprise project. Enterprise projects facilitate project-level management and grouping of cloud resources and users. – **AZ**: This parameter is only available for dedicated load balancers. You can create load balancers in multiple AZs to improve service availability. If diaster recovery is required, you are advised to select multiple AZs. – **Frontend Subnet**: This parameter is only available for dedicated load balancers. It is used to allocate IP addresses to load balancers to receive traffic from clients. – **Backend Subnet**: This parameter is only available for dedicated load balancers. It is used to allocate IP addresses for load balancers to routing traffic to pods. – **Network Specifications**: This parameter is only available for dedicated load balancers. – **EIP**: If you select **Auto create**, you can select a bandwidth billing option and set the bandwidth. – **Resource Tag**: You can add resource tags to classify resources. You can create predefined tags on the Tag Management Service (TMS) console. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency.

You can click **Edit** in the **Set ELB** area to set 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.

- **Type**: This option is disabled by default. You can also select **Source IP address**. In source IP address-based sticky sessions, requests from the same IP address are forwarded to the same backend server.

  NOTE

  When **Source IP hash** is used for load balancing, sticky sessions are not available.

- **Health Check**: Configure health check for the load balancer.
  - **Global health check**: applies only to ports of the same protocol. You are advised to select **Custom health check**.
  - **Custom health check**: applies to **ports** used by different protocols.

Table 4-23 Health check parameters

Parameter	Description
Protocol	When the protocol is set to **TCP**, both TCP and HTTP are supported. When the protocol is set to **UDP**, only UDP 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 ports (NodePort or container port of the Service) are used for health check. You can also specify another port for health check. If a port is specified, a service port named **cce-healthz** will be added for the Service. – **Node 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 has an elastic network interface associated, 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**.

- **Port**
  - **Protocol**: protocol used by the Service.
  - **Service Port**: port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: port on which the workload listens. For example, Nginx uses port 80 by default.
  - **Health Check**: If **Health Check** is set to **Custom health check**, you can configure health check for ports that come with different protocols. For details, see **Table 4-23**.

  NOTE

  When a LoadBalancer Service is created, a random node port number (NodePort) is automatically generated.
- **Annotation**: A LoadBalancer Service has some advanced features, which are implemented by annotations. For details, see **Using Annotations to Configure Load Balancing**.

Step 4 Click **OK**.

----End

Using kubectl to Create a Service (Using an Existing Load Balancer)

You can set the Service when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 Connecting to a Cluster Using kubectl**.

Step 2 Create the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files and edit them.

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

Before enabling sticky session, ensure that the following conditions are met:

- The workload protocol is TCP.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    kubernetes.io/elb.id: <your_elb_id>                    # ELB 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 (min)
    kubernetes.io/elb.health-check-flag: 'on'              # Enable the ELB health check function.
    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
```

This example uses annotations to implement some advanced features of load balancing, such as sticky sessions and health check. For details, see **Table 4-24**.

For more annotations and examples related to advanced features, see see **Using Annotations to Configure Load Balancing**.

Table 4-24 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	ID of an enhanced 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**.
kubernetes.io/elb.class	Yes	String	The value can be: • **performance**: dedicated load balancer **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 is supported. That is, access requests from the same IP address are forwarded to the same backend server. • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter 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 4-25** Object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave blank this parameter 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 at the same time.
kubernetes.io/elb.health-check-option	No	**Table 4-26** Object	ELB health check configuration items.

Table 4-25 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 4-26 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 access type has been set, 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        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 4-27 Accessing Nginx through the LoadBalancer Service

----End

Using kubectl to Create a Service (Automatically Creating a Load Balancer)

You can set the Service when creating a workload using kubectl. This section uses an Nginx workload as an example to describe how to add a LoadBalancer Service using kubectl.

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 Connecting to a Cluster Using kubectl**.

Step 2 Create the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files and edit them.

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

Before enabling sticky session, ensure that the following condition is met:

- The workload protocol is TCP.

Example of a Service using a dedicated load balancer on a public network:

```
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",
      "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 (min)
    kubernetes.io/elb.health-check-flag: 'on'                  # Enable the ELB health check function.
    kubernetes.io/elb.health-check-option: '{
      "protocol":"TCP",
      "delay":"5",
      "timeout":"10",
      "max_retries":"3"
    }'
spec:
  selector:
    app: nginx
  ports:
  - name: cce-service-0
    targetPort: 80
    nodePort: 0
    port: 80
    protocol: TCP
  type: LoadBalancer
```

This example uses annotations to implement some advanced features, such as sticky sessions and health check. For details, see **Table 4-27**.

For more annotations and examples related to advanced features, see see **Using Annotations to Configure Load Balancing**.

Table 4-27 annotations parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Select a proper load balancer type. The value can be: • **performance**: dedicated load balancer

Parameter	Mandatory	Type	Description
kubernetes.io/elb.autocreate	Yes	**elb.autocreate** object	Whether to automatically create a load balancer associated with the Service. **Example** • If a public network load balancer will be automatically created, set this parameter to the following value: {"type":"public","bandwidth_name":"cce-bandwidth-1551163379627","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"} • If a private network load balancer will be automatically created, set this parameter to the following value: {"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 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 is supported. That is, access requests from the same IP address are forwarded to the same backend server. • Disabling sticky session: Do not configure this parameter. • Enabling sticky session: Set this parameter 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 4-25** Object	Sticky session timeout.
kubernetes.io/elb.health-check-flag	No	String	Whether to enable the ELB health check. • Enabling health check: Leave blank this parameter 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 at the same time.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.health-check-option	No	Table 4-26 Object	ELB health check configuration items.

Table 4-28 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 default value is 1 to 2000 Mbit/s. 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
vip_subnet_cidr_id	No	String	Specifies the 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.
available_zone	Yes	Array of strings	AZ where the load balancer is located. You can obtain all supported AZs by **querying the AZ list**. This parameter is available only for dedicated load balancers.

Parameter	Mandatory	Type	Description
l4_flavor_name	Yes	String	Flavor name of the Layer-4 load balancer. You can obtain all supported types by **querying 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 **querying 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	Specifies the 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

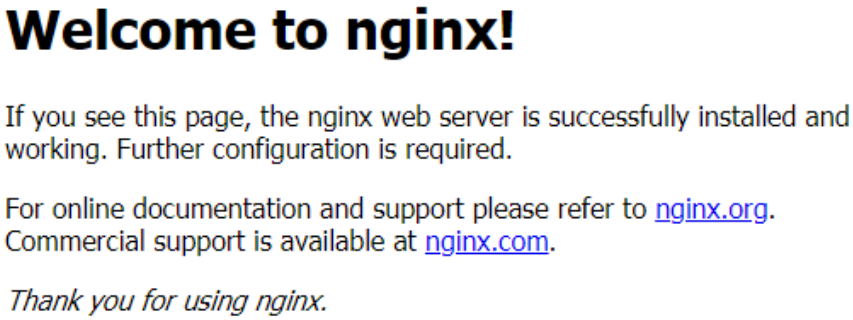
If information similar to the following is displayed, the access type has been set, and the workload is accessible.

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.247.0.1      <none>         443/TCP        3d
nginx        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.

Nginx is accessible.

Figure 4-28 Accessing Nginx through the LoadBalancer Service



----End

4.5.1.2.2 Configuring Security Group Rules for ICMP Traffic

Scenario

Dedicated load balancers are used for CCE Autopilot clusters. If both the listener protocol and the health check protocol are UDP, you need to allow ICMP traffic from the backend subnet of the load balancer in the security group associated with the elastic network interfaces.

Procedure

Step 1 Log in to the CCE console, choose **Service List** > **Networking** > **Virtual Private Cloud**. In the navigation pane on the left, choose **Access Control** > **Security Groups**.

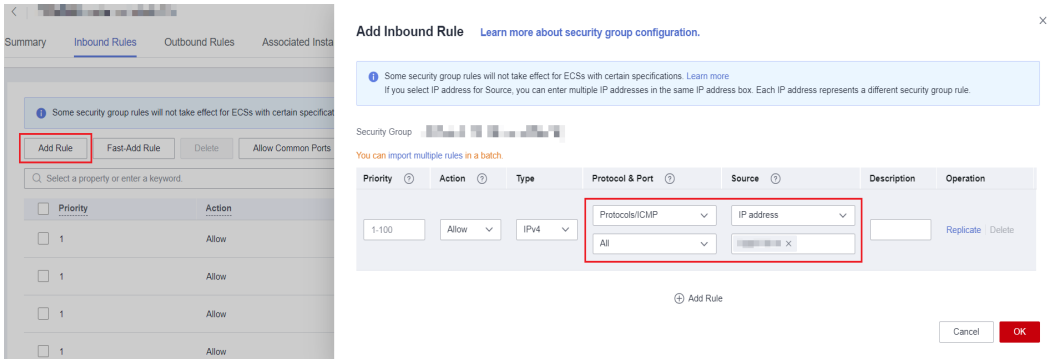
Step 2 In the security group list, locate the security group associated with the elastic network interfaces. The default security group is named *{Cluster name}-cce-eni-{Random ID}*.

If you specify a custom security group for the cluster, select this security group.

Step 3 Click the name of the security group. On the **Inbound Rules** tab, click **Add Rule** to add an inbound rule. For details, see **Figure 4-29**.

- **Protocol & Port**: Select all ICMP ports.
- **Source**: Enter the backend subnet of the load balancer.

Figure 4-29 Adding a security group rule



Step 4 Click **OK**.

----End

4.5.1.3 Headless Service

The preceding types of 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
```

4.5.2 Ingresses

4.5.2.1 LoadBalancer Ingresses

4.5.2.1.1 Creating an ELB Ingress on the Console

Prerequisites

- A workload is available in the cluster (because an ingress enables network access for workloads). If no workload is available, deploy a workload by referring to **4.4.1 Creating a Workload**.
- A Service has been configured for the workload.

Precautions

- Other resources should not use the load balancer automatically created by an ingress. If the load balancer is used by other resources, there will be residual resources when the ingress is deleted.
- After an ingress is created, you can only upgrade and maintain the configuration of each load balancer on the CCE console. Do not modify the configuration on the ELB console. If you modify the configuration on the ELB console, the ingress may be abnormal.
- The URL specified in an ingress forwarding policy must be the same as that used to access the backend Service. If the URL is not the same, 404 will be returned.
- Dedicated load balancers must be of the application type (HTTP/HTTPS) and each have a private IP address bound.
- If multiple ingresses are used to connect to the same ELB port in the same 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 an ELB Ingress

An Nginx workload is used as an example to describe how to add an ELB ingress.

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

Step 2 In the navigation pane on the left, choose **Services & Ingresses**. On the **Ingresses** tab, click **Create Ingress** in the upper right corner.

Step 3 Configure the parameters.

- **Name**: Enter a name for the ingress, for example, **ingress-demo**.
- **Load Balancer**: Select the load balancer type and whether to use an existing load balancer or create a new one.

  Only dedicated load balancers are allowed. A dedicated load balancer must be of the application (HTTP/HTTPS) type and work on a private network.

  You can select **Use existing** or **Auto create** to obtain a load balancer. For details about the configuration of different creation modes, see **Table 4-29**.

  Table 4-29 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/Application-oriented 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. The predefined tags are available to all resources that support tags. You can use predefined tags to improve the tag creation and resource migration efficiency.

- **Listener**: Ingress configures a listener for the load balancer, which listens to and distributes the requests. After the configuration is complete, a listener will be created on the load balancer. The default listener name is in the format of k8s_<Protocol>_<Port>, for example, *k8s_HTTP_80*.
  - **External Protocol**: **HTTP** and **HTTPS** are available.
  - **External Port**: Port for the load balancer to receive requests. You can specify any port.
  - **Certificate Source**: TLS secrets and ELB server certificates are supported.
  - **Server Certificate**: When an HTTPS listener is added to the load balancer, bind a certificate to the load balancer for encrypted transmission for HTTPS data.
    - **TLS secret**: For details about how to create a secret certificate, see **4.9.3 Creating a Secret**.
    - **ELB server certificate**: Use the certificate created in the ELB service.

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**: Server Name Indication (SNI) is an extended protocol of TLS. It allows multiple TLS-compliant domain names to be provided for external access using the same IP address and port, and different domain names can use different security certificates. If 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 selected.
    - SNI is supported only in clusters of v1.15.11 and later.
    - You need to specify the domain name for the SNI certificate. Only one domain name can be specified for each certificate. Wildcard-domain certificates are supported.

- **Forwarding Policy**: When the access address of a request matches the forwarding policy (that 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 the domain name used for access. 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.
  - **URL Matching Rule**
    - **Prefix match**: If the URL is set to **/healthz**, the URL that meets **/healthz** 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. 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.
  - **URL**: access path, for example, **/healthz**.

NOTE

The access path added here must exist in the backend applications. If it does not exist, requests will fail to be forwarded.

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, 404 will be returned.

- **Destination Service**: Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
- **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 feature is disabled by default. There are two options:
    - **Load balancer cookie**: Enter the **Stickiness Duration** , which ranges from **1** to **1440** minutes.

      NOTE

      When **Source IP hash** is used for load balancing, sticky sessions are not available.

  - **Health Check**: Set the health check configuration of the load balancer. If this feature is enabled, you need to configure the following parameters.

Parameter	Description
Protocol	When the protocol of the target Service is TCP and HTTP are supported. ○ **Check Path**: This parameter is only available for HTTP health check. It specifies the URL for health check. The check path must start with a slash (/) and contain 1 to 80 characters.
Port	By default, the service ports (Node Port or container port of the Service) are used for health check. You can also specify another port for health check. If a port is specified, a service port named **cce-healthz** will be added for the Service. ○ **Node 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 has an elastic network interface associated, 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.
- **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 **Creating an Ingress - Automatically Creating a Load Balancer** and **Creating an Ingress - Interconnecting with an Existing Load Balancer**

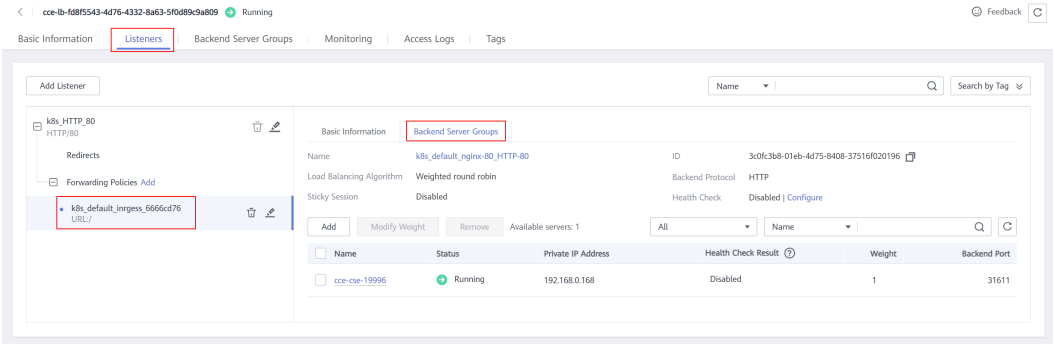
Step 4 After the configuration is complete, click **OK**. After the ingress is created, it is displayed in the ingress list.

On the ELB console, you can view the ELB automatically created through CCE. The default name is **cce-lb-ingress.UID**. Click the load balancer name to access its details page. On the **Listeners** tab, view the route settings of the ingress, such as the URL, listener port, and backend port.

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.

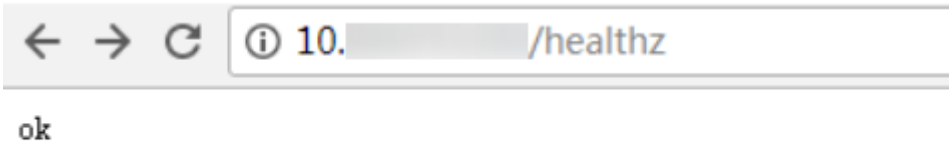
Figure 4-30 ELB routing configuration



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.

Figure 4-31 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?**

4.5.2.1.2 Using kubectl to Create an ELB Ingress

Scenario

This section uses an Nginx workload as an example to describe how to create an ELB 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 **Creating an Ingress - Automatically Creating a Load Balancer**.
- If a load balancer is available in the same VPC, perform the operation by referring to **Creating an Ingress - Interconnecting with an Existing Load Balancer**.

Prerequisites

- A workload is available in the cluster (because an ingress enables network access for workloads). If no workload is available, deploy a sample Nginx workload by referring to **4.4.1 Creating a Workload**.
- A Service has been configured for the workload.
- Dedicated load balancers must be of the application type (HTTP/HTTPS) and each have a private IP address bound.

Creating an Ingress - Automatically Creating a Load Balancer

The following describes how to run the kubectl command to automatically create a load balancer when creating an ingress.

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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

Example of using a dedicated load balancer on a public network:

```
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",
          "elb_virsubnet_ids":["******"],
          "available_zone": [
              "ap-southeast-1a"
          ],
          "l7_flavor_name": "L7_flavor.elb.s1.small"
      }'
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
```

Table 4-30 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	The value can be: • **performance**: dedicated load balancer
ingressClassName	Yes	String	**cce**: The self-developed ELB ingress is used. This parameter is mandatory when an ingress is created by calling the API.
kubernetes.io/elb.port	Yes	Integer	This parameter indicates the external port registered with the address of the LoadBalancer Service. Supported range: 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 4-31**. **Example** • If a public network load balancer will be automatically created, set this parameter to the following value: {"type":"public","bandwidth_name":"cce-bandwidth-******","bandwidth_chargemode":"bandwidth","bandwidth_size":5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"} • If a private network load balancer will be automatically created, set this parameter to the following value: {"type":"inner","name":"A-location-d-test"}
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 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 4-31 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**

Parameter	Mandatory	Type	Description
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.
bandwidth_sharetype	Yes for public network load balancers	String	Bandwidth sharing mode. • **PER**: dedicated bandwidth

Parameter	Mandatory	Type	Description
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 **querying 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 **querying 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 **querying 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	Specifies the 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
```

kubectl get ingress

If information similar to the following is displayed, the ingress has been created successfully and the workload is accessible.

```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 4 Enter **http://*121.**.**.***:80** in the address box of the browser to access the workload.

121.**.**.** indicates the IP address of the unified load balancer.

----End

Creating an Ingress - Interconnecting with an Existing Load Balancer

CCE allows you to connect to an existing load balancer when creating an ingress.

NOTE

- Dedicated load balancers must be of the application type (HTTP/HTTPS) and each has a private IP address bound.

The YAML file is configured 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
```

Table 4-32 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.

Parameter	Mandatory	Type	Description
kubernetes.io/elb.class	Yes	String	Load balancer type. • **performance**: dedicated load balancer **NOTE** If an ELB Ingress accesses an existing dedicated load balancer, the dedicated load balancer must be of the application load balancing (HTTP/HTTPS) type.

4.5.2.2 Nginx Ingresses

4.5.2.2.1 Creating Nginx Ingresses on the Console

Prerequisites

- A workload is available in the cluster (because an ingress enables network access for workloads). If no workload is available, deploy a workload by referring to **4.4.1 Creating a Workload**.
- A ClusterIP Service has been configured for the workload. For details about how to configure the Service, see **4.5.1.1 ClusterIP**.
- To add an Nginx ingress, ensure that the NGINX Ingress Controller add-on has been installed in the cluster. For details, see **Installing the Add-on**.

Constraints

- It is not recommended that you modify any configuration of a load balancer on the ELB console. If you modify the configuration on the ELB console, the Service will be abnormal. If you have modified the configuration, uninstall the nginx-ingress add-on and reinstall it.
- The URL specified in an ingress forwarding policy must be the same as that used to access the backend Service. If the URL is not the same, 404 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.

Procedure

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 In the navigation pane on the left, choose **Services & Ingresses**. On the **Ingresses** tab, click **Create Ingress** in the upper right corner.

Step 3 Configure ingress parameters.

- **Name**: Enter a name for the 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 when the **4.11.5 NGINX Ingress Controller** add-on has been installed in the cluster.

  After you switch on  , nginx-ingress is interconnected to provide layer-7 access. You can configure the following parameters:

  TLS: nginx-ingress supports HTTP and HTTPS. The default listening port reserved during nginx-ingress installation is **80** for HTTP requests and **443** for HTTPS requests. To use HTTPS, configure the server certificate.

  - **Server Certificate**: When creating an HTTPS listener, bind a TLS certificate to support encrypted authentication for HTTPS data transmission. For details on how to create a secret, see **4.9.3 Creating a Secret**.
  - **SNI**: an extended protocol of TLS. It allows multiple TLS-compliant domain names to be provided for external access using the same IP address and port, and different domain names can use different security certificates. If 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**: Enter the domain name used for access. 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.
  - **URL 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.
  - **URL**: 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 applications. If it does not exist, requests will fail to be forwarded.

  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, 404 will be returned.

  - **Destination Service**: Select an existing Service or create a Service. Services that do not meet search criteria are automatically filtered out.
  - **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. You can use **annotations** to query the configurations supported by nginx-ingress.

Step 4 Click **OK**.

After the ingress is created, it is displayed in the ingress list.

----End

4.5.2.2.2 Using kubectl to Create an Nginx Ingress

Scenario

An Nginx workload is used as an example to describe how you can create an Nginx ingress using kubectl.

Prerequisites

- The NGINX Ingress Controller add-on has been installed in a cluster. For details about how to install the add-on, see **Installing the Add-on**.
- A workload is available in the cluster (because an ingress enables network access for workloads). If no workload is available, deploy a workload by referring to **4.4.1 Creating a Workload**.
- A ClusterIP Service has been configured for the workload. For details about how to configure the Service, see **4.5.1.1 ClusterIP**.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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 uses HTTP as an example to describe how to configure the YAML file:

```
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.
```

Table 4-33 Key parameters

Parameter	Mandatory	Type	Description
ingressClassName	Yes	String	**nginx**: indicates that Nginx ingress is used. This option cannot be used if the nginx-ingress add-on is not installed. 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
```

View the created ingress.

kubectl get ingress

If information similar to the following is displayed, the ingress has been created successfully and the workload is accessible.


```
NAME           HOSTS     ADDRESS         PORTS   AGE
ingress-test   *         121.**.**.**    80      10s
```

Step 4 Enter **http://*121.**.**.***:80** in the address box of the browser to access the workload.

121.**.**.** indicates the IP address of the unified load balancer.

----End

4.5.3 Container Network Settings

4.5.3.1 Configuring an EIP for a Pod

Scenario

In CCE Autopilot clusters, pods use elastic network interfaces or supplementary network interfaces for networking so you can directly bind EIPs to pods.

To bind an EIP to a pod, simply set the value of the **yangtse.io/pod-with-eip** annotation to **true** when creating the pod. Then, the EIP is automatically allocated and bound to the pod.

Constraints

- To access a pod with an EIP bound from the Internet, you need to add security group rules to allow the Internet traffic to the pod.
- 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 EIP name, deleting, unbinding, or binding the EIP, and changing the billing mode of 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

When creating a pod, set the **pod-with-eip** annotation to **true**. An EIP will be automatically allocated and bound to the pod.

The following uses a Deployment named **nginx** as an example. For details about annotations, see **Table 4-35**.

- 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: 250m
            memory: 500Mi
          requests:
            cpu: 250m
            memory: 500Mi
    imagePullSecrets:
      - name: default-secret
```

Table 4-34 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 with the region. For details, see the EIP console. For example, the following options are available 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

- 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: 250m
              memory: 500Mi
            requests:
              cpu: 250m
              memory: 500Mi
      imagePullSecrets:
        - name: default-secret
```


Table 4-35 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 types vary with regions. For details, see the EIP console.
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 parameter settings for an EIP with a dedicated bandwidth, see **Table 4-34**. • Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	None

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']
...
```

Deleting an EIP with a Pod

When you delete a pod, the EIP automatically allocated to the pod will also be deleted.

4.5.3.2 Configuring a Static EIP for a Pod

Scenario

In CCE Autopilot clusters, static public IP addresses (EIPs) can be assigned to StatefulSets or pods that are created directly.

Constraints

- The static EIP function must be enabled together with the function of automatically allocating an EIP for a pod. For details, see **4.5.3.1 Configuring an EIP for a Pod**.
- Only StatefulSet pods or pods that are created directly can have 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 4-36**.

- 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 4-36 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 4-37 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.
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

Annotation	Mandatory	Default Value	Description	Value Range
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

Table 4-38 Annotations of an EIP with a shared bandwidth

annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to bind an EIP to a pod	**false** or **true**
yangtse.io/eip-network-type	No	5_bgp	EIP type	• 5_bgp • 5_union • 5_sbgp The types vary 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 parameter settings for an EIP with a dedicated bandwidth, see **Table 4-34**. • Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	-

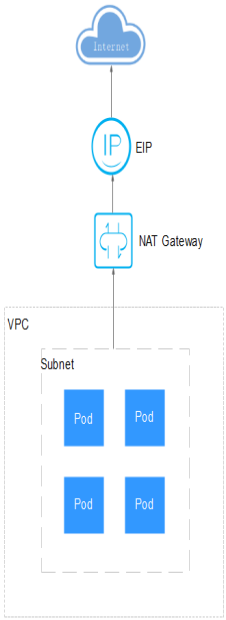
Deleting a Static EIP

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.

4.5.4 Accessing Public Networks from a Container

You can use NAT Gateway to enable the pods in a VPC to access public networks. NAT Gateway provides source network address translation (SNAT), which translates private IP addresses to an EIP bound to the gateway, providing secure and efficient access to the Internet. **Figure 4-32** shows the SNAT architecture. SNAT allows the pods in a VPC to access the Internet without having an EIP bound. SNAT supports a large number of concurrent connections, which makes it suitable for applications that need to handle a large number of requests.

Figure 4-32 SNAT



Procedure

To enable a container pod to access the Internet, perform the following steps:

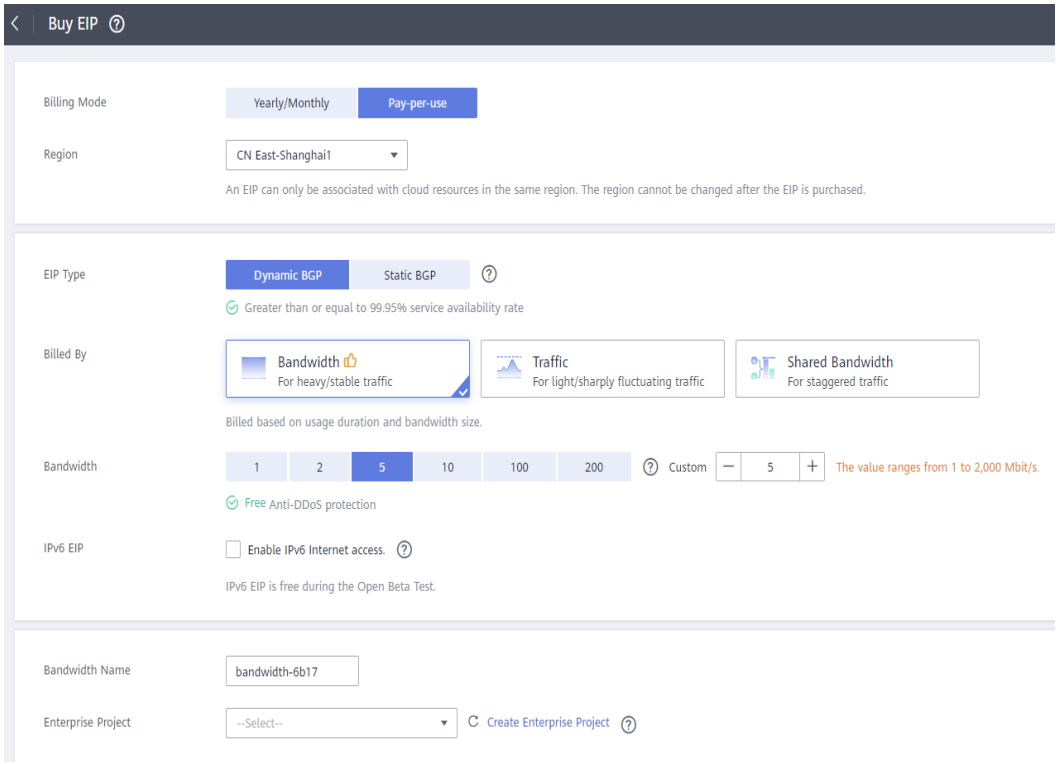
Step 1 Assign an EIP.

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 at 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 4-33 Buying an elastic IP address



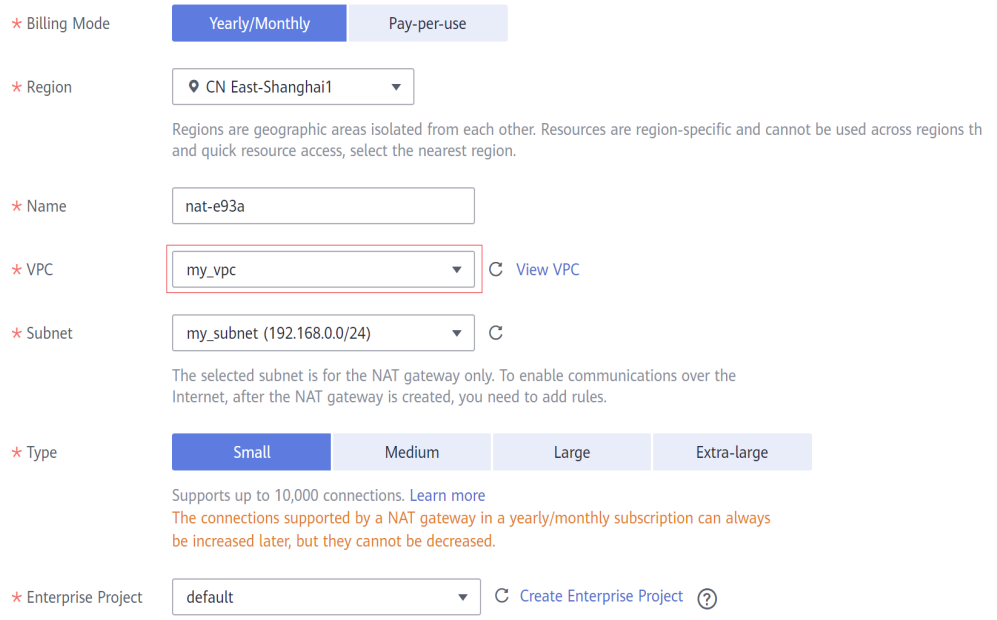
Step 2 Create a NAT gateway. For details, see **Buying a Public NAT Gateway**.

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 at the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
4. On the displayed page, click **Buy Public NAT Gateway** in the upper right corner.
5. Configure parameters as required.

   NOTE

   Select the same VPC.

Figure 4-34 Buying a NAT gateway



Step 3 Configure an SNAT rule and bind the EIP to the subnet. For details, see **Adding an SNAT Rule**.

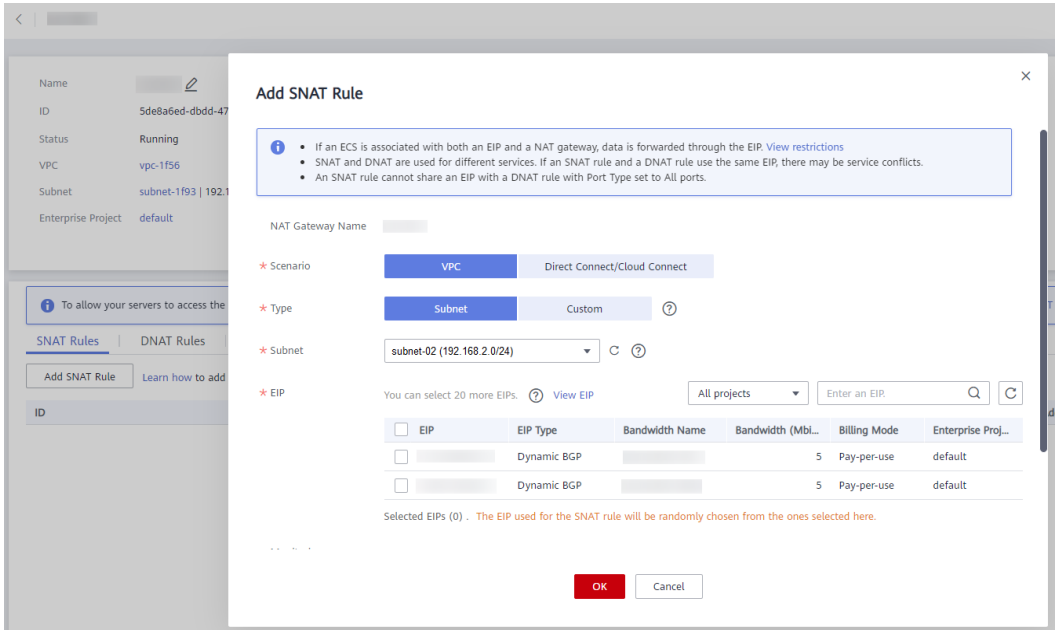
1. Log in to the management console.
2. Click in the upper left corner of the management console and select a region and a project.
3. Click at the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
4. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
5. On the **SNAT Rules** tab page, click **Add SNAT Rule**.
6. Set parameters as required.

NOTE

SNAT rules take effect by network segment. Set **Subnet** to the subnet where the pods are located.

If there are multiple network segments, you can create multiple SNAT rules or select a user-defined network segment as long as the network segment contains the subnet where the pods are located.

Figure 4-35 Adding an SNAT rule



After the SNAT rule is configured, workloads can access public networks from the container. Public networks can be pinged from the container.

----End

4.5.5 Network Management FAQs

4.5.5.1 Configuring Cluster Security Group Rules

When a CCE Autopilot cluster is created, two security groups are automatically created, one for the master nodes, and the other for the elastic network interfaces. The name of the security group for the master nodes is in the format of *{Cluster name}-cce-control-{Random ID}*, and that for the elastic network interfaces is in the format of *{Cluster name}-cce-eni-{Random ID}*.

You can modify the security group rules on the VPC console as required. (Log in to the management console, choose **Service List** > **Networking** > **Virtual Private Cloud**. On the page displayed, choose **Access Control** > **Security Groups** in the navigation pane on the left, locate the corresponding security groups, and modify their rules.)

NOTICE

- Modifying or deleting default rules in a security group may affect cluster running. If you need to modify security group rules, do not modify the rules of the port that CCE running depends on.
- When adding a security group rule, ensure that this rule does not conflict with the existing rules. If there is a conflict, existing rules may become invalid, affecting cluster running.

Security Group of the Master Nodes

A security group named *{Cluster name}*-**cce-control-***{Random ID}* is automatically created for the master nodes. **Table 4-39** lists the default ports in the security group.

Table 4-39 Default ports in the security group of the master nodes

Direction	Port	Source	Description	Modifiable	Modification Suggestion
Inbound	All	Security group of the master nodes	Allow traffic from all IP addresses in the security group	No	N/A
	All	VPC CIDR block	Allow traffic from all IP addresses in the VPC CIDR block	No	N/A
Outbound	All	All IP addresses: **0.0.0.0/0**	Allow traffic from the masters nodes to any IP address over any port.	No	N/A

Security Group of the Elastic Network Interfaces

When a CCE Autopilot cluster is created, an additional security group named *{Cluster name}*-**cce-eni-***{Random ID}* is also created. By default, pods in the cluster are associated with this security group. **Table 4-40** lists the default ports in the security group.

Table 4-40 Default ports in the security group for the elastic network interfaces

Direction	Port	Source	Description	Modifiable	Modification Suggestion
Inbound	All	Security group of the elastic network interfaces	Allow traffic from all IP addresses in this security group.	No	N/A

Direction	Port	Source	Description	Modifiable	Modification Suggestion
		VPC CIDR block	Allow traffic from all IP addresses in the VPC CIDR block	No	N/A
		CIDR block of the master nodes	Allow the master nodes to access kubelet on each worker node, for example, by running **kubectl exec** *{pod}*.	No	N/A
Outbound	All	All IP addresses: **0.0.0.0/0**	Allow traffic from the elastic network interface to any IP address over any port.	Yes	If you want to harden security by allowing traffic over specific ports, you can modify the rule to allow these ports. For details, see **Hardening Outbound Rules for the Security Group of the Elastic Network Interfaces**.

Hardening Outbound Rules for the Security Group of the Elastic Network Interfaces

By default, all security groups created by CCE allow all outbound traffic. You are advised to retain this configuration. If you want to harden security by allowing traffic over specific ports, configure the ports listed in the following table.

Table 4-41 Minimum number of outbound rules

Port	Allowed CIDR Block	Description
All ports	Security group of the elastic network interfaces	Allow mutual access within the security group so containers can communicate with each other.

Port	Allowed CIDR Block	Description
TCP port 5443	VPC CIDR block	Allow access from kube-apiserver, which provides lifecycle management for Kubernetes resources.
TCP port 443	100.125.0.0/16	Access the OBS port or SWR port to pull images.
UDP port 53	100.125.0.0/16	Allow traffic over the port for DNS resolution.
TCP port 443	VPC CIDR block	Pull the images through the SWR endpoint.
All ports	198.19.128.0/17	Allow access to VPC Endpoint.
TCP port 9443	VPC CIDR block	Allow the network add-on on the worker nodes to access the master nodes.
All ports	198.19.128.0/17	Allow access to VPC Endpoint.

4.6 Storage

4.6.1 SFS Turbo

4.6.1.1 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 a 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:

- **4.6.1.2 Using an Existing SFS Turbo File System Through a Static PV**: static mode. You use an existing file system to create a PV and then mount the PV to the workload through a PVC.
- **4.6.1.4 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System**: 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, see **Billing**.

4.6.1.2 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 and implement data persistence and sharing in workloads.

Prerequisites

- You have created an available SFS Turbo file system, and the SFS Turbo file system and the cluster are in the same VPC.
- If you want to create a cluster using commands, use kubectl to connect to the cluster. For details, see **4.3.4.1 Connecting to a Cluster Using kubectl**.

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. In the navigation pane on the left, choose **Storage**. Then click the **PVCs** tab. Click **Create PVC** in the upper right corner. In the dialog box displayed, configure the PVC parameters.

Parameter	Description
PVC Type	In this example, select **SFS Turbo**.
PVC Name	Enter the PVC name, which must be unique in the same namespace.
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 a PV and PVC at the same time on the console.
PV[a]	Select an existing PV in the cluster. Create a PV in advance. For details, 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**.
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 supported, indicating that the PV is not deleted when the PVC is deleted. For details, see **PV Reclaim Policy**.

Parameter	Description
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **4.6.1.3 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 on the left and view the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create an application.

1. In the navigation pane on the left, choose **Workloads**. 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**.

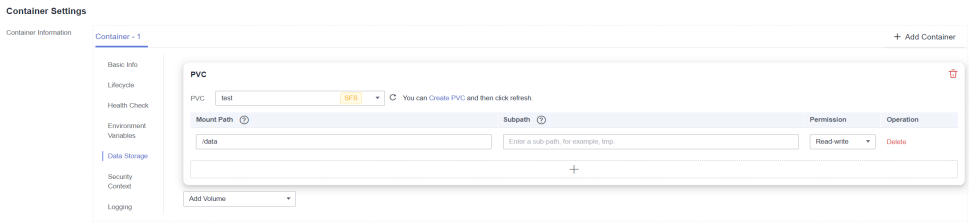
   Table 4-42 describes the parameters for mounting the volume. For details about other parameters, see **4.4 Workloads**.

Table 4-42 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS Turbo volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the 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 cause 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, which leads to a container startup failure or workload creation failure. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container 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. Enter a subpath, for example, **tmp**, indicating that data in the mount path of the container is stored in the **tmp** directory 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 volume 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 volume 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

(kubectl) Using an Existing SFS File System

Step 1 Use kubectl to connect to the cluster.

Step 2 Create a PV.

1. Create the **pv-sfsturbo.yaml** file.

```
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          # Storage class name of the SFS Turbo file system.
mountOptions: []                        # Mount options.
```

Table 4-43 Key parameters

Parameter	Mandatory	Description
volumeHandle	Yes	SFS Turbo volume ID. How to obtain: Log in to the console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. In the list, click the name of the target SFS Turbo volume. 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 console, choose **Service List** > **Storage** > **Scalable File Service**, and select **SFS Turbo**. You can obtain the shared path of the file system from the **Mount Address** column.
everest.io/enterprise-project-id	No	Project ID of the SFS Turbo volume. How to obtain: On the SFS console, click **SFS Turbo** in the left 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 **3.8.5.3 Configuring SFS Turbo Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	Only **Retain** is supported. For details, see **PV Reclaim Policy**. **Retain**: When a PVC is deleted, the PV and underlying storage resources are not deleted. Instead, you must manually delete these resources. After that, the PV is in the **Released** status and cannot be bound to the PVC again.
storage	Yes	Requested capacity in the PVC, in Gi.
storageClassName	Yes	The storage class 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      # Storage class of the SFS Turbo volume, which must be the same as that of the PV.
  volumeName: pv-sfsturbo    # PV name.
```

Table 4-44 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, click **SFS Turbo** in the left 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.

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	Storage class name, which must be the same as the storage class of the PV in **Step 2.1**. The storage class name of SFS Turbo volumes is **csi-sfsturbo**.
volumeName	Yes	PV name, which must be the same as the PV name in **Step 2.1**.

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 view 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 4-45**.

Table 4-45 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	Create a PV on the CCE console.	1. In the navigation pane on the left, choose **Storage**. Then click the **PVs** tab. Click **Create Volume** in the upper right corner. In the dialog box displayed, configure the parameters. • **Volume Type**: Select **SFS Turbo**. • **SFS Turbo**: Click **Select SFS Turbo**. On the page displayed, select the SFS Turbo volume that meets the requirements and click **OK**. • **PV Name**: Enter the PV name, which must be unique in the same cluster. • **Access Mode**: Only **ReadWriteMany** is available. 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. For details, see **PV Reclaim Policy**. • **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **4.6.1.3 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. In the navigation pane on the left, choose **Storage**. Then 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	You can view event names, event types, number of occurrences, Kubernetes events, first occurrence time, and last occurrence time of the PVC or PV.	1. In the navigation pane on the left, choose **Storage**. Then 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 (event data is retained for one hour).

Operation	Description	Procedure
Viewing a YAML file	You can view, copy, and download the YAML files of a PVC or PV.	1. In the navigation pane on the left, choose **Storage**. Then 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.

4.6.1.3 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.

Constraints

Due to the restrictions of NFS, if an SFS volume is mounted to a node for multiple times, link-related mount options (such as **timeo**) take effect only when the SFS volume is mounted for the first time. For example, if an SFS volume is mounted to multiple pods running on a node, the values of the mount options configured later will not overwrite the existing parameter values.

SFS Turbo Mount Options

CCE Autopilot presets the options described in **Table 4-46** for mounting SFS Turbo volumes.

Table 4-46 SFS Turbo mount options

Parameter	Value	Description
vers	3	File system version. Currently, only NFS v3 is supported. Value: **3**
nolock	Leave it 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	Leave it 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**.

You can set 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 connect to the cluster. For details, see **4.3.4.1 Connecting to a Cluster Using kubectl**.

Step 2 Set 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           # SFS Turbo storage class 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 **4.6.1.2 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=tcp,local_lock=all,addr=**.**.**.**)
```

----End

4.6.1.4 Dynamically Creating and Mounting Subdirectories of an SFS Turbo File System

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.

CCE Autopilot allows you to dynamically create subdirectories in an SFS Turbo file system and mount these subdirectories to containers so that SFS Turbo file systems can be shared and the SFS Turbo storage capacity can be used more economically and properly.

Creating an SFS Turbo Volume of the subpath Type

CAUTION

Do not expand, disassociate, or delete a **subpath** volume.

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
mountOptions:
- 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/
  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
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.

The following configuration is used by default. For details, see **Setting Mount Options**. Do not set **nolock** to **true**. If you do this, 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 subdirectory. The shared path can be obtained on the SFS Turbo console. The subdirectory is user-defined. The PVCs created using the StorageClass are located in this subdirectory.
- **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
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: sfsturbo-subpath-sc
  volumeMode: Filesystem
```


In this example:

- **name**: indicates the name of the PVC.
- **storageClassName**: specifies the name of the StorageClass created in the previous step.
- **storage**: In the subpath mode, it is useless to specify this parameter. The storage capacity is limited by the total capacity of the SFS Turbo file system. If the total capacity of the SFS Turbo file system is insufficient, expand the capacity on the SFS Turbo page in a timely manner.

Step 5 Run the **kubectl create -f sfs-turbo-test.yaml** command to create a PVC.

----End

NOTE

It is meaningless to conduct capacity expansion on an SFS Turbo volume created in the subpath mode. This operation does not expand the capacity of the SFS Turbo file system. Ensure that the total capacity of the SFS Turbo file system is not used up.

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
  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
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-sfs-turbo-example
      restartPolicy: Always
      imagePullSecrets:
      - name: default-secret
      volumes:
      - name: pvc-sfs-turbo-example
        persistentVolumeClaim:
          claimName: sfs-turbo-test
```

In this example:

- **name**: indicates the name of the Deployment.
- **image**: specifies the image used by the Deployment.
- **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
  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'
          resources: {}
          volumeMounts:
            - name: sfs-turbo-160024548582479676
              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: sfs-turbo-160024548582479676
        namespace: default
        annotations: {}
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
        storageClassName: sfsturbo-subpath-sc
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
```



```
  type: RollingUpdate
revisionHistoryLimit: 10
```

In this example:

- **name**: indicates the name of the StatefulSet.
- **image**: specifies the image used by the StatefulSet.
- **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**: indicates the name of the StorageClass.

Step 2 Create the StatefulSet.

kubectl create -f statefulset-test.yaml

----End

4.6.2 emptyDir

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 the Console to Use a Temporary 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 on the left, choose **Workloads**. In the right pane, click the **Deployments** tab.

Step 3 In the upper right corner of the page, click **Create Workload**. In the **Container Settings** area, click the **Data Storage** tab, click **Add Volume**, and then select **EmptyDir**.

Step 4 Table 4-47 describes the parameters for mounting the volume. For details about other parameters, see 4.4.1 Creating a Workload.

Table 4-47 Parameters for mounting an emptyDir volume

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 option is suitable when data volume is small and efficient read and write is required. • If this option is not selected, data is stored in local disks. This fits to the scenarios where a large amount of data needs to be stored, with low requirements for reading and writing efficiency. **NOTE** • If **Memory** is selected, pay attention to the memory size. If the storage capacity exceeds the memory size, OOM will occur. • If **Memory** is selected, the size of an emptyDir volume is the same as the pod specifications. • If **Memory** is not selected, emptyDir volumes will not occupy the system memory.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the 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 cause 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, which leads to a container startup failure or workload creation failure. **NOTICE** If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container, or 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. Enter a subpath, for example, **tmp**, indicating that data in the mount path of the container is stored in the **tmp** directory 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 volume 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 kubectl to Use a Temporary Path

Step 1 Use kubectl to connect to the cluster. For details, see **4.3.4.1 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        # Path to which an EV 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

4.7 Observability

4.7.1 Monitoring Center

4.7.1.1 Enabling Cluster Monitoring

To enable monitoring for a cluster, you need to install the Cloud Native Cluster Monitoring add-on for metric collection. After cluster monitoring is enabled, cluster metrics are collected and reported to the selected AOM instance.

NOTICE

- After cluster monitoring is enabled, cluster metrics are reported to the selected AOM instance. Basic metrics are free of charge but custom metrics are charged by AOM. For details, see **Pricing Details**.
- Running the Cloud Native Monitoring add-on in a cluster consumes cluster resources. Ensure that there are required cluster resources for installing the add-on. To view 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.

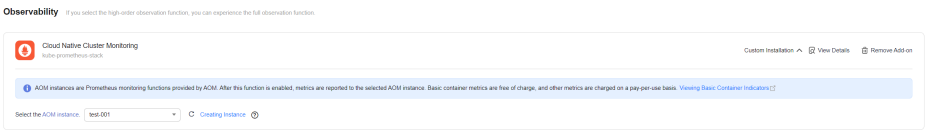
Constraints

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.

Enabling Cluster Monitoring

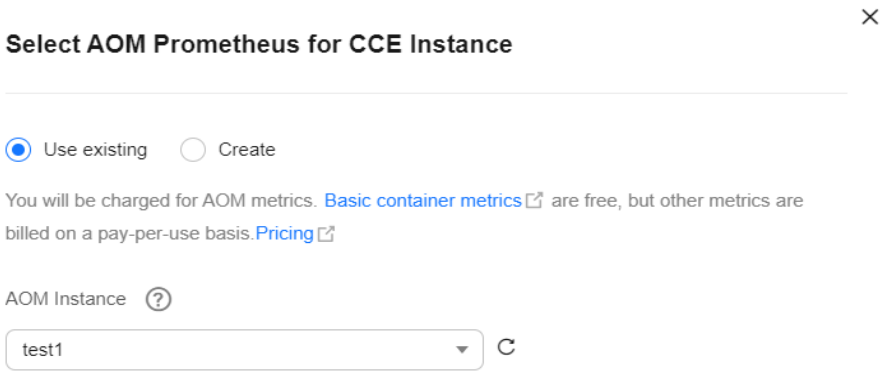
- **Enabling cluster monitoring during cluster purchase**
  a. Log in to the CCE console and purchase a cluster.
  b. On the **Select Add-on** page, select the **Cloud Native Cluster Monitoring** add-on.
  c. On the **Add-on Configuration** page, select the AOM instance to be interconnected with the add-on. If there is no access code, create one first.

  Figure 4-36 Enabling cluster monitoring

- **Enabling cluster monitoring on the Monitoring Center page**
  a. Click the cluster name. In the navigation pane on the left, choose **Monitoring Center**.
  b. Click **Enable** and then select the AOM instance that metrics are reported to.

Figure 4-37 Enabling cluster monitoring



c. Wait for 3 to 5 minutes until the monitoring data is reported to the AOM instance.

   The functions of Monitoring Center are available.

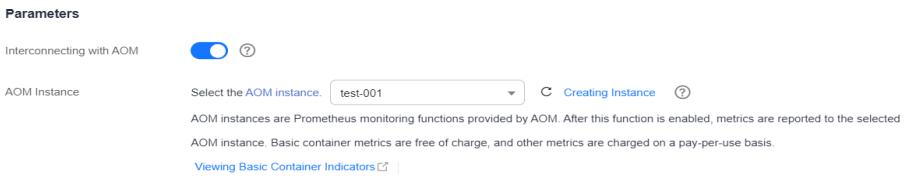
- **Enabling cluster monitoring on the Add-ons page**

  a. Click the cluster name and choose **Add-ons** in the navigation pane.

  b. Select the Cloud Native Cluster Monitoring add-on and click **Install**.

  c. Enable the option for interconnecting with AOM to report the metrics to the selected AOM instance.

  Figure 4-38 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 cluster monitoring, uninstall the Cloud Native Cluster Monitoring add-on on the **Add-ons** page or disable the option for interconnecting with AOM.

4.7.1.2 Container Insights

4.7.1.2.1 Clusters

Container Insights allows you to observe resource usages and cluster health. On the **Clusters** tab, you can view the monitoring data of the cluster from **Resource 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.

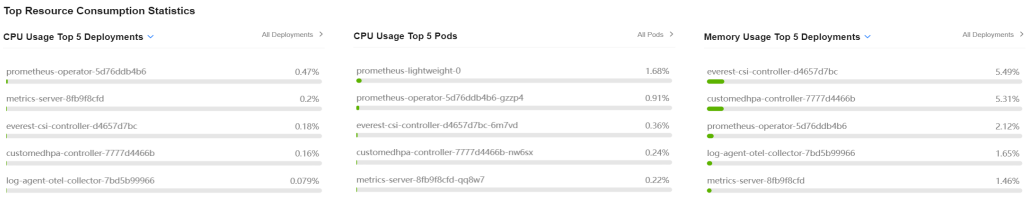
Step 2 In the navigation pane on the left, choose **Monitoring Center**. Then click **Container Insights** > **Clusters**.

----End

Resource Overview

Resource Overview displays the percentages of abnormal resources in workloads and pods and the total number of namespaces.

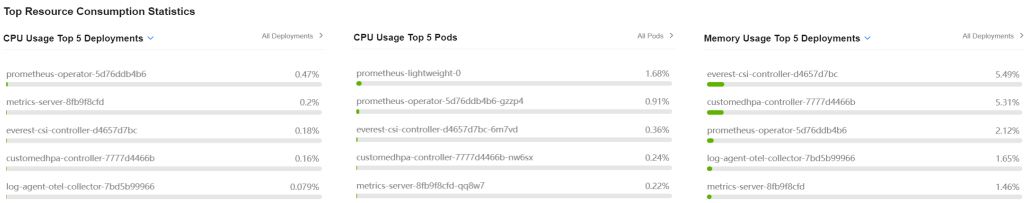
Figure 4-39 Resource Overview



Top Resource Consumption Statistics

CCE collects statistics on top 5 Deployments, StatefulSets, and pods by CPU and memory usages, helping you identify workloads with high resource consumption. To view all data, click the **Workloads** or **Pods** tab.

Figure 4-40 Top Resource Consumption Statistics



Monitoring metrics

- CPU Usage

  Workload CPU usage = Average CPU usage in each pod of the workload

  Pod CPU usage = Number of CPU cores used by a pod/Sum of workload container CPU limits

- Memory Usage

  Workload memory usage = Average memory usage in each pod of the workload

  Pod memory usage = Memory used by a pod/Sum of workload container memory limits

Data Plane Monitoring

By default, resource usages are collected from each dimension in the last hour, last 8 hours, and last 24 hours. To view more monitoring data, click **View All Metrics** to access the **Dashboard** page. For details, see **4.7.1.3.1 Using Dashboard**.

NOTE

You can hover over a chart to view the monitoring data in each minute.

- **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.

4.7.1.2.2 Workloads

On the **Workloads** tab, you can view the resource usages of workloads. This page provides information about all workloads in a cluster and monitoring data of a single workload, such as the CPU/memory usage and network inbound/outbound rate.

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 on the left, choose **Monitoring Center**. On the **Container Insights** tab, click **Workloads**.

Information about all workloads is displayed. To view the monitoring data of a workload, click the workload name to go to the **Overview** tab. You can also click **Pods** or **Monitoring** to view corresponding information.

----End

Workload List

You can view the name, status, number of normal pods, number of total pods, namespace, image, used CPUs, used memory, CPU usage, and memory usage of each workload.

Figure 4-41 Workloads

You can select a workload type in the upper right corner, or select **Workload name**, **Status**, and **Namespace** above the list to quickly locate the required workload.

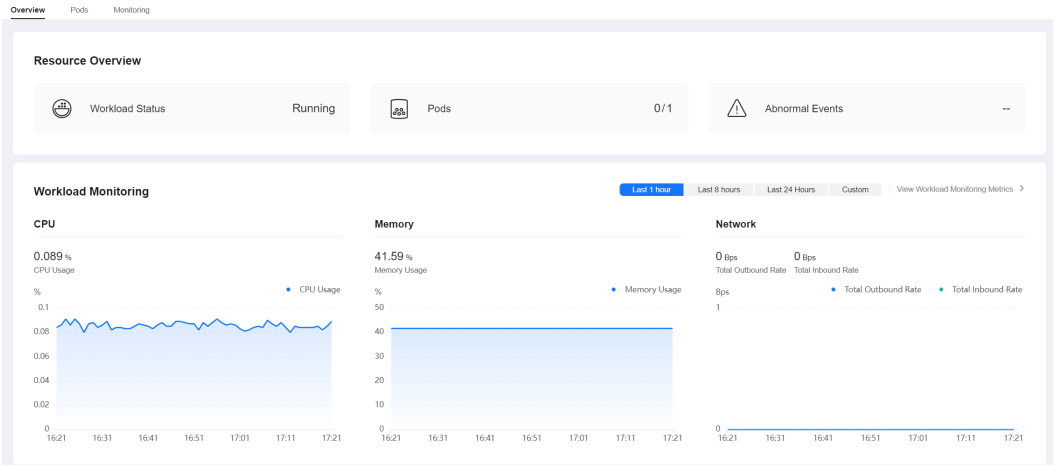
You can click **Export** in the upper right corner of the list to export all workloads or the selected workloads. The exported file is in .xlsx format, and the file name contains the timestamp.

Overview

You can click the workload name to view the resource overview, such as the workload status, number of normal pods, number of total pods, and abnormal

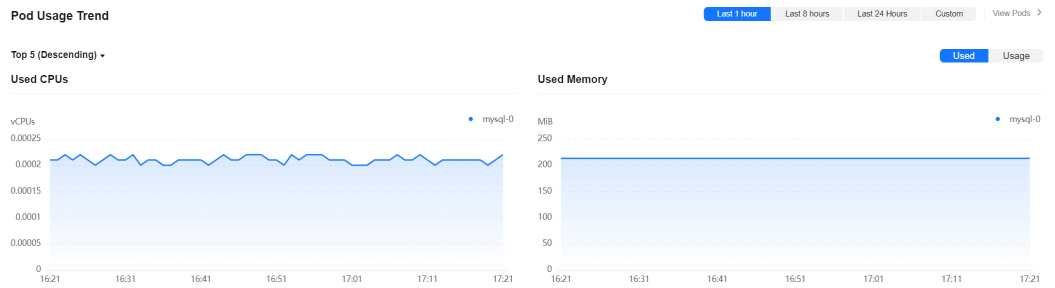
events. You can also view the monitoring overview of the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 4-42 Resource overview and monitoring overview



The **Overview** tab also shows the pod usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each pod of the workload. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left corner to view the top 5 data in descending or ascending order.

Figure 4-43 Pod usage trend

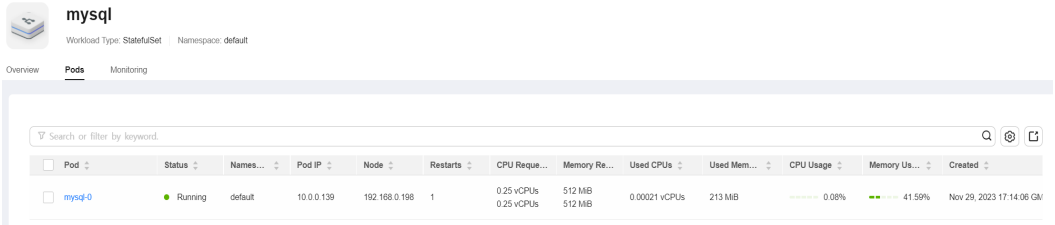


For more metrics, go to the **Monitoring** tab.

Pods

You can view the name, status, namespace, IP address, node, number of restarts, CPU request, CPU limit, memory request, memory limit, used CPUs, used memory, CPU usage, and memory usage of each pod.

Figure 4-44 Pods



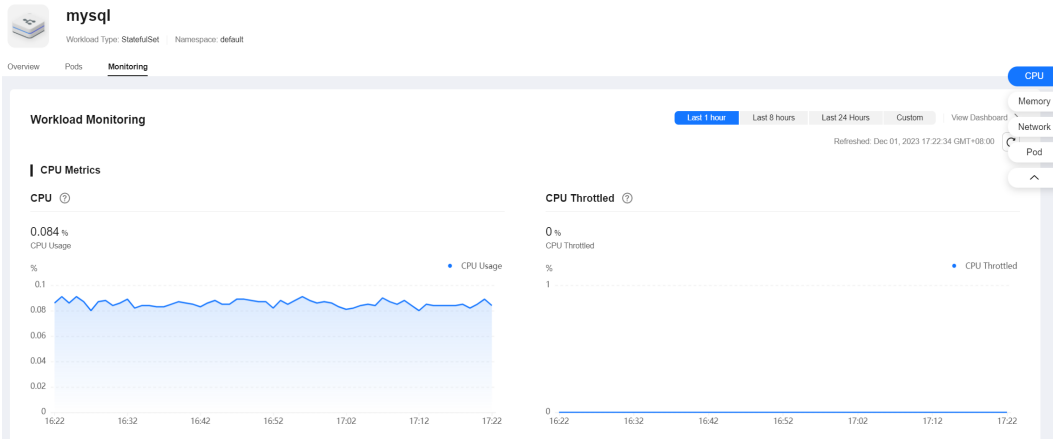
You can find the desired pod by name, status, namespace, IP address, or node. You can click **Export** in the upper right corner of the list to export data of all pods or the selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **4.7.1.2.3 Pod List**.

Monitoring

This tab shows 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 the **Dashboard** page. For details, see **4.7.1.3.1 Using Dashboard**.

Figure 4-45 Workload monitoring



- **CPU Metrics**
  - CPU usage: the percentage of the CPU used by containers in all pods of the workload in different time periods with respect to the total CPU limit for all containers.
  - CPU throttled: the average percentage of time duration containers have been throttled in all pods of the workload in different time periods.
- **Memory Metrics**
  - Memory usage: the percentage of memory used by containers in all pods of the workload in different time periods with respect to the total memory limit for all containers.
- **Networking Metrics**
  - Total outbound rate: the total number of bytes transmitted by containers in all pods of the workload per second in different time periods.
  - Total inbound rate: the total number of bytes received by containers in all pods of the workload per second in different time periods.
  - Packet loss rate (transmit): the percentage of packets not received by the recipient to packets sent from containers in all pods of the workload in different time periods.

  - Packet loss rate (receive): the percentage of packets not received by containers in all pods of the workload to packets sent to the containers in different time periods.
- **Pod Metrics**
  - Pod CPU usage: the percentage of CPU used by each pod of the workload in different time periods with respect to the CPU limit for each pod.
  - Pod memory usage: the percentage of memory used by each pod of the workload in different time periods with respect to the memory limit for each pod.
  - Pod status and quantity: the total number of pods in the **Unavailable**, **Unready**, **Running**, **Completed**, or **Other** state of the workload in different time periods.
  - Pod quantity trend: the number of pods (replicas) of the workload in different time periods.

4.7.1.2.3 Pod List

To view the resource usages of pods, go to the **Pods** tab, where you can view information about all pods in a cluster and monitoring data of each pod, such as the CPU usage, memory usage, inbound rate, and outbound rate.

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 on the left, choose **Monitoring Center**. Then click **Container Insights** > **Pods**.

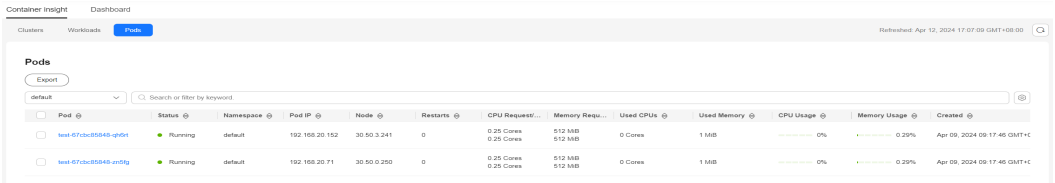
Information about all pods is displayed. To view the monitoring data of a pod, click the pod name to go to the **Overview** tab. You can also click the **Containers** or **Monitoring** tab to view the corresponding information.

----End

Pods

You can view the name, status, namespace, IP address, node, number of restarts, CPU request, CPU limit, memory request, memory limit, used CPUs, used memory, CPU usage, and memory usage of each pod.

Figure 4-46 Pods



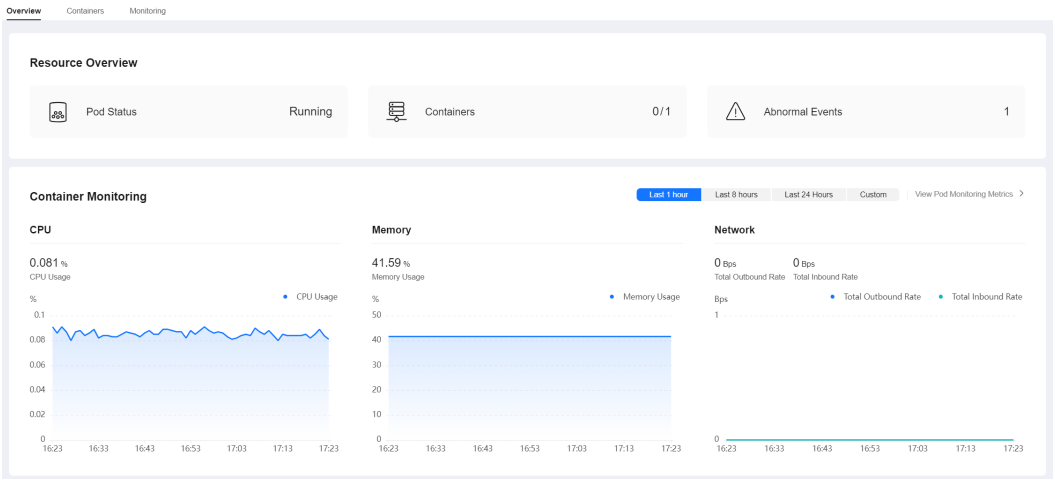
You can select a namespace in the upper right corner, or select **Pod**, **Status**, **Namespace**, **Pod IP**, and **Node** above the list to quickly locate the required pod.

You can click **Export** to export data of all pods or the 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 node in the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 4-47 Resource overview and monitoring overview



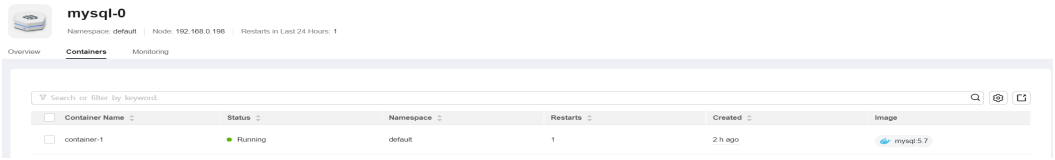
The **Overview** tab also shows the container usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each container in the pod. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left corner to view the top 5 data in descending or ascending order.

For more metrics, go to the **Monitoring** tab.

Containers

This tab contains details such as the name, status, namespace, number of restarts, and image of each container.

Figure 4-48 Containers



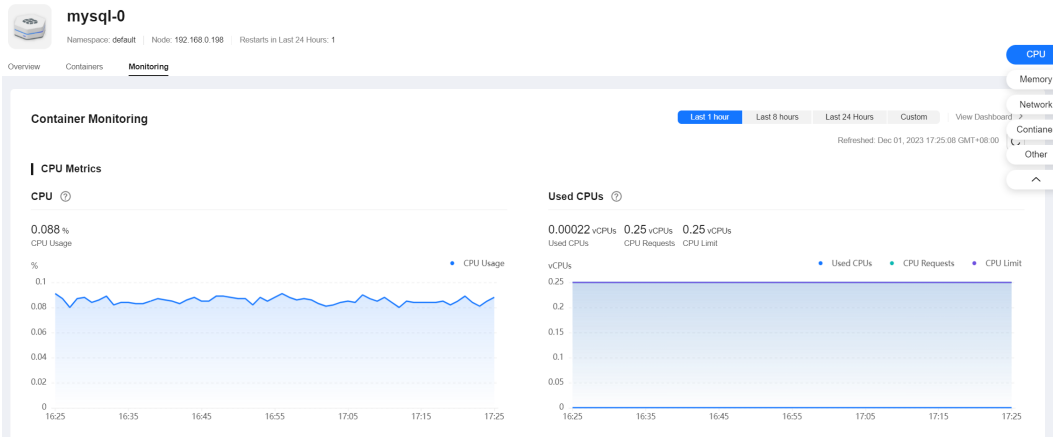
You can find the desired container by name, status, or namespace. You can click **Export** in the upper right corner of the list to export data of all containers or the selected containers. The exported file is in .xlsx format, and the file name contains the timestamp.

Monitoring

This tab shows the resource usage of the pod in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring data,

click **View Dashboard** to access the **Dashboard** page. For details, see **4.7.1.3.1 Using Dashboard**.

Figure 4-49 Pod monitoring



- **CPU Metrics**
  - CPU usage: the percentage of CPU used by all containers in a pod in different time periods with respect to the total CPU limit for all containers.
  - Used CPU: the CPU that the pod is using.
  - CPU request: the CPU requested for the pod.
  - CPU limit: the CPU limit configured for 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 percentage of memory used by all containers in the pod in different time periods with respect to the total memory limit for all containers.
  - Used memory: the memory that the pod is using.
  - Memory request: the memory requested for the pod.
  - Memory limit: the memory limit configured for the pod. When the used memory is close to this limit, OOM will occur.
- **Networking Metrics**
  - Total outbound rate: the total number of bytes transmitted 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 percentage of CPU used by each container in the pod in different time periods with respect to the CPU limit for each container.
  - Container memory usage: the percentage of memory used by each container in the pod in different time periods with respect to the memory limit for each container.

  - Container CPU throttled: the percentage of time duration each container has been throttled in different time periods.
  - Container network packet loss rate: the percentage of packets not received by each container of the pod to packets sent to the container in different time periods.
- **Other Metrics**
  - Historical pod status: the status of the pod in different periods.
  - Historical container status: the status of each container in the pod in different time periods.

4.7.1.3 Dashboard

4.7.1.3.1 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 and pod view.

Prerequisites

- 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 on the left, 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 the parameters.

Step 5 Specify the view window.

Select or customize time periods in the upper right corner of the page, and click  to refresh the page.

----End

4.7.2 Logging

4.7.2.1 Collecting Logs

Cloud Native Logging is an add-on based on Fluent Bit and OpenTelemetry for collecting logs and Kubernetes events. This add-on supports CRD-based log

collection policies, and collects and forwards standard output logs, container file logs, and Kubernetes events in a cluster. It also reports all abnormal Kubernetes events and some normal Kubernetes events to AOM.

Constraints

- A maximum of 50 log collection rules can be configured for each cluster.
- This add-on cannot collect .gz, .tar, and .zip logs or access symbolic links of logs.
- In each cluster, up to 10,000 single-line logs can be collected per second, and up to 2,000 multi-line logs can be collected per second.
- 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.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You only pay for log volume that exceeds the quota. For details, see **Price Calculator**.

Log Collection

Step 1 Enable log collection.

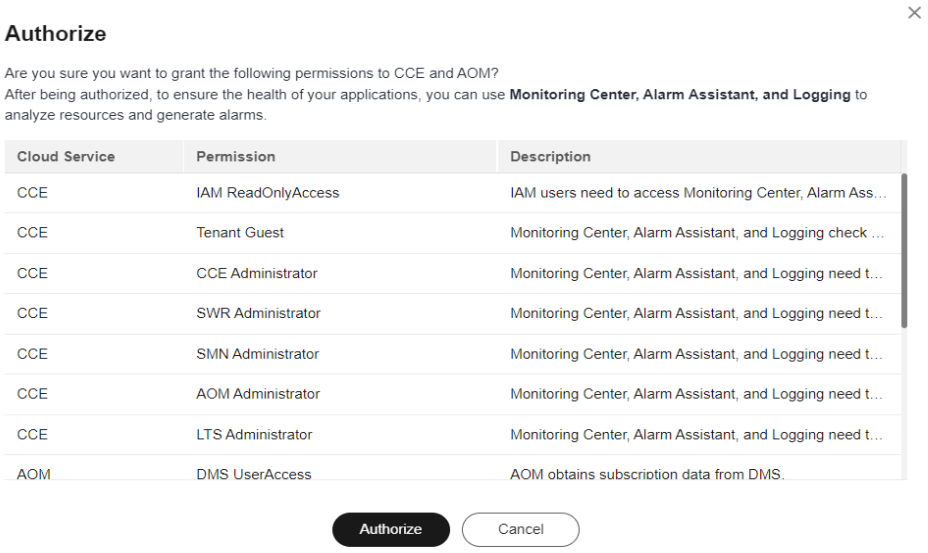
Enabling log collection during cluster creation

1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**.
3. On the **Select Add-on** page, select **Cloud Native Logging**.
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, and standard output logs in all namespaces will be reported to LTS.
   - Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.
5. Click **Next: Confirm Configuration**. On the displayed page, click **Submit**.

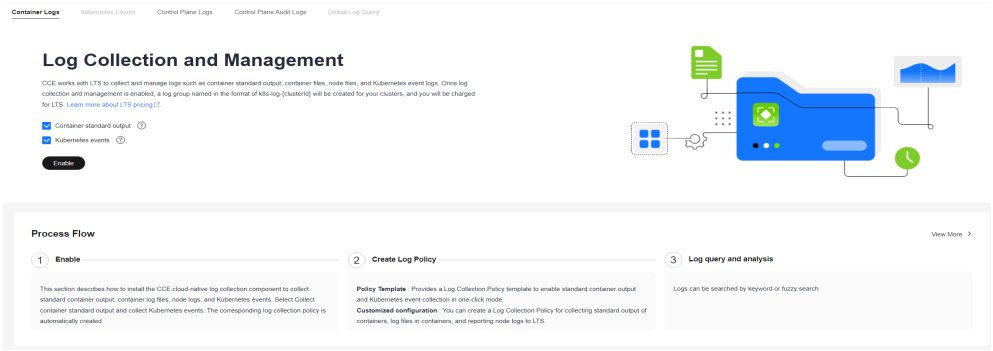
Enabling log collection for an existing cluster

1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.
2. (Optional) If you are not authorized, obtain required permissions first.

   In the displayed dialog box, click **Authorize**.



3. Click **Enable** and wait for about 30 seconds until the log page is automatically displayed.



Standard output logs: A log collection policy named **default-stdout** will be created, and standard output logs in all namespaces will be reported to LTS.

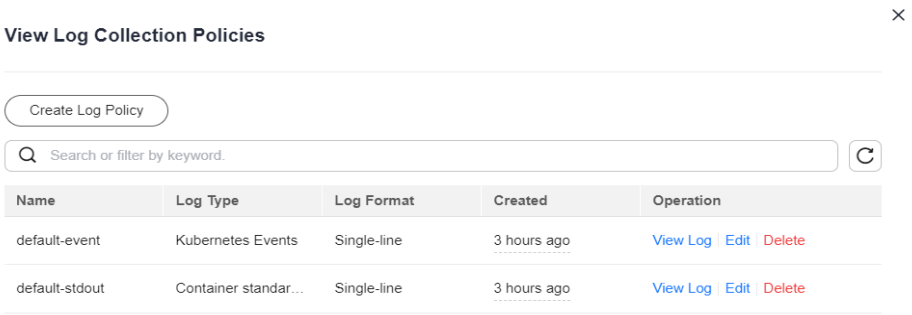
Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.

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 on the left, choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.

Figure 4-50 Viewing log collection policies



If **Container standard output** and **Kubernetes events** are selected during add-on installation, two log collection policies will be created, and the collected logs will be reported to the default log group and log streams.

- default-stdout

  Standard output logs are collected to the default log stream **stdout-***{Cluster ID}* in the default log group **k8s-logs-***{Cluster ID}*.

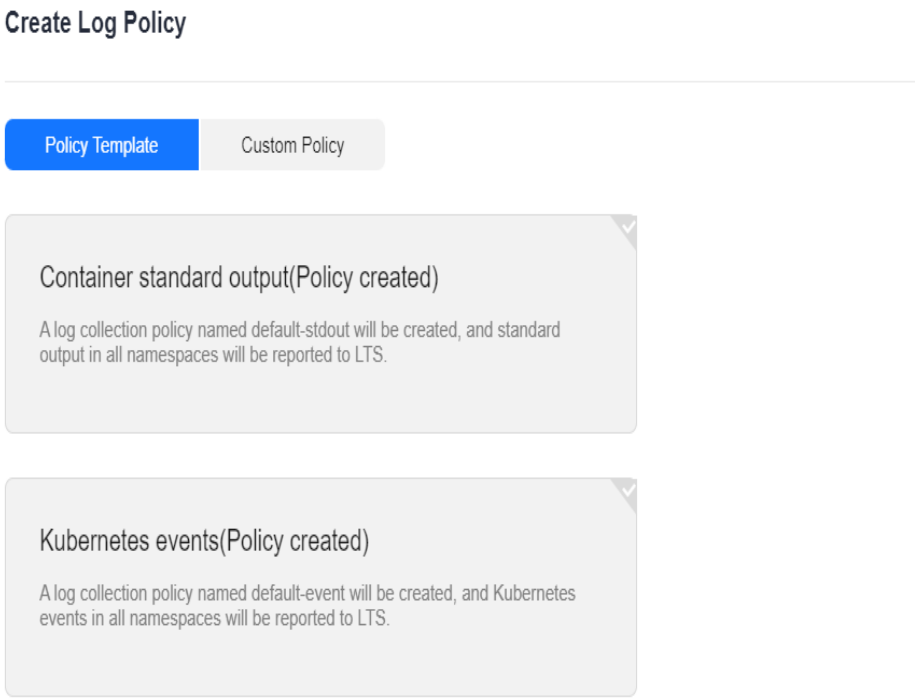
- default-event

  Kubernetes events are collected to the default log stream **event-***{Cluster ID}* in the default log group **k8s-logs-***{Cluster ID}*.

3. Click **Create Log Policy** and configure parameters as required.

   Policy Template: If **Container standard output** and **Kubernetes events** are not selected during add-on installation or their log collection policies are deleted, you can use this option to create default log collection policies.

   Figure 4-51 Policy template

   Custom Policy: You can use this option to create custom log collection policies.

Figure 4-52 Custom policy

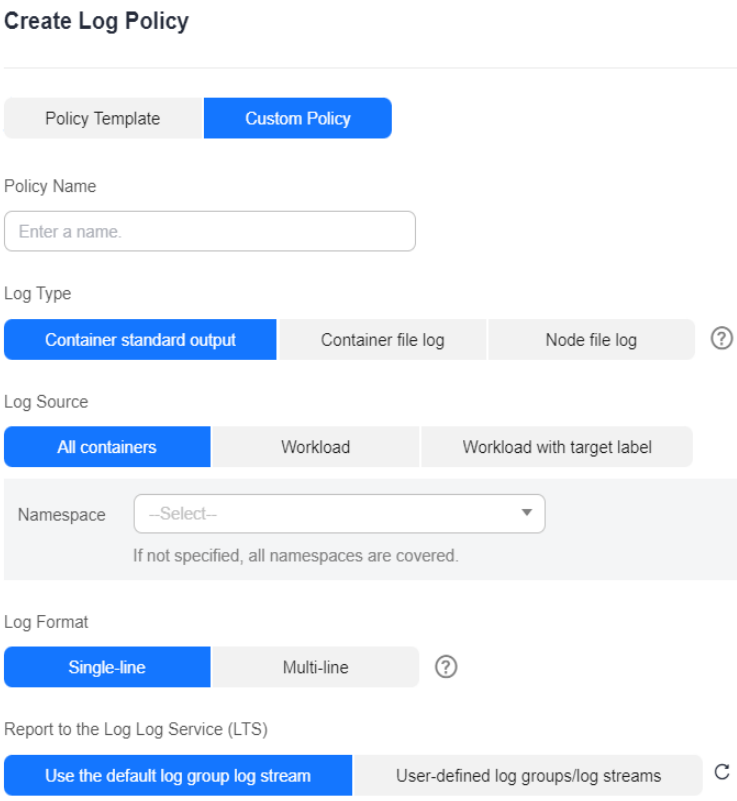


Table 4-48 Custom policy parameters

Parameter	Description
Log Type	Type of logs to be collected. – **Container standard output**: used to collect container standard output 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 create a log collection policy by workload or instance label.
Log Source	Containers whose logs are to be collected. – **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.

Parameter	Description
Collection Path	Path of files where logs are to be collected. 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. The file name can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), asterisks (*), question marks (?), and periods (.). Enter an absolute path for the log directory. Logs in the format of .gz, .tar, or .zip are not supported. A maximum of three levels of directories can be matched using wildcards. The level-1 directory does not support wildcards. The directory name and file name must be complete names and support asterisks (*) and question marks (?) as wildcards. An asterisk (*) can match multiple characters. A question mark (?) can match only one character. Example: – If the directory is **/var/logs/*** and the file name is ***.log**, any log files with the extension .log in all directories in the **/var/logs** directory will be reported. – 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. 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. 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**.

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. If you select the multi-line text, you need to enter the log matching format.
	Example:
	If logs need to be collected by line, enter **\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.***.
	The following 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)`

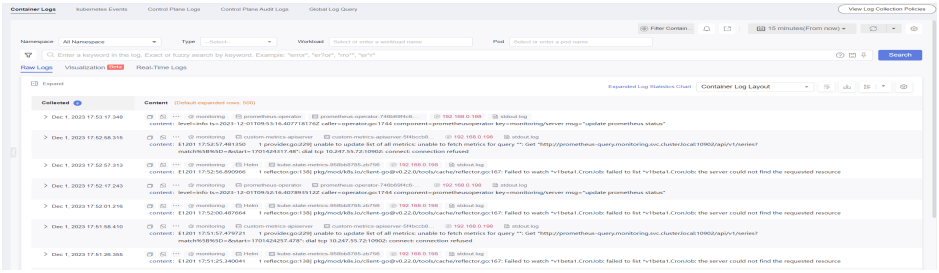
Report to the Log Tank Service (LTS)	This parameter is used to configure the log group and log stream for log reporting. – Default log groups/log streams: The default log group (**k8s-log-*{Cluster ID}***) and default log stream (**stdout-*{Cluster ID}***) are automatically selected. – Custom log groups/log 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 log read and write. You can create log streams in a log group to store different types of logs for finer log management. When you install the add-on or create a log policy based on a template, the following log streams are automatically created: – **stdout-*{Cluster ID}*** for standard output logs, for example, **stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3** – **event-*{Cluster ID}*** for Kubernetes events, for example, **event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**

4. Click **Edit** to modify an existing log collection policy.
5. Click **Delete** to delete an existing log collection policy.

Step 3 View the logs.

1. On the CCE console, click the cluster name to access the cluster console. In the navigation pane on the left, 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 4-53 Querying container logs

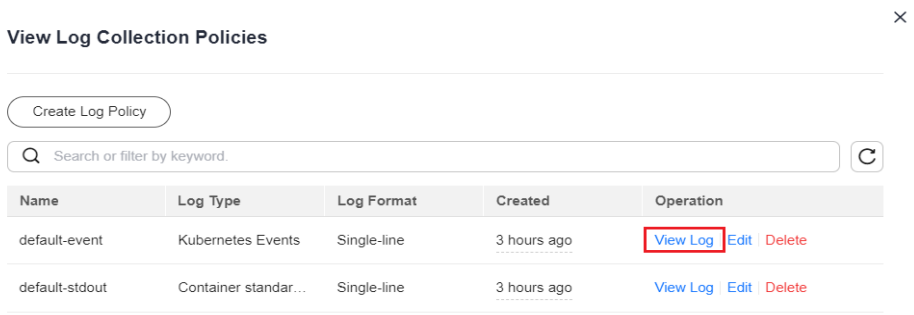
   - **Kubernetes Events**: displays all Kubernetes events in the default log stream **event-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   - **Global Log Query**: You can view logs in the log streams of all log groups. You can specify a log stream to view the logs. By default, 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 4-54 Global log query

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 4-55 Viewing logs

----End

Troubleshooting

- **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
  ```

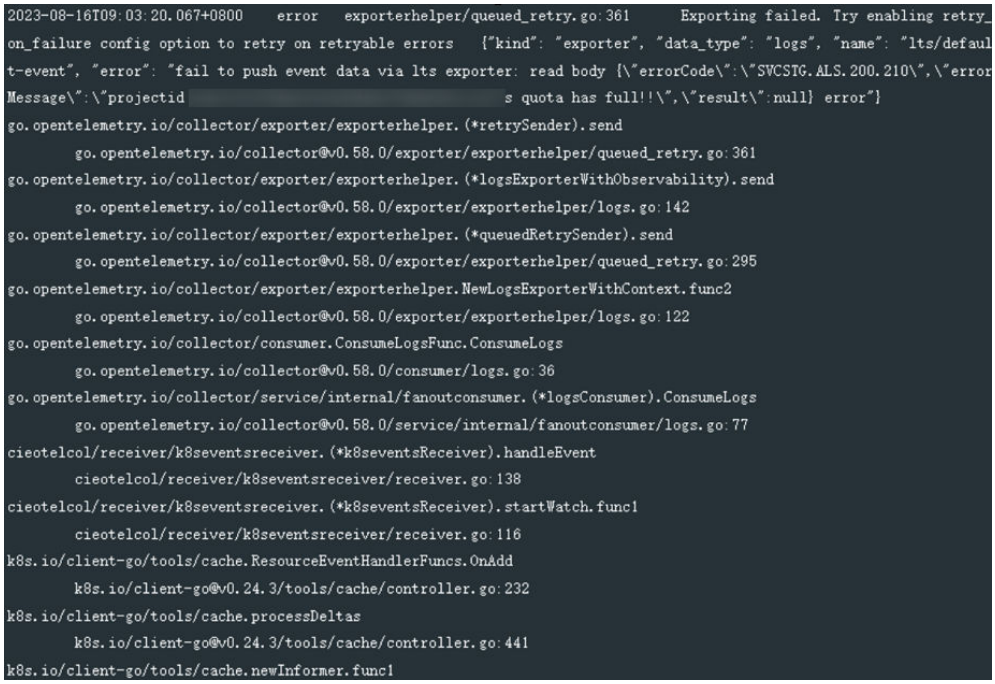
- **"Failed to create log group, the number of log groups exceeds the quota" is reported in the standard output log of log-operator.**

  Example:

  ```
  2023/05/05 12:17:20.799 [E] call 3 times failed, resion: 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**.

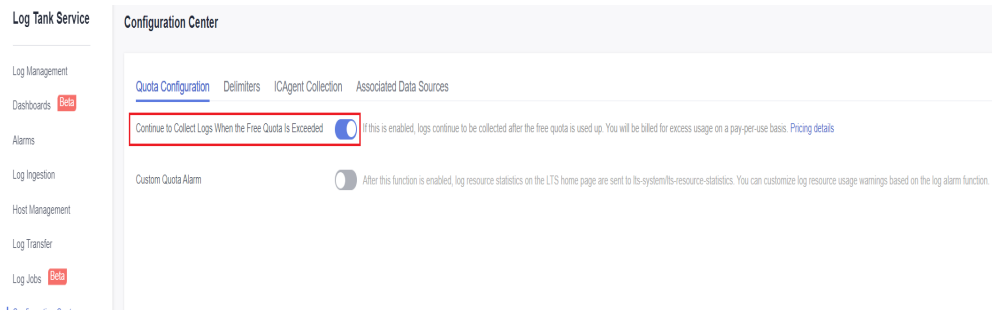
- **Logs cannot be reported, and "log's quota has full" is reported in the standard output log of the OTel component.**

  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 on the left, and enable **Continue to Collect Logs When the Free Quota Is Exceeded**.

Figure 4-56 Quota configuration



- **Text logs cannot be collected because wildcards are configured for 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**.

4.7.2.2 Collecting Kubernetes Events

The Cloud Native Logging add-on works with LTS to collect and store Kubernetes events and works with AOM to generate alarms.

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**.

Reporting Kubernetes Events to LTS

The Cloud Native Logging add-on has not been installed in a cluster.

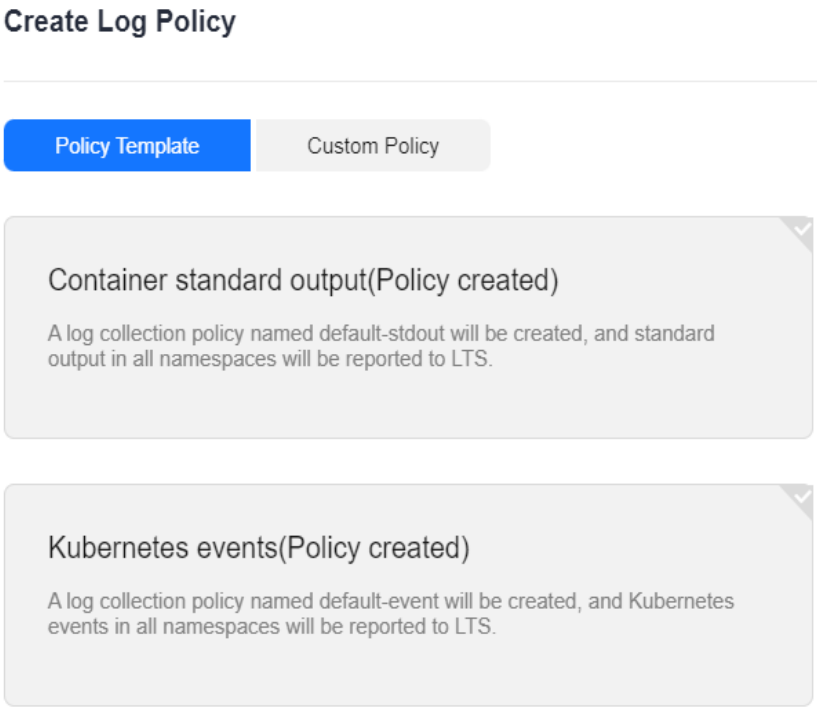
You can select **Kubernetes events** when installing the Cloud Native Logging add-on. A default log collection policy will be created, and all events collected will be reported to LTS. For details about how to install the add-on, see **4.7.2.1 Collecting Logs**.

The Cloud Native Logging add-on has been installed in a cluster.

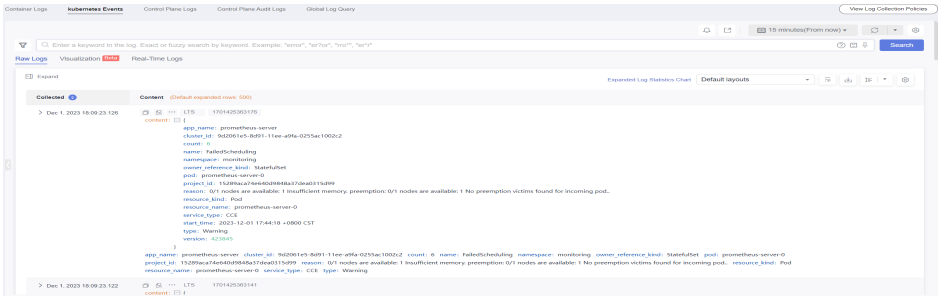
1. Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.
3. Click **Create Log Policy** and configure parameters as required.

Policy Template: If **Kubernetes events** is not selected during add-on installation or the log collection policy is deleted, you can use this option to create a default log collection policy.



4. On the **Logging** page, select the log stream configured in the log collection policy to view the events reported to LTS.



Reporting Kubernetes Events to AOM

After the Cloud Native Logging add-on is installed, all Warning events and some Normal events will be reported to AOM by default. The reported events can be used to configure alarms.

Custom Event Reporting

If the reported events cannot meet requirements, you can modify the settings for the events.

Step 1 Run the following command on the cluster to edit 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:    # Name of the Normal event to be collected. If this parameter is not specified, all
Normal events will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Name of the Normal event that is not collected. If this parameter is not specified,
all Normal events will be collected.
        - ScaleDown
      warningEvents:    # Used to configure Warning events
        enable: true    # Whether to enable Warning event collection
        includeNames:    # Name of the Warning event to be collected. If this parameter is not specified, all
Warning events will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Name of the Warning event that is not collected. If this parameter is not specified,
all Warning 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

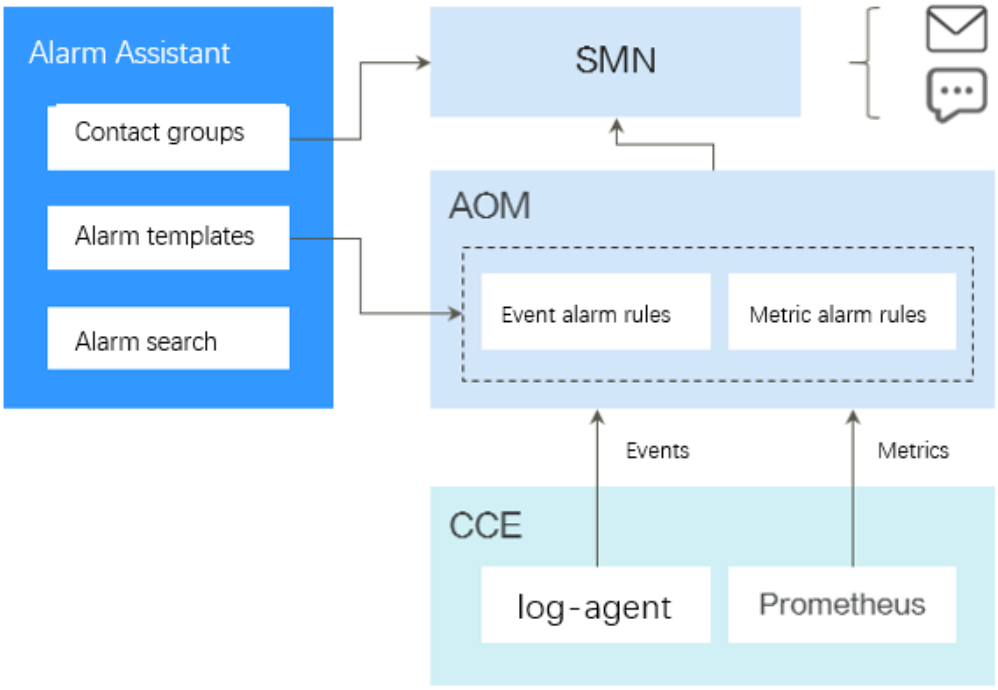
4.7.3 Alarm Assistant

4.7.3.1 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 Assistant that allows you to quickly configure and view common alarms (such as resource usage alarms).

Figure 4-57 Alarm Assistant architecture



- Alarm Assistant

  Based on the alarm capabilities of AOM, Alarm Assistant provides quick alarm search and configuration for clusters. You can use Alarm Assistant 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.

4.7.3.2 Configuring Alarms in Alarm Assistant

By using AOM, Alarm Assistant can promptly detect cluster faults and generate alarms for service stability. Alarm Assistant 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

Only Huawei Cloud accounts, HUAWEI IDs, or IAM users with CCE administrator or FullAccess permissions can perform all operations using Alarm Assistant. IAM users with the CCE ReadOnlyAccess permission can only view all resources.

Enabling Alarm Assistant

Step 1 Click the name of the target cluster and choose **Alarm Assistant** in the navigation pane.

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 **Configuring Alarm Notification Recipients**.

Step 3 Click **OK**.

NOTE

Metric alarm rules can be created in Alarm Assistant 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 **4.7.1.1 Enabling Cluster Monitoring**.

Event alarms in **Table 4-49** can be reported only when Kubernetes event collection is enabled in Logging. For details, see **4.7.2.2 Collecting Kubernetes Events**.

----End

Configuring Alarm Rules

After Alarm Assistant is enabled for 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 name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and click the **Alarm Rules** tab. Then configure and manage alarm rules.

By default, Alarm Assistant 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 4-49** lists default alarm rules.

----End

Table 4-49 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
	UpdateLoadBalancerFailed	Check whether a load balancer is updated.	Event	Cloud Native Logging	N/A
	Pod OOM	Check whether an OOM occurs in the pod.	Event	CCE Node Problem Detector Cloud Native Logging	PodOOMKilling

Rule Type	Alarm Item	Description	Alarm Type	Dependency Item	PromQL/Event Name
Cluster status rule set	Unavailable cluster	Check whether the cluster is available.	Event	Cloud Native Logging	N/A

Configuring Alarm Notification Recipients

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 configure contact groups to manage endpoints that have subscribed to alarm messages. After creating a contact group, associate alarm rule set with the group. When an alarm is triggered, the subscription endpoints in the contact group can receive the alarm messages.

Step 1 Log in to the CCE console.

Step 2 On the cluster list page, click the name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and click the **Contact Group** tab.

Step 4 Click **Create Contact Group** and configure parameters.

- **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 you do not specify **Alarm message display name**, the sender will be **username@example.com**. The display name of an alarm message can be changed after the 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. If the endpoint receives the request, confirm the request as prompted. After the confirmation is complete, the subscription endpoint changes to **Confirmed**.

Step 7 Click  to enable the contact group so that the contact group is bound to the alarm rule set.

NOTE

An alarm rule set can be bound to a maximum of five contact groups.

----End

Viewing Alarms

You can view the latest historical alarms on the **Alarm list** tab.

Step 1 Log in to the CCE console.

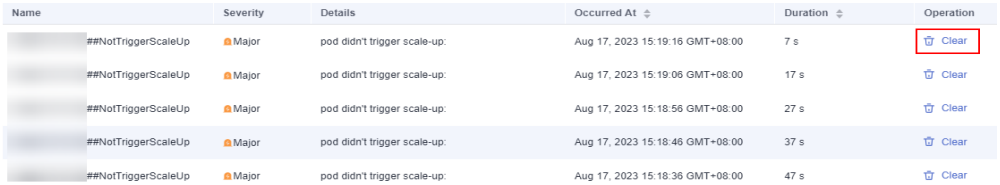
Step 2 On the cluster list page, click the name of the target cluster to go to the details page.

Step 3 Choose **Alarm Assistant** in the navigation pane and 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 you confirm that an alarm has been handled, click **Clear** in the **Operation** column. After the alarm is cleared, you can view it in the historical alarm list.

Figure 4-58 Querying alarms



----End

4.7.3.3 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 **4.7.1.1 Enabling Cluster Monitoring**.

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

Step 2 Choose **Alarm Assistant** in the navigation pane, click **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
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 a contact group. For details about the parameters, see **Configuring Alarm Notification Recipients**.	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 *$event.metadata.* A maximum of 10 alarm tags can be added.
  - **Alarm Annotation**: An attribute that is not used for alarm identification. In the message template, the annotation value is referenced as *$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 **4.7.2.1 Collecting Logs**.

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

Step 2 Choose **Alarm Assistant** in the navigation pane, click **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.	ScalingReplicaSet

Parameter	Description	Example Value
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 a contact group. For details about the parameters, see **Configuring Alarm Notification Recipients**.	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

4.8 Namespaces

4.8.1 Creating a Namespace

When to Use Namespaces

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.

Constraints

A maximum of 6,000 Services can be created in each namespace. The Services mentioned here indicate the Kubernetes Service resources added for workloads.

Namespace Types

Namespaces can be created in either of the following ways:

- 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), such as public add-ons and container charts.
  - **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.

Creating a 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 on the left, choose **Namespaces** and then click **Create Namespace** in the upper right corner.

Step 3 Configure the parameters based on **Table 4-50**.

Table 4-50 Parameters for creating a namespace

Parameter	Description
Name	Unique name of the created namespace.
Description	Description about the namespace.

Parameter	Description
Quota Management	Resource quotas can limit the number of resources available in namespaces, for resource allocation by namespace. **NOTICE** **You are advised to set resource quotas in the namespace as required to prevent cluster or node exceptions caused by resource overload.** Enter an integer. If the quota of a resource is not specified, no limit is posed 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

Using kubectl to Create a Namespace

Define a namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

Run the **kubectl** command to create it.

```
$ kubectl create -f custom-namespace.yaml
namespace/custom-namespace created
```

You can also run the **kubectl create namespace** command to create a namespace.

```
$ kubectl create namespace custom-namespace
namespace/custom-namespace created
```

4.8.2 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.

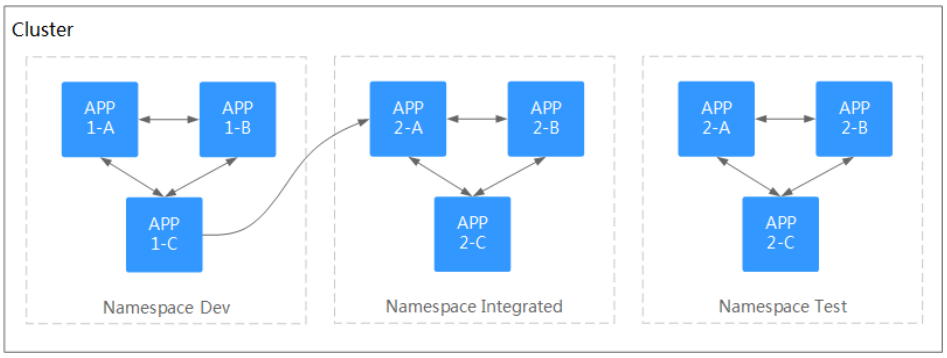
  Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

- 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.

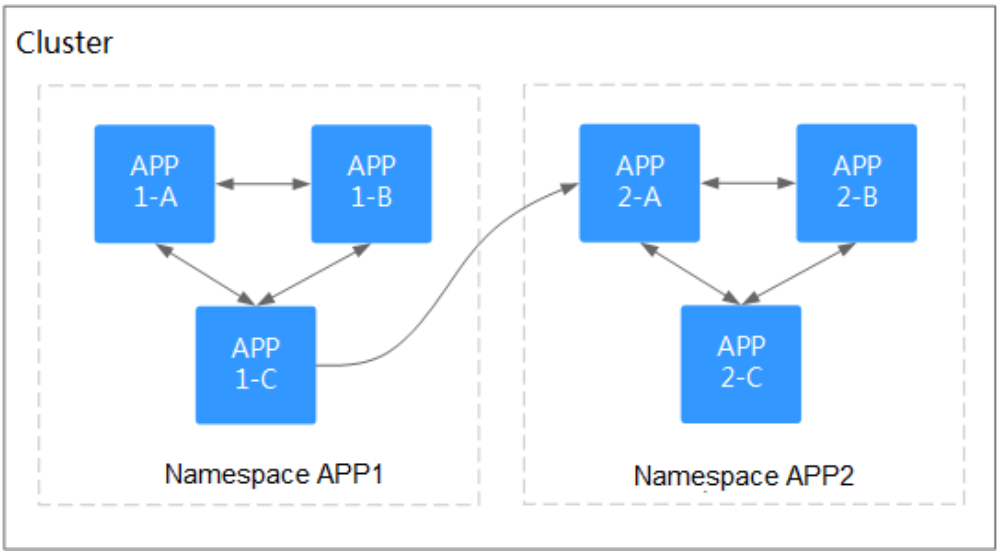
Figure 4-59 One namespace for one environment



- **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 4-60 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 on the left, choose **Namespaces**.

Step 2 Locate the row containing the target namespace and choose **More** > **Manage Label** in the **Operation** column.

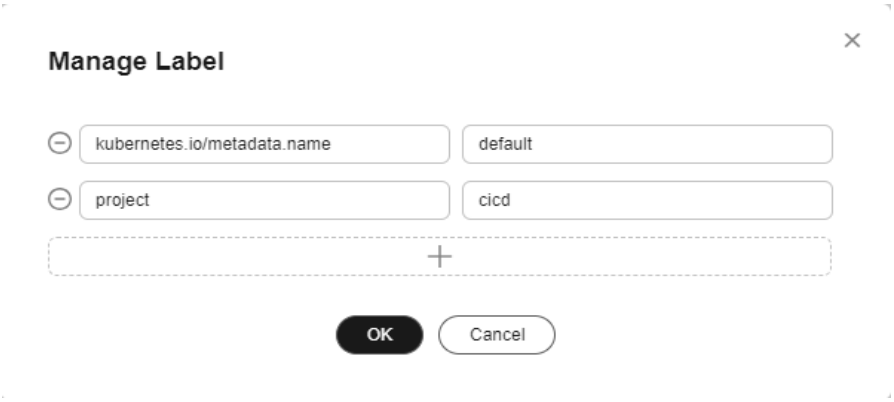
Step 3 In the dialog box that is displayed, the existing labels of the namespace are displayed. Modify the labels as needed.

- 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 ⊖ next the label to be deleted and then **OK**.

Figure 4-61 Adding or deleting a namespace label



Step 4 Switch to the **Manage Label** dialog box again and check the modified 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

4.8.3 Setting Resource Quotas

Namespace-level resource quotas limit the amount of resources available to teams or users when these teams or users use the same cluster. The quotas include the

total number of a type of objects and the total amount of compute resources (CPU and memory) consumed by the objects.

Usage

By default, running pods can use the CPUs and memory of a node without restrictions. This means the pods in a namespace may exhaust all resources of the cluster.

Kubernetes provides namespaces for you to group workloads in a cluster. By setting resource quotas for each namespace, you can prevent resource exhaustion and ensure cluster reliability.

You can configure quotas for resources such as CPU, memory, and the number of pods in a namespace. For more information, see **Resource Quotas**.

Constraints

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. The resource quota limits the total resource consumption of each namespace and records the resource information in the cluster. Therefore, after the **enable-resource-quota** option is enabled, the probability of resource creation conflicts increases in large-scale concurrency scenarios, affecting the performance of batch resource creation.

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 on the left, choose **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 Set the resource quotas and click **OK**.

NOTICE

- After setting 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.

----End

4.9 ConfigMaps and Secrets

4.9.1 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.

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 4-51 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 connect to the cluster. For details, see **4.3.4.1 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 4-52 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 4-53**.

Table 4-53 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 4-51**. 3. Click **OK**.
Deleting a ConfigMap	Select the configuration you want to delete and click **Delete**. Follow the prompts to delete the ConfigMap.

4.9.2 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

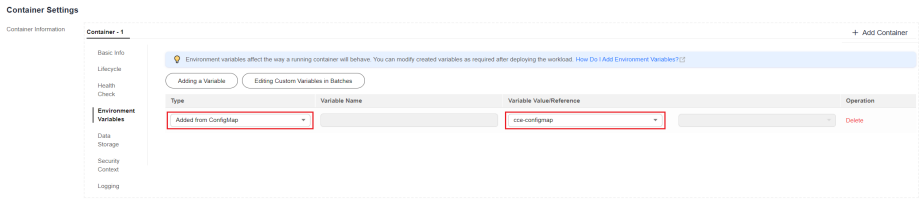
Using the CCE console

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**. In the dialog box displayed, click **Create Workload** in the upper right corner.

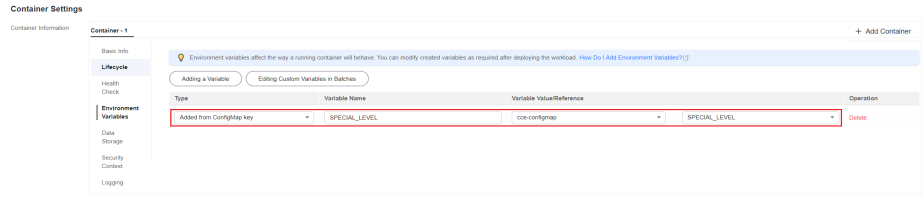
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **Added from 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 connect to the cluster. For details, see **3.2.3.1 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).

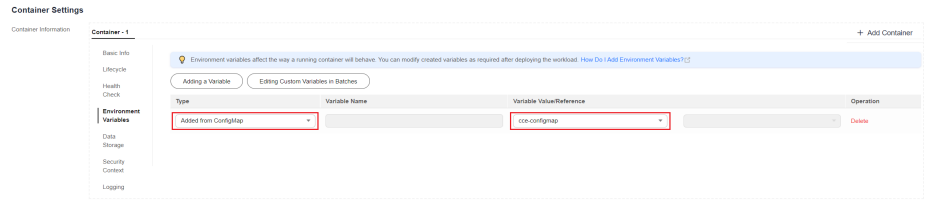
Using the CCE console

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**.

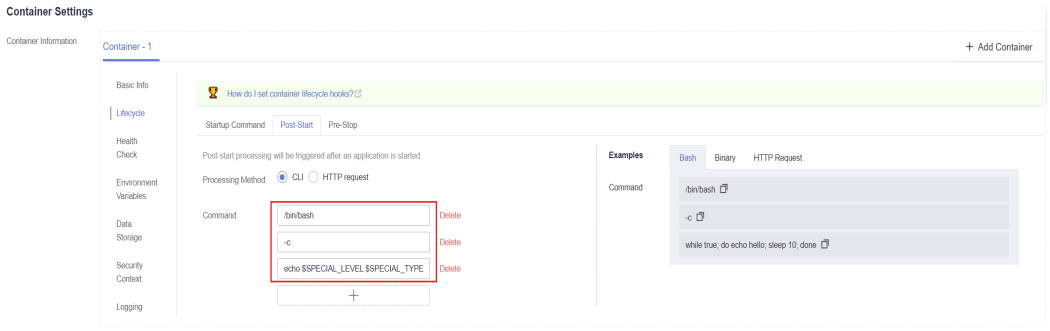
- **Added from ConfigMap**: Select a ConfigMap to import all of its keys as environment variables.



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, that is, the 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 connect to the cluster. For details, see **3.2.3.1 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, the **cce-configmap** ConfigMap is imported to the workload. *SPECIAL_LEVEL* and *SPECIAL_TYPE* are the environment variable names in the workload, that is, the 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 After the workload runs properly, the following content is entered into 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 **4.9.1 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.

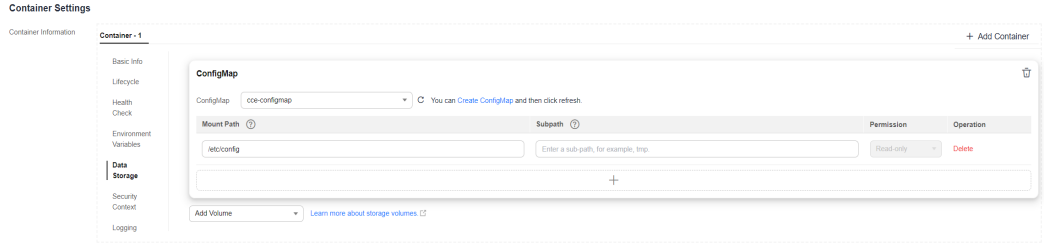
When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **ConfigMap** from the drop-down list.

Step 3 Select parameters for mounting a ConfigMap volume, as shown in **Table 4-54**.

Table 4-54 Mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created beforehand. For details, see **4.9.1 Creating a ConfigMap**.
Mount Path	Enter a mount point. 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, which leads to a container startup failure or workload creation failure. **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 is used by default. • The subpath can be the key and value of a ConfigMap or 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 ConfigMap/secret updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 4-62 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 connect to the cluster. For details, see **3.2.3.1 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

4.9.3 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.

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 4-55 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 connect to the cluster. For details, see **4.3.4.1 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 4-56**.

NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

Table 4-56 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 4-55**. 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
******
```

4.9.4 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 **4.9.5 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

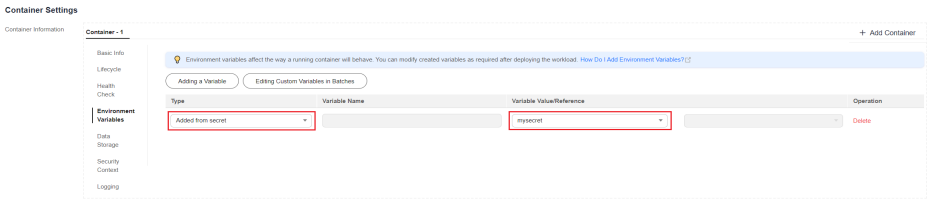
Using the CCE console

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.

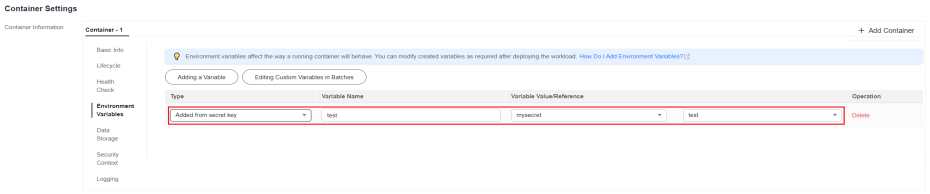
When creating a workload, click **Environment Variables** in the **Container Settings** area, and click **Add Variable**.

- **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 connect to the cluster. For details, see **3.2.3.1 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 **4.9.3 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.

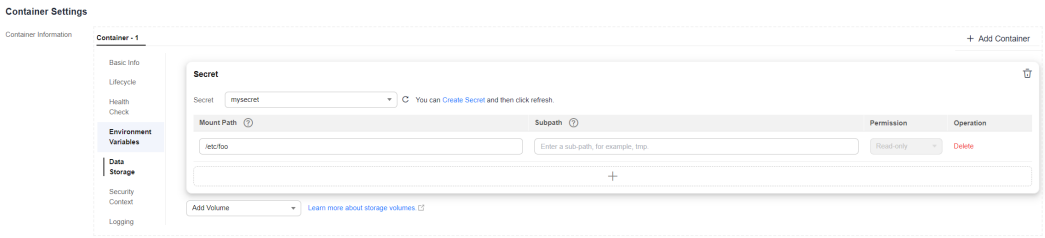
Step 3 Select parameters for mounting a secret volume, as shown in **Table 4-57**.

Table 4-57 Mounting a secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created beforehand. For details, see **4.9.3 Creating a Secret**.
Mount Path	Enter a mount point. 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 cause 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, which leads to a container startup failure or workload creation failure. **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 is used by default. • The subpath can be the key and value of a ConfigMap or 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 ConfigMap/secret updates.
Permission	Read-only, indicating that data volume in the path is read-only.

Figure 4-63 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 connect to the cluster. For details, see **3.2.3.1 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 are 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

4.9.5 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 of 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 data of **paas.elb** is the temporary AK/SK data, which is used to create ELB load balancers during Service and ingress creation. The data of paas.elb is periodically updated and expires after a certain period of time.

In practice, you will not directly use paas.elb. However, do not delete it. Otherwise, ELB load balancers will fail to be created.

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.

```
$ 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>
```

4.10 Auto Scaling

4.10.1 Scaling a Workload

4.10.1.1 Workload Scaling Mechanisms

How HPA Works

Horizontal Pod Autoscaler (HPA) is a controller that controls horizontal pod scaling. HPA periodically checks the pod metrics, calculates the number of replicas 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 the numbers of cluster nodes and pods, and CPU and memory

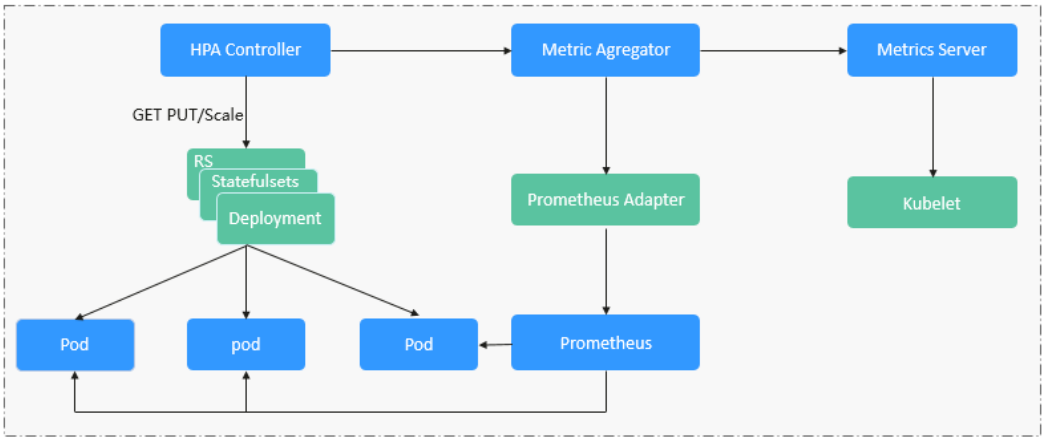
usages of containers. Kubernetes does not provide such monitoring capabilities itself. You can use extensions to monitor and collect your data. CCE integrates **Metrics Server** to realize such capabilities:

- **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 works with Metrics Server for workload scaling based on CPU or memory.

Figure 4-64 shows how HPA works.

Figure 4-64 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:

  desiredReplicas = ceil[currentReplicas x (currentMetricValue/ desiredMetricValue)]

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 replicas 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.

  Use the formula: ratio = currentMetricValue/desiredMetricValue

  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.

4.10.1.2 HPA Policies

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

The following add-on has been installed in the cluster:

- **4.11.2 Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.

Creating an HPA 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 on the left, choose **Policies**. On the **Scaling Policies**, click the **HPA Policies** tab and then **Create HPA Policy** in the upper right corner.

Step 3 Configure basic information.

- **Policy Name**: Enter a name for the policy.
- **Namespace**: Select the namespace that the workload belongs to.
- **Associated Workload**: Select the workload that the policy is configured for.

Step 4 Configure other parameters.

Table 4-58 HPA policy parameters

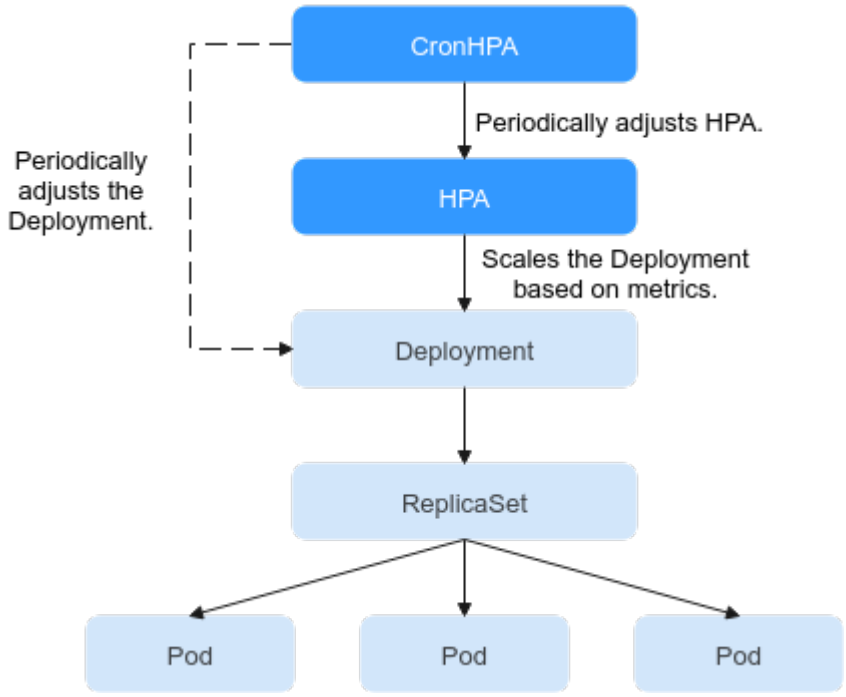
Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Scaling Behavior	• **Default**: Scale workloads using the Kubernetes default behavior. For details, see **Default Behavior**. • **Custom**: Scale workloads 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 = CPUs or memory used by pods/Requested CPUs or memory. • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. 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. 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 v1.15 or later.**

Step 5 Click **Create**.

----End

4.10.1.3 CronHPA Policies

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.

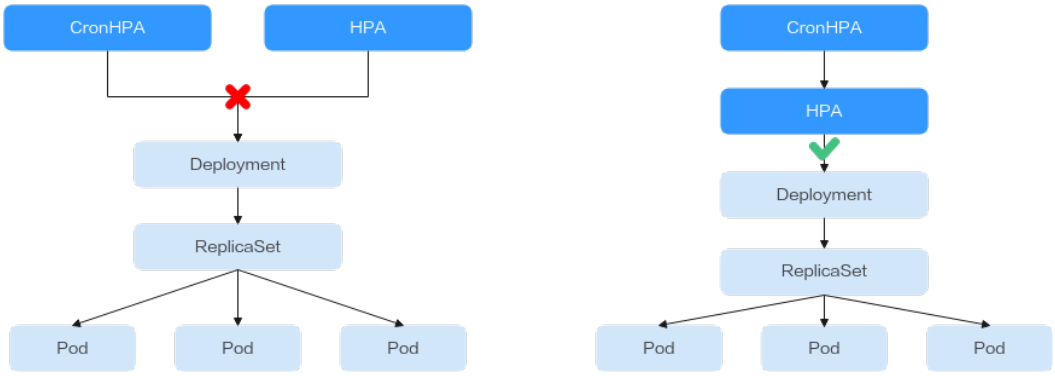
Prerequisites

The **4.11.6 CCE Advanced HPA** add-on has been installed in the cluster.

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 4-65 CronHPA scaling scenarios

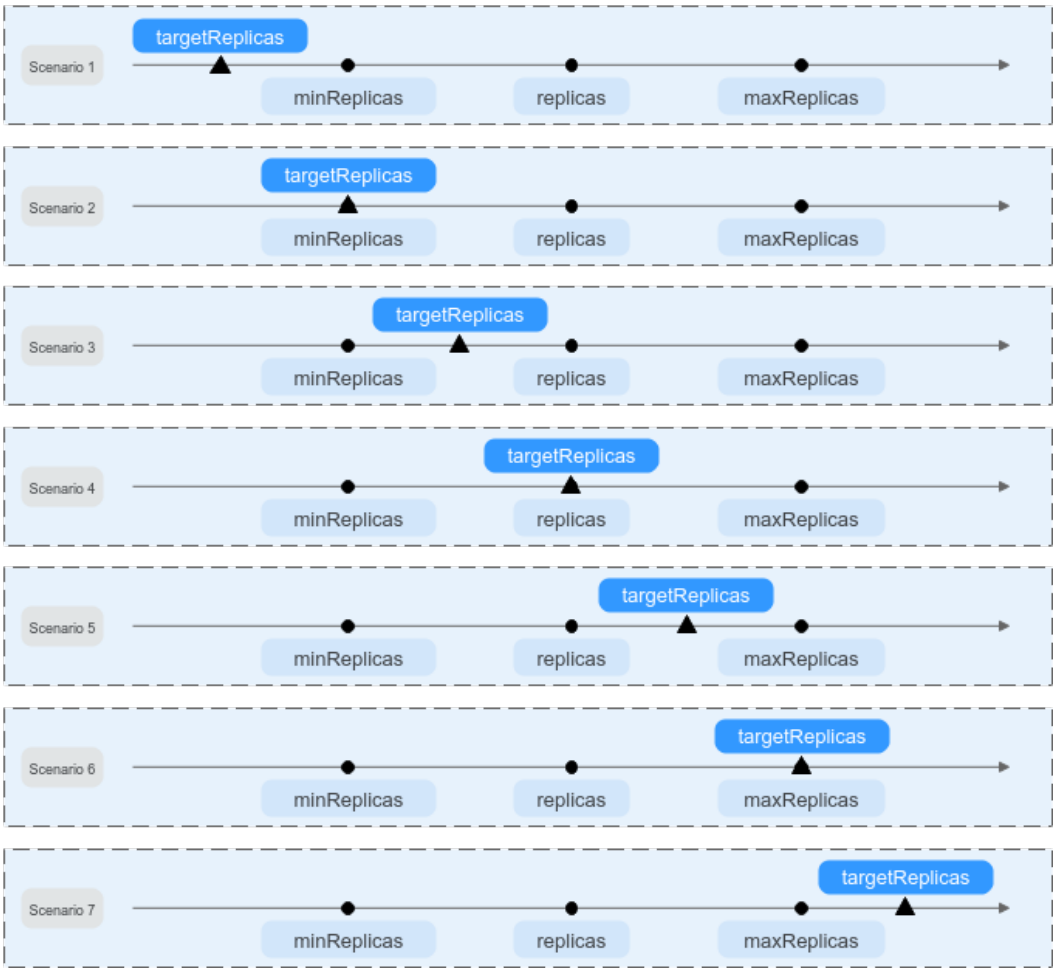


Figure 4-65 shows possible scaling scenarios. The following examples detail how CronHPA modifies the number of pods in HPAs.

Table 4-59 CronHPA scaling parameters

Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
1	**targetReplicas** < minReplicas ≤ replicas ≤ maxReplicas	4	5	5/10	HPA: 4/10 Deployments: 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 Deployments: 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.
3	minReplicas < **targetReplicas** < replicas ≤ maxReplicas	4	5	1/10	HPA: 4/10 Deployments: 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.

Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
4	minReplicas < **targetReplicas** = replicas < maxReplicas	5	5	1/10	HPA: 5/10 Deployments: 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.
5	minReplicas ≤ replicas < **targetReplicas** < maxReplicas	6	5	1/10	HPA: 6/10 Deployments: 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 Deployments: 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**.

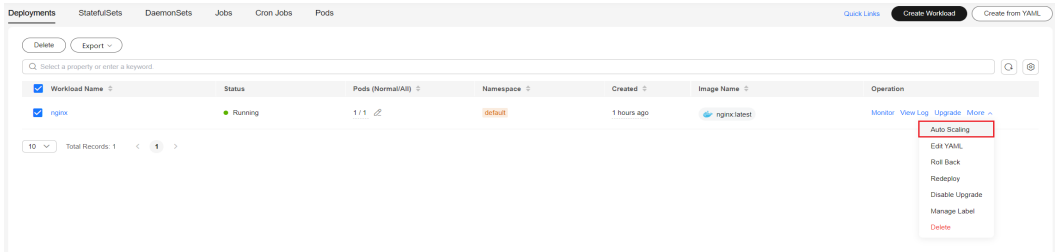
Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas / maxReplicas)	Result	Operation Description
7	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Deployments: 11	When the value of **targetReplicas** is greater than that of **maxReplicas**: • 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 4-66 Scaling a workload



Step 3 Set **Policy Type** to **HPA+CronHPA** to 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 **4.10.1.2 HPA Policies**.

Figure 4-67 Enabling the HPA policy

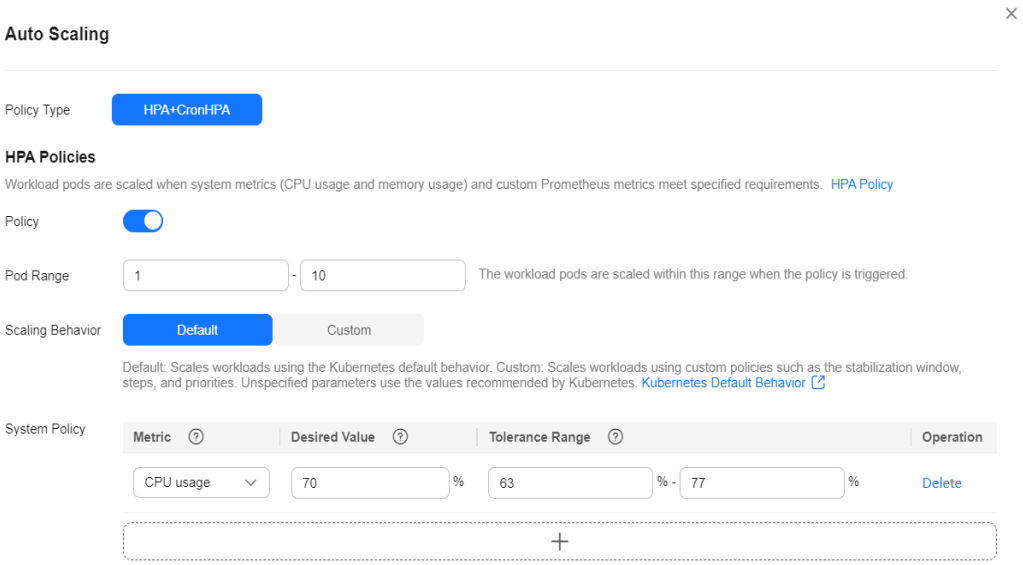


Table 4-60 HPA policy parameters

Parameter	Description
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the pods are scaled within this range.
Scaling Behavior	• **Default**: Scale workloads using the Kubernetes default behavior. For details, see **Default Behavior**. • **Custom**: Scale 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.

Parameter	Description
System Policy	• **Metric**: You can select **CPU usage** or **Memory usage**. **NOTE** Usage = CPUs or memory used by pods/Requested CPUs or memory • **Desired Value**: Enter the desired average resource usage. This parameter indicates the desired value of the selected metric. 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. 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 only supported in clusters v1.15 or later.**

Step 5 Click + in the CronHPA policy rule. In the dialog box displayed, configure scaling policy parameters.

Figure 4-68 Enabling the CronHPA policy

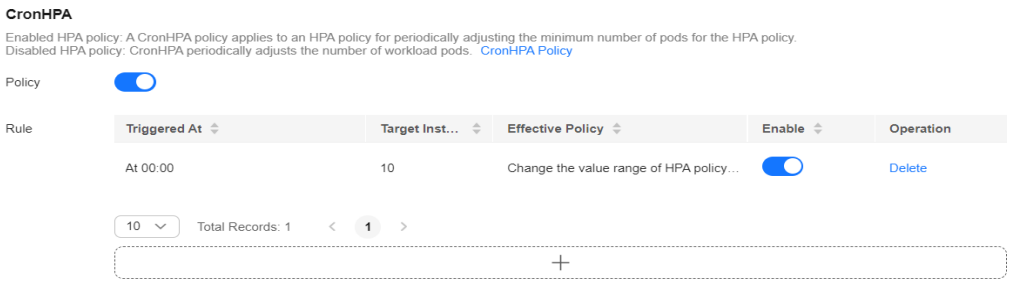


Table 4-61 CronHPA policy parameters

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 4-59**.
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 the kubectl command

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 4-62 Key fields of CronHPA

Field	Description
apiVersion	API version. The value is fixed at **autoscaling.cce.io/v2alpha1**.
kind	API type. The value is fixed at **CronHorizontalPodAutoscaler**.

Field	Description
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

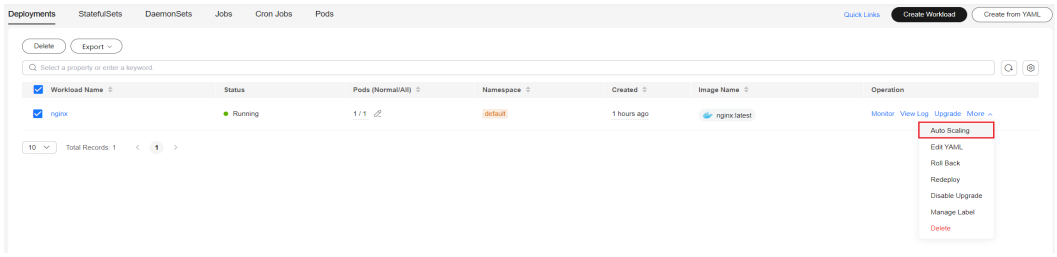
CronHPA adjusts associated Deployments separately to periodically adjust the number of Deployment pods. The method is as follows:

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 4-69 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 4-70 Using CronHPA to adjust the number of workload pods

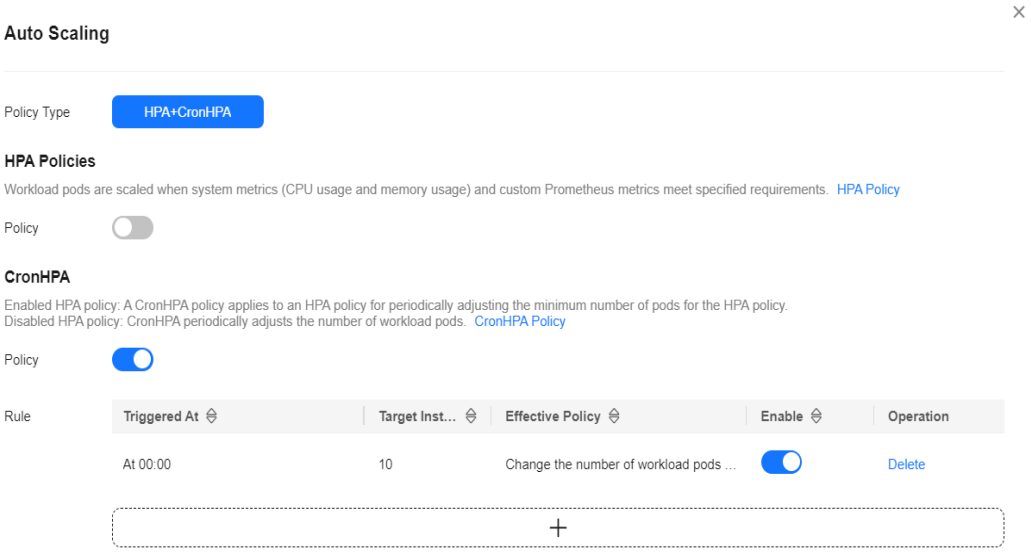


Table 4-63 CronHPA policy parameters

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 commands

```
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
```

4.10.1.4 Managing Workload Scaling Policies

Scenario

After an HPA policy is created, you can view, edit (both on the console and using the YAML file), or delete the policy.

Viewing an HPA Policy

You can view the rules, status, and events of an HPA 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 on the left, choose **Policies**. On the **HPA Policies** tab, locate the HPA policy you want to view and click ⌄ next to the policy.

Step 3 In the expanded area, choose **View Events** in the **Operation** column. If the policy malfunctions, locate and rectify the fault based on the error message displayed on the page.

NOTE

You can also view an HPA 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 on the left, choose **Workloads**. Click the workload name to view its details.
3. On the workload details page, switch to the **Policies** tab to view the HPA policy. You can also view the policies you configured in **Workload Scaling**.

Table 4-64 Event types and names

Event Type	Event Name	Description
Normal	SuccessfulRescale	The scaling is performed successfully.
Abnormal	InvalidTargetRange	Invalid target range.
	InvalidSelector	Invalid selector.
	FailedGetObjectMetric	Objects fail to be obtained.
	FailedGetPodsMetric	Pods fail to be obtained.
	FailedGetResourceMetric	Resources fail to be obtained.
	FailedGetExternalMetric	External metrics fail to be obtained.
	InvalidMetricSourceType	Invalid metric source type.
	FailedConvertHPA	HPA conversion failed.
	FailedGetScale	The scale fails to be obtained.
	FailedComputeMetricsReplicas	Failed to calculate metric-defined replicas.
	FailedGetScaleWindow	Failed to obtain ScaleWindow.
	FailedRescale	Failed to scale the workload.

----End

Editing HPA Policy

Edit an existing HPA policy when needed.

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

Step 2 In the navigation pane on the left, choose **Policies**. On the **HPA Policies** tab, locate the HPA policy you want to edit and choose **More** > **Edit** in the **Operation** column.

Step 3 On the **Edit HPA Policy** page, modify the parameter settings based on **Table 4-58**.

Step 4 Click **OK**.

----End

Editing an HPA Policy Using YAML

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

Step 2 In the navigation pane on the left, choose **Policies**. On the **HPA Policies** tab, locate the HPA policy you want to edit and click **Edit YAML** in the **Operation** column.

Step 3 In the **Edit YAML** dialog box displayed, edit or download the YAML file.

----End

Deleting an HPA 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 on the left, choose **Policies**. On the **HPA Policies** tab, locate the HPA policy you want to edit and choose **More** > **Delete** in the **Operation** column.

Step 3 In the dialog box displayed, click **Yes**.

----End

4.11 Add-ons

4.11.1 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 more information about CoreDNS, see **DNS**.

Constraints

To run CoreDNS 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.

Editing the Add-on

This add-on is installed by default. To modify its settings, perform 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 on the left, choose **Add-ons**. Locate **CoreDNS** on the right and click **Edit**.

Step 2 On the **Edit Add-on** page, modify the specifications.

Table 4-65 CoreDNS specifications

Parameter	Description
Pods	Number of pods for the add-on. High availability is not possible if there is only one pod. If an error occurs on the node where the pod runs, the add-on will fail.
Containers	CPU and memory quotas of the containers for running the add-on. Queries per second (QPS) of the 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.

Step 3 Modify the parameters.

Table 4-66 CoreDNS 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

Advance Config	• **parameterSyncStrategy**: indicates whether to configure consistency check when the add-on is upgraded.
	– **ensureConsistent**: indicates that the configuration consistency check is enabled. If the configuration recorded in the cluster is inconsistent with the actual configuration, the add-on cannot be upgraded.
	– **force**: indicates that the configuration consistency check is ignored during an upgrade. In this case, you must ensure that the current effective configuration is the same as the original configuration. After the add-on is upgraded, restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	– **inherit**: indicates that custom settings are automatically inherited during an upgrade. After the add-on is upgraded, restore the value of **parameterSyncStrategy** to **ensureConsistent** to enable the configuration consistency check again.
	• **stub_domains**: indicates the subdomains, which allow you to configure a domain name server for a custom domain name. A subdomain is in the format of a key-value pair, where the key is the suffix of a DNS domain name and the value is one or more DNS IP addresses.
	• **upstream_nameservers**: indicates the IP address of the upstream DNS server.
	• **servers**: indicates the name servers, 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 4-67 describes common plugins. For details, see **Plugins**.
	Example:
	`{`
	`"servers": [`
	`{`
	`"plugins": [`
	`{`
	`"name": "bind",`
	`"parameters": "{$POD_IP}"`
	`},`
	`{`
	`"name": "cache",`
	`"parameters": 30`

Parameter	Description

	<pre>{
	            },
	            {
	               "name": "errors"
	            },
	            {
	               "name": "health",
	               "parameters": "{$POD_IP}:8080"
	            },
	                     {
	               "name": "ready",
	               "{$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": "."
	            }
	         ]
	      }
	   ],
	   "stub_domains": {
	      "acme.local": [
	         "1.2.3.4",
	         "6.7.8.9"
	      ]
	   },
	   "upstream_nameservers": ["8.8.8.8", "8.8.4.4"]
	}</pre>

Table 4-67 Default plugin configuration of the active CoreDNS zone

Plugin Name	Description
bind	Host IP address listened by CoreDNS. Retain the default value **{$POD_IP}**. For details, see **bind**.
cache	Enables DNS cache. For details, see **cache**.

Plugin Name	Description
errors	Errors are logged to stdout. For details, see **errors**.
health	Health check for CoreDNS. {$POD_IP}:8080 is listened to. Retain the default setting. Otherwise, the CoreDNS health check will fail and the add-on will restart repeatedly. For details, see **health**.
ready	Whether the backend server is ready to receive traffic. {$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	CoreDNS Kubernetes plugin, which provides the service parsing capability in a cluster. For details, see **kubernetes**.
loadbalance	Round-robin DNS load balancer that randomizes the order of A, AAAA, and MX records in an answer. For details, see **loadbalance**.
prometheus	API for obtaining CoreDNS metrics. {$POD_IP}:9153 is listened to in the default zone. Retain the default setting. Otherwise, Prometheus cannot collect CoreDNS metrics. For details, see **Prometheus**.
forward	Forwards any queries that are not within the cluster domain of Kubernetes to predefined resolvers (**/etc/resolv.conf**). For details, see **forward**.
reload	Automatically reloads modified Corefiles. After you modify a ConfigMap, wait for two minutes for the modification to take effect. For details, see **reload**.

log	Enables CoreDNS logging. For details, see **log**.
	Example:
	`{`
	`  "name": "log"`
	`}`

template	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 Click **Install**.

----End

Components

Table 4-68 CoreDNS 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 should be provided using the **dnsPolicy** field in **dnsConfigPod**.

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

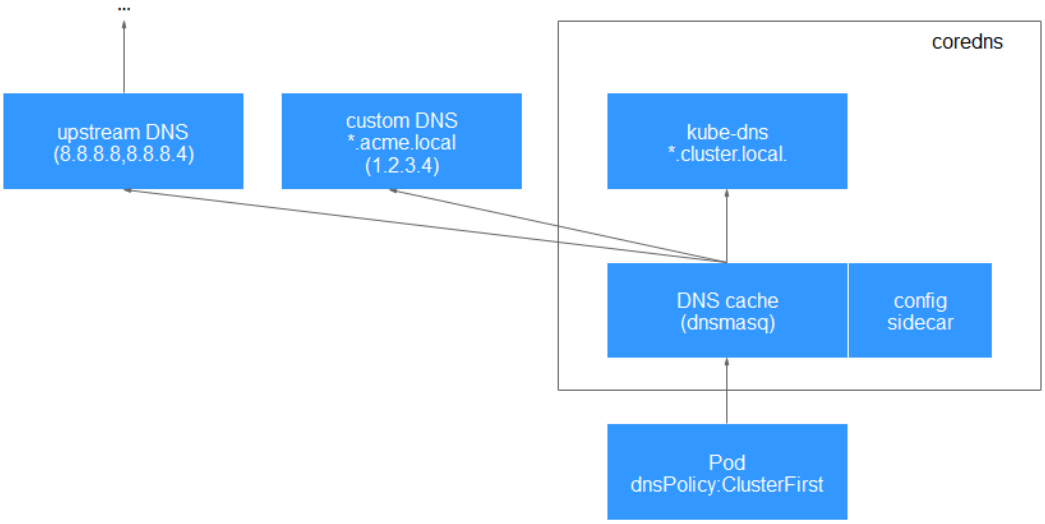
Without stub domain configurations: Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

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 4-71 Routing



4.11.2 Kubernetes Metrics Server

Kubernetes provides resource usage metrics, such as container CPU usage and memory usage, through the Metrics API. These metrics can be directly accessed by users (by running the **kubectl top** command) or used by a controller in the cluster (for example, HPA) to make decisions.

Kubernetes Metrics Server is an aggregator for monitoring data of core cluster resources. You can quickly install this add-on on the CCE console.

After this add-on is installed, you can create HPA policies. For details, see **4.10.1.2 HPA Policies**.

The official community project and documentation are available at **https://github.com/kubernetes-sigs/metrics-server**.

Editing 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 on the left, choose **Add-ons**. Locate **Kubernetes Metrics Server** on the right and click **Edit**.

Step 2 On the **Edit Add-on** page, modify the specifications.

Table 4-69 Kubernetes Metrics Server specifications

Parameter	Description
Pods	Number of pods for the add-on.
Containers	CPU and memory quotas of the containers for running the add-on.

Step 3 Click **Install**.

----End

Components

Table 4-70 metrics-server 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

4.11.3 Cloud Native Cluster Monitoring

Introduction

The Cluster Native Cluster Monitoring add-on (kube-prometheus-stack) uses Prometheus-operator and Prometheus to provide easy-to-use, end-to-end Kubernetes cluster monitoring.

This add-on allows the monitoring data to be interconnected with Container Intelligent Analysis (CIA) so that you can view monitoring data and configure alarms on the CIA console.

Open source community: **https://github.com/prometheus/prometheus**

Permissions

The node-exporter component of the Cluster 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

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Add-ons**. Locate **Cloud Native Cluster Monitoring** on the right and click **Install**.

Step 2 On the **Install Add-on** page, select a version.

Step 3 Configure the parameters.

Step 4 Click **Install**.

To use this add-on to provide system resource metrics (such as CPU and memory usages) for workload auto scaling, enable the Metric API. For details, see

Providing Resource Metrics Through the Metrics API. After the configuration, you can use Prometheus to collect system resource metrics without repeatedly installing the Kubernetes Metrics Server add-on.

----End

Components

All Kubernetes resources created during kube-prometheus-stack add-on installation are created in the namespace named **monitoring**.

Table 4-71 kube-prometheus-stack components

Component	Description	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.	Deployment
prometheus (workload name: prometheus-lightweight)	A Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.	StatefulSet
kubeStateMetrics (workload name: kube-state-metrics)	Converts the Prometheus metric data into a format that can be identified by Kubernetes APIs. **NOTE** If the components run in multiple pods, only one pod provides metrics.	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	Pod

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 the following information 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
```

4.11.4 Cloud Native Logging

Introduction

The Cloud Native Logging add-on (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, 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. For details about how to collect logs, see **4.7.2.1 Collecting Logs**.

Constraints

The constraints on using the log-agent add-on are as follows:

- A maximum of 50 log collection rules can be configured for each cluster.
- log-agent cannot collect .gz, .tar, or .zip log files.
- If the container runtime is containerd, stdout logs cannot be multi-line.
- In each cluster, no more than 10,000 single-line logs can be collected per second, and no more than 2,000 multiple-line logs can be collected per second.

Permissions

The fluent-bit component reads and collects the standard output logs and container file 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 on the left, choose **Add-ons**. Locate **Cloud Native Logging** on the right and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 4-72 Cloud Native Logging specifications

Parameter	Description
Instances	Number of pods that will be created to match the selected add-on specifications. If you select **Custom**, you can adjust the number of pods as required.
Containers	The add-on contains the following containers, whose specifications can be adjusted as required: • **fluent-bit**: log collector, which is installed on each node as a DaemonSet. • **log-operator**: parses and updates log rules. • **otel-collector**: forwards logs collected by **fluent-bit** to LTS.

Step 3 Click **Install**.

----End

Components

Table 4-73 log-agent components

Component	Description	Resource Type
fluent-bit	A lightweight log collector and forwarder for collecting logs.	Pod
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

4.11.5 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 uses community templates and images. This add-on generates Nginx configuration and stores the configuration using ConfigMaps. 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 NGINX Ingress Controller Works**.

You can visit the **open source community** for more information.

NOTE

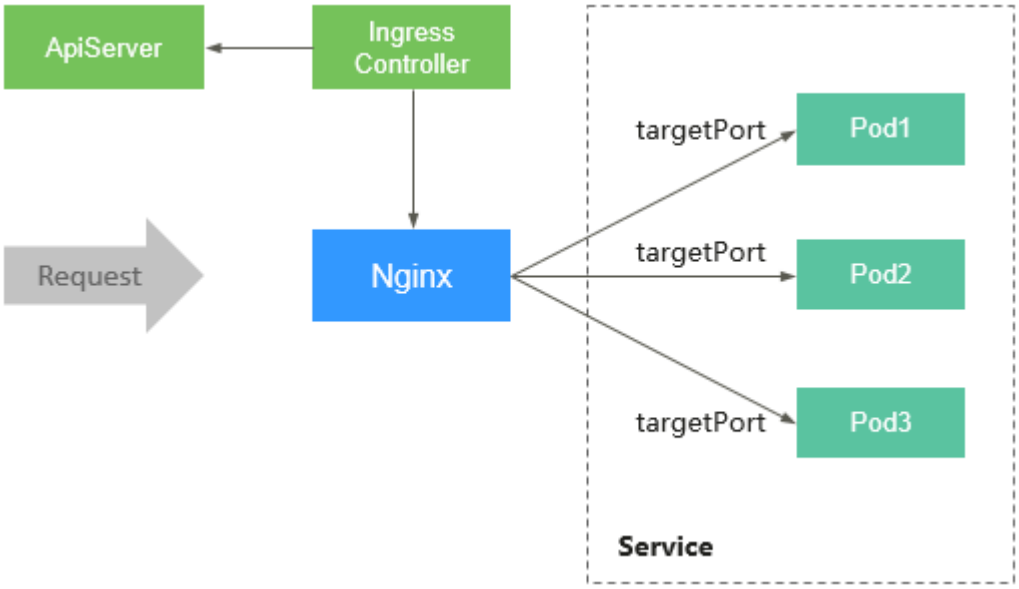
- When installing the NGINX Ingress Controller, you can specify Nginx parameters. These parameters take effect globally and are contained in the **nginx.conf** file. You can search for the parameters in **ConfigMaps**. If the parameters are not included in ConfigMaps, the configurations will not take effect.
- After the NGINX Ingress Controller is installed, you can interconnect the **ingress** you create on the CCE console with Nginx and set Nginx ingress functions using **Annotations**. For details about the supported annotation fields, see **Annotations**.
- Do not manually modify or delete the load balancer and listener that is automatically created by CCE. If the load balancer or listener is deleted, the workload will be abnormal. If you have modified or deleted them by mistake, uninstall the Nginx Ingress Controller and re-install it.

How the NGINX Ingress Controller Works

The 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 configuration changes apply. When the NGINX Ingress Controller detects that a 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 4-72** shows how the NGINX Ingress Controller 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 etcd and can be 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 4-72 How the NGINX Ingress Controller works



Constraints

- Dedicated load balancers with private IP addresses bound and used for network load balancing (over TCP or UDP) should be selected.
- During the upgrade of the NGINX Ingress Controller, 10s is reserved for deleting the NGINX Ingress Controller deployed on the backend servers associated with a load balancer.
- 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 creating a workload, you must have an available cluster. If no cluster is available, create one by performing the operations in **4.3.3 Buying a CCE Autopilot 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 on the left, choose **Add-ons**. Locate **NGINX Ingress Controller** on the right and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 4-74 NGINX Ingress Controller specifications

Parameter	Description
Instances	You can adjust the number of add-on instances as required.
Containers	You can adjust the container specifications of an add-on instance as required.

Step 3 Configure the add-on parameters.

- **Load Balancer**: Select a 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.
- **Admission Check**: If this option is enabled, an admission check will be performed on the configuration of Nginx ingresses. If the configuration files the admission check, requests to the ingresses will be intercepted. For details about verification, see **Access Control**.

  NOTE

  - Admission checks slow down the responses to ingress requests.
  - Admission checks are only available for the add-on v2.4.1 or later.
- **Nginx Parameters**: Configuring the **nginx.conf** file will affect all managed ingresses. You can search for related parameters through **ConfigMaps**. If the parameters you configured are not included in the options listed in ConfigMaps, the parameters will not take effect.

  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"
  }
  ```
- **Default 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.

Step 4 Click **Install**.

----End

Components

Table 4-75 NGINX Ingress Controller components

Component	Description	Resource Type
cceaddon-nginx-ingress-controller	Nginx-based ingress controller that provides flexible routing and forwarding for clusters.	Deployment
cceaddon-nginx-ingress-default-backend	Default backend of the Nginx ingress. The message "default backend - 404" is returned.	Deployment
cceaddon-nginx-ingress-admission-create	After webhooks are enabled, you can create a certificate for webhook verification.	Job
cceaddon-nginx-ingress-admission-patch	After webhooks are enabled, you can update the created certificate to the webhook configuration.	Job

4.11.6 CCE Advanced HPA

CCE Advanced HPA (cce-hpa-controller) is an in-house add-on, which can be used to flexibly scale in or out Deployments based on metrics such as CPU usage and memory usage.

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.

Constraints

To use CCE Advanced HPA, install an add-on that provides metrics APIs. Select one of the following add-ons based on your cluster version and actual requirements.

- **4.11.2 Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage. It is supported by all cluster versions.

- **4.11.3 Cloud Native Cluster Monitoring**: Auto scaling based on basic resource metrics. Prometheus needs to be registered as a metrics API. For details, see **Providing 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. Click **Add-ons** in the navigation pane, locate **CCE Advanced HPA** on the right, and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 4-76 CCE Advanced HPA specifications

Parameter	Description
Pods	Number of pods for the add-on. High availability is not possible if there is only one pod. If an error occurs on the node where the pod runs, the add-on will fail.
Containers	CPU and memory quotas of the container allowed for the selected add-on specifications. You can adjust the container specifications of an add-on instance as required.

Step 3 Click **Install**.

----End

Components

Table 4-77 cce-hpa-controller 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

4.12 Helm Chart

4.12.1 API Resource Restrictions on a Template

Resource	Restriction Item	Description	Recommended Alternative Solution
namespaces	-	Supported	For security purposes, CCE Autopilot does not allow you to deployment workloads in the system namespace (such as **kube-system**). Also, you cannot create, modify, delete, or execute any resources.
nodes	-	Supported	You can query nodes but cannot create, delete, and modify nodes.
persistentvolumeclaims	-	Supported	-
persistentvolumes	-	Supported	-
pods	HostPath	Mounting a file on the local host to a pod is not allowed.	Use emptyDir or cloud storage.
	HostNetwork	Mapping the host port to a pod is not allowed.	Use load balancing (**type=LoadBalancer**).
	HostPID	Sharing the host's PID namespace to pods is not allowed.	Users are unaware of the node. Do not need to use the restriction item.
	HostIPC	Container processes are not allowed to communicate with processes on the host.	Users are unaware of the node. Do not need to use the restriction item.
	NodeName	Scheduling pods to specific nodes is not allowed.	Users are unaware of the node. Do not need to use the restriction item.
	Privileged containers	Not supported	-

Resource	Restriction Item	Description	Recommended Alternative Solution
	Linux capabilities	**SETPCAP**, **MKNOD**, **AUDIT_WRITE**, **CHOWN**, **DAC_OVERRIDE**, **FOWNER**, **FSETID**, **KILL**, **SETGID**, **SETUID**, **NET_BIND_SERVICE**, **SYS_CHROOT**, **SETFCAP**, and **SYS_PTRACE** are supported. You can also enable **NET_RAW**, **SYS_PTRACE**, and **NET_ADMIN** by setting **SecurityContext**.	Use allowed values.
	Node affinity and anti-affinity	Pods cannot be scheduled to specified nodes or nodes with certain labels, or a batch of pods cannot be scheduled to nodes with certain labels.	• You do not need to specify a node for scheduling, but you can specify a pod to an AZ. • A batch of pods can be scheduled to multiple AZs.
	allowPrivilegeEscalation (whether privilege escalation is allowed)	Not supported	Keep the default settings.
	RuntimeClassName	Not supported	Keep the default settings.
	Time zone synchronization (the **/etc/localtime** file on the host)	Not supported	Keep the default settings.
serviceaccounts	-	System configurations cannot be modified, and system roles cannot be bound.	Keep the default settings.

Resource	Restriction Item	Description	Recommended Alternative Solution
services	-	Services of the NodePort type are not allowed, and only dedicated load balancer can be used for Services.	Use load balancing (**type=LoadBalancer**).
daemonsets	apps	DaemonSets are not allowed.	Deploy multiple images in a pod using sidecars.
deployments	apps	Supported. The restricted fields are the same as those in **pods**.	Use allowed values.
replicasets	apps	Supported. The restricted fields are the same as those in **pods**.	Use allowed values.
statefulsets	apps	Supported. The restricted fields are the same as those in **pods**.	Use allowed values.
cronjobs	batch	Supported. The restricted fields are the same as those in **pods**.	Use allowed values.
jobs	batch	Supported. The restricted fields are the same as those in **pods**.	Use allowed values.
clusterrolebindings	rbac.authorization.k8s.io	Supported. The system group, system user, and cce-service group cannot be bound.	Use allowed values.
rolebindings	rbac.authorization.k8s.io	Supported. The system group, system user, and cce-service group cannot be bound.	Use allowed values.
storageclasses	storage.k8s.io	OBS and EVS storage classes cannot be created. Other functions are supported.	Use allowed values.

4.12.2 Deploying an Application from a Chart

On the CCE console, you can upload a Helm chart package, deploy it, and manage the deployed pods.

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 4-78**, the parameters marked with * are mandatory.

  Table 4-78 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 on the left, 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. In the navigation pane on the left, choose **App Templates** and then click **Upload Chart** in the upper right corner.

Step 2 Click **Select File**, select the chart to be uploaded, and click **Upload**.

----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 on the left, choose **App Templates**.

Step 2 On the **My Charts** tab, click **Install** of the target chart.

Step 3 Set workload installation parameters by referring to **Table 4-79**.

Table 4-79 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. In the navigation pane on the left, choose **App Templates** and then 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. Click **Upgrade**, and then click **Submit**.

Step 5 Click **Back to Release List**. If the chart status changes to **Upgrade successful**, the workload is successfully 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. In the navigation pane on the left, choose **App Templates** and then click the **Releases** tab.

Step 2 Click **More** > **Roll Back** for the workload to be rolled back, select the workload version, and click **Roll back to this version**.

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. In the navigation pane on the left, choose **App Templates** and then 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

