Cloud Container Engine Autopilot

User Guide

Issue	01
Date	2024-11-21

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions

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 Clusters

1.1 Kubernetes Version Release Notes

1.1.1 Kubernetes 1.28 Release Notes

CCE has passed the Certified Kubernetes Conformance Program and is a certified Kubernetes offering. CCE allows you to create clusters of Kubernetes 1.28. This topic 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 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 alpha stage are disabled by default, those in beta stage are enabled by default, and those in GA stage are always enabled and they cannot be disabled.

The function of turning on or off the features in GA stage will be removed in later Kubernetes versions. CCE policies for new features are the same as those in the community.

- The version skew policy is expanded to three versions.

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

- Retroactive Default StorageClass moves to GA.

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

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

- Native sidecar containers are introduced.

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

- Mixed version proxy is introduced.

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

- Non-graceful node shutdown moves to GA.

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

- NodeSwap moves to beta.

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

- Two job-related features are added.

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

  - Delayed creation of replacement pods

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

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

  - Backoff limit per index

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

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

- Some CEL related features are improved.

  CEL related capabilities are enhanced.

  - CEL used to validate CRDs moves to beta.

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

  - CEL admission control graduates to beta.

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

    - ValidatingAdmissionPolicy can correctly handle the **authorizer** variable.

        - ValidatingAdmissionPolicy can have the **messageExpression** field checked.
        - The ValidatingAdmissionPolicy controller is added to kube-controller-manager to check the type of the CEL expression in ValidatingAdmissionPolicy and save the reason in the **status** field.
        - CEL expressions can contain a combination of one or more variables, which can be defined in ValidatingAdmissionPolicy. These variables can be used to define other variables.
        - CEL library functions can be used to parse resources specified by **resource.Quantity** in Kubernetes.
- Other features
    - The ServiceNodePortStaticSubrange feature gate moves to beta. With this feature enabled, static port range can be reserved to avoid conflicts with dynamically allocated ports. For details, see **Avoiding Collisions Assigning Ports to NodePort Services**.
    - The alpha feature ConsistentListFromCache is added to allow the API server to serve consistent lists from cache. Get and list requests can read data from the cache instead of etcd.
    - In Kubernetes 1.28, kubelet can configure the drop-in directory (alpha). This feature allows you to add support for the **--config-dir** flag to kubelet so that you can specify an insert directory that overwrites the kubelet configuration in **/etc/kubernetes/kubelet.conf**.
    - ExpandedDNSConfig moves to GA and is enabled by default. With this feature enabled, DNS configurations can be expanded.
    - The alpha feature CRDValidationRatcheting is added. This feature allows CRs with failing validations to pass if a Patch or Update request does not alter any of the invalid fields.
    - **--concurrent-cron-job-syncs** is added to kube-controller-manager to configure the number of workers for the cron job controller.

API Changes and Removals

- **NetworkPolicyStatus** is removed. There is no status attribute in a network policy.
- **annotationbatch.kubernetes.io/cronJob-scheduled-timestamp** is added to job objects to indicate the creation time of a job.
- The **podReplacementPolicy** and **terminating** fields are added to job APIs. With these fields specified, once a previously created pod is terminated in a job, the job immediately starts a new pod to replace the pod. The new fields allow you to specify whether to replace the pod immediately after the previous pod is terminated (original behavior) or replace the pod after the existing pod is completely terminated (new behavior). This is an alpha feature, and you can enable it by turning on the **JobPodReplacementPolicy** feature gate in your cluster.
- The **BackoffLimitPerIndex** field is available in a job. Pods specified by a job share a backoff mechanism. When backoff times of the job reach the limit, this job is marked as failed and resources, including indexes that are not

running, are cleared up. This field allows you to configure backoff limit for a single index. For details, see **Backoff limit per index**.

- The **ServedVersions** field is added to the **StorageVersion** API. This change is introduced by mixed version proxy. The new field is used to indicate a version that can be provided by the API server.
- **SelfSubjectReview** is added to **authentication.k8s.io/v1**, and **kubectl auth whoami** goes to GA.
- **LastPhaseTransitionTime** is added to **PersistentVolume**. The new field is used to store the last time when a volume changes to a different phase.
- **resizeStatus** in **PVC.Status** is replaced by **AllocatedResourceStatus**. The new field indicates the statuses of the storage resize operation. The default value is an empty string.
- If **hostNetwork** is set to **true** and ports are specified for a pod, the **hostport** field will be automatically configured.
- StatefulSet pods have the pod index set as a pod label **statefulset.kubernetes.io/pod-index**.
- **PodHasNetwork** in the **Condition** field of pods has been renamed to **PodReadyToStartContainers**. The new field specifies that containers are ready to start after the network, volumes, and sandbox pod have been created.
- A new configuration option **delayCacheUntilActive** is added to **KubeSchedulerConfiguration**. If **delayCacheUntilActive** is set to **true**, kube-scheduler on the leader will not cache scheduling information. This reduces the memory pressure of other master nodes, but slows down the failover speed after the leader failed.
- The **namespaceParamRef** field is added to **admissionregistration.k8s.io/v1alpha1.ValidatingAdmissionPolicy**.
- The **reason** and **fieldPath** fields are added to CRD validation rules to allow you to specify reason and field path after verification failed.
- The CEL expression of ValidatingAdmissionPolicy supports namespace access via namespaceObject.
- API groups ValidatingAdmissionPolicy and ValidatingAdmissionPolicyBinding are promoted to betav1.
- A ValidatingAdmissionPolicy now has its **messageExpression** field checked against resolved types.

Feature Gate and Command Line Parameter Changes and Removals

- **–short** is removed from kubelet. Therefore, the default output of **kubectl version** is the same as that of **kubectl version –short**.
- **--volume-host-cidr-denylist** and **--volume-host-allow-local-loopback** are removed from kube-controller-manager. **--volume-host-cidr-denylist** is a comma-separated list of CIDR ranges. Volume plugins at these IP addresses are not allowed. If **--volume-host-allow-local-loopback** is set to **false**, the local loopback IP address and the CIDR ranges specified in **--volume-host-cidr-denylist** are disabled.
- **--azure-container-registry-config** is deprecated in kubelet and will be deleted in later Kubernetes versions. Use **--image-credential-provider-config** and **--image-credential-provider-bin-dir** instead.

- **--lock-object-namespace** and **--lock-object-name** are removed from kube-scheduler. Use **--leader-elect-resource-namespace** and **--leader-elect-resource-name** or **ComponentConfig** instead. (**--lock-object-namespace** is used to define the namespace of a lock object, and **--lock-object-name** is used to define the name of a lock object.)
- KMS v1 is deprecated and will only receive security updates. Use KMS v2 instead. In later Kubernetes versions, use **--feature-gates=KMSv1=true** to configure a KMS v1 provider.
- The DelegateFSGroupToCSIDriver, DevicePlugins, KubeletCredentialProviders, MixedProtocolLBService, ServiceInternalTrafficPolicy, ServiceIPStaticSubrange, and EndpointSliceTerminatingCondition feature gates are removed.

References

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

1.1.2 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 tolerations) in the job's pod template can be modified. For details, see **Mutable Scheduling Directives**.
- Downward API hugepages are stable.

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

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

- Accessing node logs using Kubernetes APIs is supported.

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

- ReadWriteOncePod access mode moves to beta.

  Kubernetes 1.22 introduced a ReadWriteOncePod access mode for PVs and PVCs. This feature has evolved into the beta phase. A volume can be mounted to a single pod in read/write mode. Use this access mode if you want to ensure that only one pod in the cluster can read that PVC or write to it. For details, see **Access Modes**.

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

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

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

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

- VolumeManager reconstruction goes to beta.

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

- Server side field validation and OpenAPI V3 are stable.

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

- StatefulSet start ordinal moves to beta.

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

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

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

- StatefulSet PVC auto deletion moves to beta.

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

- Volume group snapshots are introduced.

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

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

  In Kubernetes 1.5, the **--prune** flag was introduced in **kubectl apply** to delete resources that are no longer needed. This allowed **kubectl apply** to automatically clear resources removed from the current configuration. However, the existing implementation of **--prune** has design defects that degrade its performance and lead to unexpected behaviors. In Kubernetes 1.27, **kubectl apply** provides ApplySet-based pruning, which is in the alpha phase. For details, see **Declarative Management of Kubernetes Objects Using Configuration Files**.

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

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

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

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

- Pod startup is accelerated.

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

- KMS V2 moves to beta.

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

Deprecations and Removals

- In Kubernetes 1.27, the feature gates that are used for volume extension and in the GA status, including ExpandCSIVolumes, ExpandInUsePersistentVolumes, and ExpandPersistentVolumes are removed and can no longer be referenced in the **--feature-gates** flag.

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

the dockershim code is removed. This parameter was deprecated in 1.24 and later versions and is removed from Kubernetes 1.27.

References

For more details about the performance comparison and function evolution between Kubernetes 1.27 and other versions, see **Kubernetes v1.27 Release Notes**.

1.2 CCE Autopilot Cluster Version Release Notes

Indexes

- **v1.28**
- **v1.27**

v1.28

Table 1-1 Release notes for the v1.28 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Update	Enhancement	Vulnerability Fixing
v1.28.5-r0	v1.28.3	● Supported APM probes during workload creation. ● Supported access to kube-apiserver using a private IP address.	-	Fixed some security issues.
v1.28.4-r0	v1.28.3	● Custom metrics, such as network and disk metrics, can be used create HPA policies. ● kube-apiserver can be accessed from a public network.	When YAML is used to create an application, the parameters that are not supported by CCE Autopilot and do not affect functionality were automatically ignored.	Fixed some security issues.
v1.28.3-r0	v1.28.3	● Supported automatic cluster upgrade. ● Hosted the Everest storage add-on at the backend. ● Supported OBS volumes.	-	Fixed some security issues.

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 1-2 Release notes for the v1.27 patch

CCE Cluster Patch Version	Kubernetes Version	Feature Update	Enhancement	Vulnerability Fixing
v1.27.7-r0	v1.27.4	- Supported APM probes during workload creation. - Supported access to kube-apiserver using a private IP address.	-	Fixed some security issues.
v1.27.6-r0	v1.27.4	- Custom metrics, such as network and disk metrics, can be used create HPA policies. - kube-apiserver can be accessed from a public network.	When YAML is used to create an application, the parameters that are not supported by CCE Autopilot and do not affect functionality were automatically ignored.	Fixed some security issues.

CCE Cluster Patch Version	Kubernetes Version	Feature Update	Enhancement	Vulnerability Fixing
v1.27.5-r0	v1.27.4	- Supported automatic cluster upgrade. - Hosted the Everest storage add-on at the backend. - Supported OBS volumes.	-	Fixed some security issues.
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**.	-	-

1.3 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

- 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	Minimum Usage	Region Limits	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	Select one VPC for each cluster to provide an isolated, private virtual network environment for the cluster.	Maximum number of VPCs that can be created by each account in a region: 5	
	Subnet	1 per cluster	At least one subnet must be selected for each cluster to allocate container IP addresses. By default, the cluster control plane occupies eight IP addresses for control plane deployment and interconnection with external services.	Maximum number of subnets that can be created by each account in a region: 50	

Service	Quota Item	Minimum Usage	Minimum Usage	Region Limits	Quota Increase
	Security group	2 per cluster	Two security groups are automatically created for each cluster for network access control of the cluster control plane and elastic network interfaces.	Maximum number of security groups that can be created by each account in a region: 100	
	Security group rules	7 per cluster	Seven security group rules are automatically added for each cluster to allow traffic over specified ports and ensure normal network communication in the 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	Reserve at least three endpoints for each cluster so that the cluster can access peripheral services such as SWR and OBS.	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	Each cluster requires at least two private domain names for normal communication within the cluster or across clusters.	Maximum number of private zones that can be created by each account in a region: 50	
	Record set	6 per cluster	Each cluster requires at least six DNS record sets for mapping specified domain names to IP addresses or other domain names in the 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**.
Cluster Name	Enter a cluster name. Cluster names in the same account must be unique.
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. For more information, see **Enterprise Management**. After you select an enterprise project (for example, default), the cluster and resources in the cluster are created in the selected enterprise project. For easier resource management, do not change the enterprise project after the cluster is created.
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.
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**.

Parameter	Description
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
Alarm Center	Alarm Center provides comprehensive cluster alarm capabilities so that alarms can be generated in a timely manner when faults occur during cluster running, ensuring service stability. If this option is enabled, the default alarm rules will be created, and notifications will be sent to the selected contact group. For details, see **Configuring Alarms in Alarm Center**.
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**. ● A tag key can have no more than 128 characters and must not begin with _sys_. It can only contain letters, digits, spaces, and the following special characters: -_.:=+@. The key cannot be empty. ● A tag value can have a maximum of 255 characters. It can only contain letters, digits, spaces, and the following special characters: -_.:/=+@. The value can be empty.
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 (**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 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 (**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 (**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 **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, but 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 **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 **Connecting to a Cluster Using kubectl**.
- Connect to multiple clusters using kubectl. For details, see **Connecting to Multiple Clusters Using kubectl**.

1.4 Connecting to a Cluster

1.4.1 Connecting to a Cluster Using kubectl

Scenario

You can use kubectl to connect a cluster.

Permissions

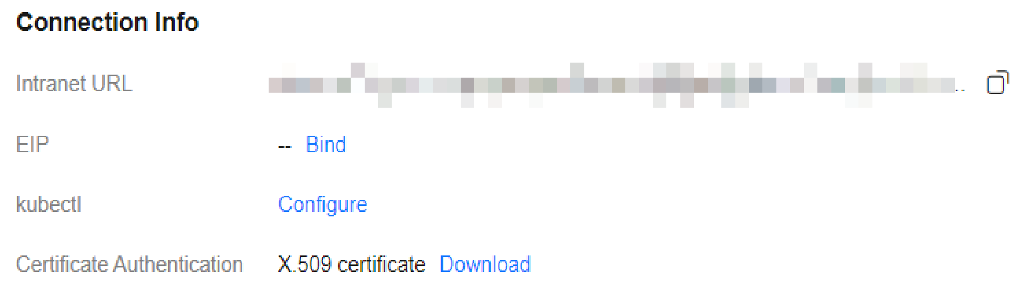
When you access a cluster using kubectl, CCE Autopilot uses **kubeconfig.json** generated on the cluster for authentication. This file contains user information, based on which CCE Autopilot determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a **kubeconfig.json** 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 1-1 Cluster connection information



Connection Info

Intranet URL	[redacted]
EIP	-- Bind
kubectl	Configure
Certificate Authentication	X.509 certificate Download

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.

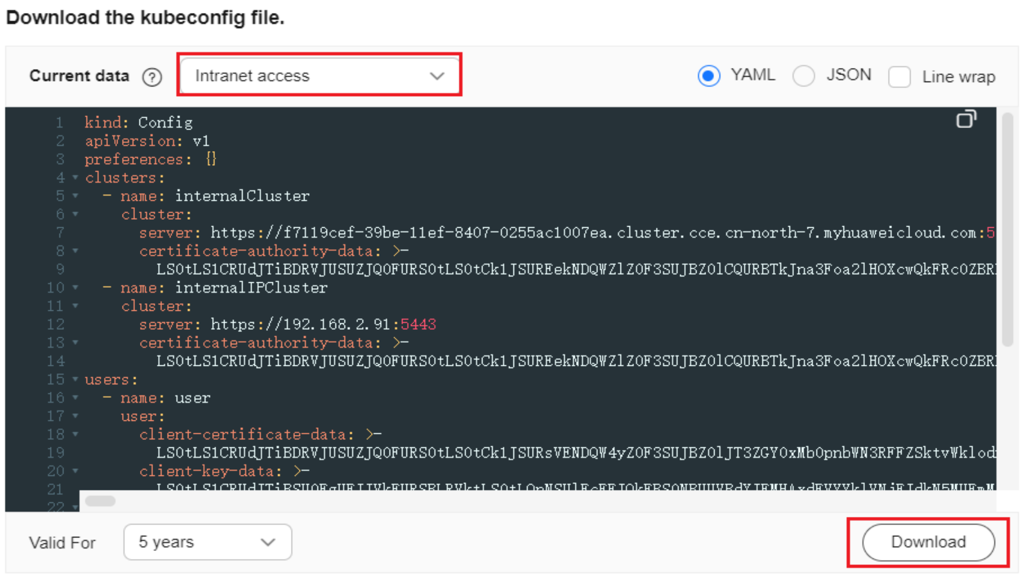
```
chmod +x kubectl
mv -f kubectl /usr/local/bin
```

Step 3 **Obtain the kubectl configuration file.**

In the **Connection Info** pane on the **Overview** page, click **Configure** next to **kubectl** to check the kubectl connection. On the displayed page, select the access method and download the configuration file.

- If a cluster version is v1.27.7-r0, v1.28.5-r0, or later, access via private domain names, access via private IP addresses, and public network access are supported.
- If a cluster version is earlier than v1.27.7-r0 or v1.28.5-r0, only private network access and public network access are supported. For private network access, only domain names can be used for access.

Figure 1-2 Downloading the configuration file



Download the kubeconfig file.

Current data ⓘ Intranet access ● YAML ○ JSON □ Line wrap

```
kind: Config
apiVersion: v1
preferences: {}
clusters:
  - name: internalCluster
    cluster:
      server: https://f7119cef-39be-11ef-8407-0255ac1007ea.cluster.cce.cn-north-7.myhuaweicloud.com:5
      certificate-authority-data: >-
        LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUREekNDQWZlZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBR
  - name: internalIPCluster
    cluster:
      server: https://192.168.2.91:5443
      certificate-authority-data: >-
        LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUREekNDQWZlZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBR
users:
  - name: user
    user:
      client-certificate-data: >-
        LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURsVENDQW4yZ0F3SUJBZ0lJT3ZGY0xMb0pnbWN3RFFZSktvWklod
      client-key-data: >-
```

Valid For 5 years Download

- The kubectl configuration file (**kubeconfig**) is used for cluster authentication. If the file is leaked, your clusters 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 the configuration file (for example, **kubeconfig.yaml**) 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.yaml $HOME/.kube/config
```

3. Switch the kubectl access mode based on service scenarios.
   - To enable access within a VPC using domain names, run the following command:

   ```
   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
```

----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 **Configure the kubectl authentication file**.

Related Operations

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

1.4.2 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 1-3**).

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 can be used only after CoreDNS is installed in a cluster.
- CloudShell cannot be used across accounts or in or sub-projects.

Figure 1-3 CloudShell

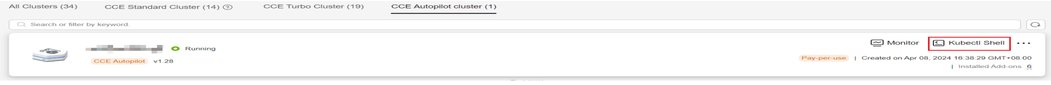
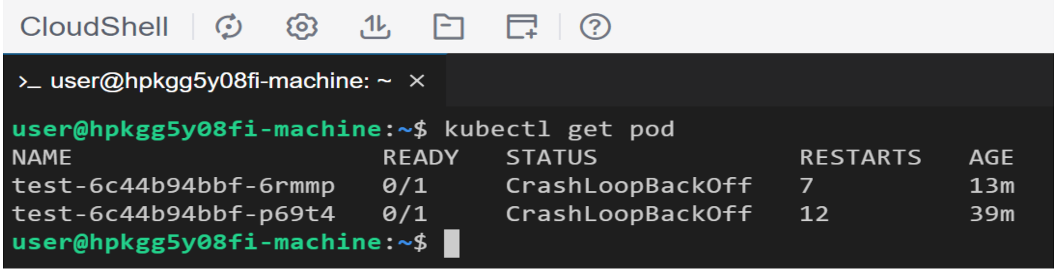


Figure 1-4 Using kubectl in CloudShell



```
user@hpkgg5y08fi-machine:~$ kubectl get pod
NAME                      READY   STATUS             RESTARTS   AGE
test-6c44b94bbf-6rmmp     0/1     CrashLoopBackOff   7          13m
test-6c44b94bbf-p69t4     0/1     CrashLoopBackOff   12         39m
user@hpkgg5y08fi-machine:~$
```

1.4.3 Connecting to a Cluster Using an X.509 Certificate

Scenario

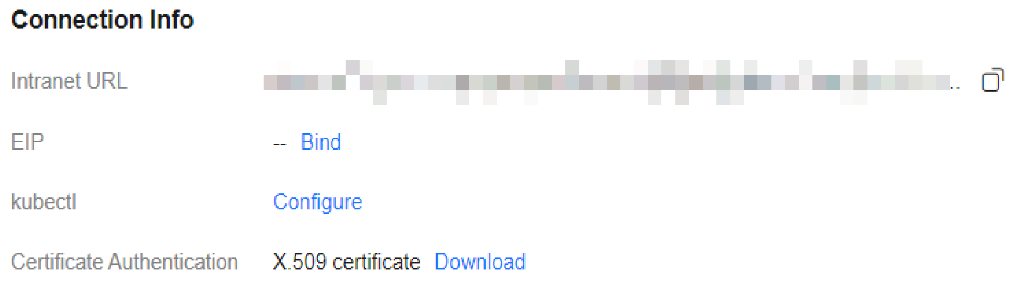
This section describes how to obtain the cluster certificate from the console and use it to access Kubernetes clusters.

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 1-5 Downloading a cluster certificate



Connection Info

Intranet URL [redacted]

EIP -- Bind

kubectl Configure

Certificate Authentication X.509 certificate Download

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 obtain the pod information. In the following information, ******:5443* indicates the private IP address or EIP and port number of the API server in the cluster.

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

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

----End

1.4.4 Configuring API Server for a Cluster for Internet Access

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

Binding an EIP to an API Server

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

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

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

NOTE

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

Step 4 Click **OK**.

----End

Configuring Access Policies for an API Server

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

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

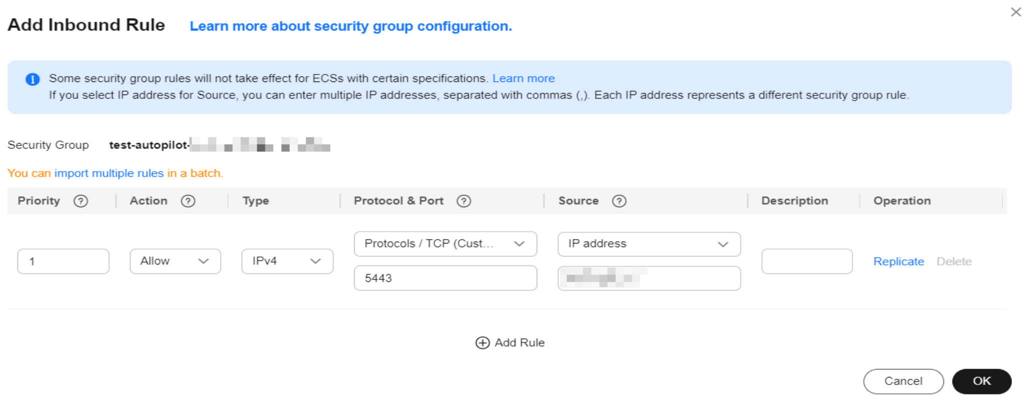
Step 2 Log in to the VPC console. In the navigation pane on the left, choose **Access Control** > **Security Groups**.

Step 3 Select **Description** as the filter criterion and paste the cluster ID to search for the target security groups.

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

Step 5 Click **Add Rule**.

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



Add Inbound Rule Learn more about security group configuration.

Some security group rules will not take effect for ECSs with certain specifications. Learn more
If you select IP address for Source, you can enter multiple IP addresses, separated with commas (,). Each IP address represents a different security group rule.

Security Group test-autopilot-[redacted]

You can import multiple rules in a batch.

Priority	Action	Type	Protocol & Port	Source	Description	Operation
1	Allow	IPv4	Protocols / TCP (Cust... 5443	IP address [redacted]		Replicate Delete

Step 6 Click **Confirm**.

----End

1.5 Managing a Cluster

1.5.1 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.)
  - 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, **Frozen** 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 (bound to PVs) in a cluster.
- 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

1.6 Upgrading a Cluster

1.6.1 Upgrade Overview

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

To ensure your service rights and benefits, upgrade your Kubernetes clusters before a maintenance period ends. Proactive cluster upgrades help you:

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

You can check Kubernetes cluster versions on the cluster list page and view if there is a new version for a cluster to upgrade. If necessary, obtain the expert consultation service.

Cluster Upgrade Path

CCE Autopilot 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.

- Kubernetes version

Source Kubernetes Version	Target Kubernetes Version
v1.27	v1.28
v1.28	/

NOTE

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.

- Patch version

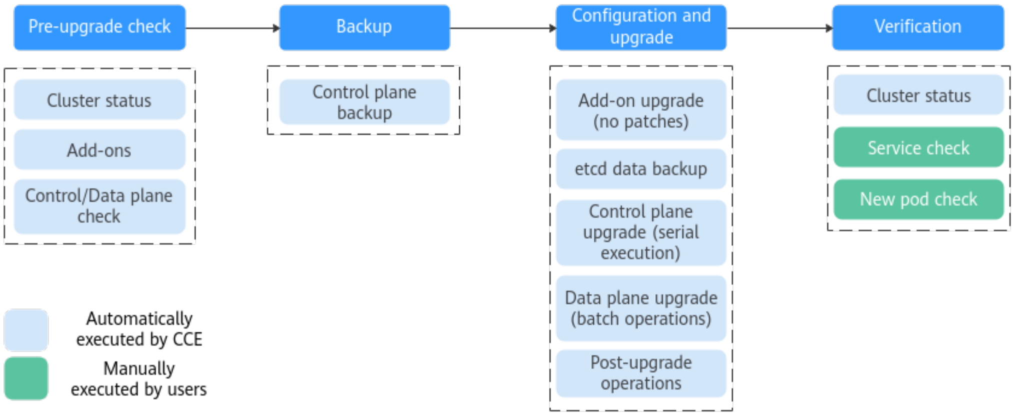
  Patch version management is available for CCE Autopilot clusters to provide new features and fix bugs and vulnerability for in-maintenance clusters without requiring a major version upgrade.

  After a new patch version is released, you can upgrade to the latest patch version at any time.

Cluster Upgrade Process

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

Figure 1-6 Process of upgrading a cluster


Pre-upgrade check
Cluster status
Add-ons
Control/Data plane check
Backup
Control plane backup
Configuration and upgrade
Add-on upgrade (no patches)
etcd data backup
Control plane upgrade (serial execution)
Data plane upgrade (batch operations)
Post-upgrade operations
Verification
Cluster status
Service check
New pod check
Automatically executed by CCE
Manually executed by users

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 workloads to ensure that the cluster meets the upgrade requirements. For more details, see **Troubleshooting for Pre-upgrade Check Exceptions**. If any check item is abnormal, rectify the fault as prompted on the console.

2. **Backup**

   You can use disk snapshots to back up the control plane details such as CCE component images, component configurations, and etcd data. Back up your data before an upgrade. If unexpected cases occur during an upgrade, you can use the backup to quickly restore the cluster.

Backup Method	Backup Object	Backup Type	Backup Time	Rollback Time	Description
etcd data backup	etcd data	Automatic backup during the upgrade	1–5 minutes	2 hours	Mandatory. The backup is automatically performed during the upgrade.
EVS snapshot backup	Control plane details, including component images, configurations, logs, and etcd data	One-click backup on web pages (manually triggered)	1–5 minutes	20 minutes	-

3. **Configuration and upgrade**

   Configure parameters before an upgrade. CCE Autopilot has provided default settings, which can be modified as needed. After the configuration, upgrade add-ons, control plane, and data plane in sequence.

   – **Add-on Upgrade Configuration**: Add-ons that have been installed in your cluster are listed. During the cluster upgrade, CCE Autopilot 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.

4. **Post-upgrade verification**

   After an upgrade, CCE Autopilot will automatically check items such as the cluster status. You need to manually check services and new pods to ensure that the cluster functions properly after the upgrade.

1.6.2 Before You Start

Before the upgrade, you can check whether your cluster can be upgraded and which versions are available on the CCE console. For details, see **Upgrade Overview**.

Precautions

Before upgrading a cluster, note the following:

- **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 Version 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.

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

- Before upgrading a cluster, **ensure no high-risk operations are performed in the cluster**. If there are high-risk operations, the cluster upgrade may fail or the configuration may be lost after the upgrade. For example, modifying the configuration of a listener configured for CCE on the ELB console is a high-risk operation. It recommended that you modify configurations on the CCE console so that the modifications can be automatically inherited during the upgrade.

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 can be rolled back using the backup data.

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**.

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.

Upgrade Backup

How to back up a node:

Backup Method	Backup Object	Backup Type	Backup Time	Rollback Time	Description
etcd data backup	etcd data	Automatic backup during the upgrade	1–5 minutes	2 hours	Mandatory. The backup is automatically performed during the upgrade.
EVS snapshot backup	Control plane details, including component images, configurations, logs, and etcd data	One-click backup on web pages (manually triggered)	1–5 minutes	20 minutes	-

1.6.3 Automatic Upgrade

CCE Autopilot clusters support automatic upgrades to ensure cluster stability and security. You can enable automatic upgrades by specifying the maintenance configuration for each cluster. The cluster is periodically upgraded to a later patch version (for example, from v1.27.2-r0 to v1.27.3-r0) based on the cluster maintenance configuration and cluster version.

NOTICE

- After automatic cluster upgrades are enabled, a cluster will not be upgraded immediately. If you want to perform the upgrade immediately, perform a manual upgrade by following the instructions in **Manual Upgrade**.
- Before enabling automatic upgrades, learn about the upgrade mode and upgrade impacts. For details, see **Upgrade Overview** and **Before You Start**.

Maintenance Configuration

In the scheduled maintenance window, CCE Autopilot automatically performs the pre-upgrade check and upgrade operations. By specifying the maintenance configuration, you can flexibly control the execution period of each automatic upgrade to minimize risks. For example, for retail businesses, you can set the maintenance window to be performed only in the early morning of working days and close it during important activities.

If you modify the cluster maintenance configuration before an automatic upgrade, the upgrade will be performed based on the new cluster maintenance configuration. If you modify the maintenance configuration during an automatic upgrade, the upgrade will not be stopped.

Upgrade Plan

Based on the cluster maintenance configuration and CCE version, an automatic upgrade plan is generated one week later at 01:00 every Monday, and an event is sent to you seven days in advance.

If you change the maintenance configuration between 00:00 and 01:00 on Monday, the upgrade plan will not be scheduled until 01:00 on the next Monday. If you want to cancel an automatic upgrade, manually cancel the upgrade plan on the console. If you manually cancel an upgrade plan, only the current upgrade is canceled, and subsequent automatic upgrades will not be affected.

A pre-upgrade check is automatically performed every day three days before the automatic upgrade. If the check fails, an event is sent. You are advised to configure AOM alarm rules to send email or SMS notifications so you can handle the failure in a timely manner to ensure successful automatic upgrade.

Enabling Automatic Cluster Upgrades

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

Step 2 On the right of the **Overview** page, locate **Maintenance Configuration** and then click **Configure**.

Step 3 Specify the frequency, start time, and duration. For example, you can configure a weekly repeated maintenance window (four hours a day from Monday to Friday).

NOTE

It is recommended that you set the cluster maintenance window to three hours or longer.

Step 4 Click **OK**. An upgrade plan will be generated based on the cluster maintenance configuration and will be automatically completed during the configured maintenance window.

----End

1.6.4 Manual Upgrade

You can upgrade your clusters to a newer version on the CCE console.

Before the upgrade, learn about the target version to which each CCE cluster can be upgraded in what ways, and the upgrade impacts. For details, see **Upgrade Overview** and **Before You Start**.

Precautions

- During the cluster upgrade, the system will automatically upgrade add-ons to a version compatible with the target cluster version. Do not uninstall or reinstall add-ons during the cluster upgrade.
- Before the upgrade, ensure that all add-ons are running. If an add-on fails to be upgraded, rectify the fault and try again.

- If an upgrade failure message is displayed during the cluster upgrade, rectify the fault as prompted and try again. If upgrade attempts fail again, **submit a service ticket** for assistance.

For more information, see **Before You Start**.

Procedure

The cluster upgrade goes through check, backup, configuration and upgrade, and verification.

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 **Cluster Upgrade**.

Step 3 CCE automatically provides you with an optimal upgrade path based on the current cluster version. Select the target cluster version, check information such as version differences, and add-on versions, and click **Go to Upgrade**.

Step 4 Perform the pre-upgrade check. Click **Start Check** and confirm the check. If there are abnormal or risky items in the cluster, handle the exceptions based on the check results displayed on the page and check again.

- **Exceptions**: View the solution displayed on the page, handle the exceptions and check again.
- **Risk Items**: may affect the cluster upgrade. Check the risk description and see whether you may be impacted. If no risk exists, click **OK** next to the risk item to manually skip this risk item and check again.

After the check is passed, click **Next**.

Step 5 Back up the cluster. During the cluster upgrade, CCE automatically backs up etcd data. You can manually back up the control plane to speed up the rollback if the control plane fails to upgrade. If manual backup is not required, click **Next**.

Backup Method	Backup Object	Backup Type	Backup Time	Rollback Time	Description
etcd data backup	etcd data	Automatic backup during the upgrade	1–5 minutes	2 hours	Mandatory. The backup is automatically performed during the upgrade.
EVS snapshot backup	Control plane details, including component images, configurations, logs, and etcd data	One-click backup on web pages (manually triggered)	1–5 minutes	20 minutes	-

Step 6 Configure the upgrade parameters.

- **Add-on Upgrade Configuration**: Add-ons that have been installed in your cluster are listed. During the cluster upgrade, CCE Autopilot 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.

Step 7 After the configuration is complete, click **Upgrade** and confirm the upgrade. The cluster starts to be upgraded. You can view the process in the lower part of the page.

NOTE

If an upgrade failure message is displayed during the cluster upgrade, rectify the fault as prompted and try again.

Step 8 After the upgrade is complete, click **Next**. Verify the upgrade based on the displayed check items. After confirming that all check items are normal, click **Complete** and confirm that the post-upgrade check is complete. For details, see Performing Post-Upgrade Verification.

You can verify the cluster Kubernetes version on the **Clusters** page.

----End

FAQs

- **What do I do if a cluster add-on fails to be upgraded during the CCE cluster upgrade?**

1.6.5 Performing Post-Upgrade Verification

1.6.5.1 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.

1.6.5.2 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.

1.6.5.3 New Pod Check

Check Items

Check whether pods can be created after the cluster is upgraded.

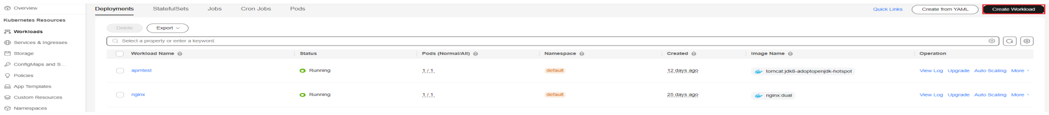
Procedure

Create pods and ensure the new and existing pods provide the same services.

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**. On the displayed page, click **Create Workload** or **Create from YAML** in the upper right corner. For details about how to create a workload, see **Creating a Workload**.

Figure 1-7 Creating a workload



Workload Name	Status	Pods (Normal/All)	Namespace	Created	Image Name	Operation
apmtest	Running	1 / 1	default	12 days ago	tomcat:jdk8-adoptopenjdk-hotspot	View Log Upgrade Auto Scaling More
nginx	Running	1 / 1	default	25 days ago	nginx:dual	View Log Upgrade Auto Scaling More

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.

```
apiVersion: apps/v1
kind: Deployment
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, locate the **post-upgrade-check** workload and choose **More** > **Delete** in the **Operation** column to delete the test workload.

----End

Solution

If pods cannot be created or are abnormal, submit a service ticket.

1.6.6 Troubleshooting for Pre-upgrade Check Exceptions

1.6.6.1 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 1-3 Check items

No.	Check Item	Description
1	**Upgrade Management**	Check whether the target cluster is under upgrade management.
2	**Add-ons**	● Check whether the add-on status is normal. ● Check whether the add-on support the target version.

No.	Check Item	Description
3	**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.
4	**SSH Connectivity of Master Nodes**	Check whether your master nodes can be accessed using SSH.
5	**Discarded Kubernetes Resources**	Check whether there are discarded resources in the cluster.
6	**cce-hpa-controller Limitations**	Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.
7	**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.
8	**HTTPS Load Balancer Certificate Consistency**	Check whether the certificate used by an HTTPS load balancer has been modified on ELB.

1.6.6.2 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 based on logs displayed on the console.

1.6.6.3 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. In the navigation pane on the left, choose **Add-ons** and then obtain add-ons. Then, handle malfunctional add-ons.

- **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. In the navigation pane on the left, choose **Add-ons** and then manually upgrade the 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. In the navigation pane on the left, choose **Add-ons** and then manually uninstall the add-on. For details about the supported add-on versions and substitutions, see **Add-ons**.

- **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. In the navigation pane on the left, choose **Add-ons** and then manually upgrade the add-on.

1.6.6.4 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.

1.6.6.5 SSH Connectivity of Master Nodes

Check Items

Check whether your master nodes can be accessed using SSH.

Solution

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

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

1.6.6.6 Discarded Kubernetes Resources

Check Items

Check whether there are discarded resources in the clusters.

Solution

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

  Error log:

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

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

  ```
  publishNotReadyAddresses: true
  ```

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

  Error log:

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

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

1.6.6.7 cce-hpa-controller Limitations

Check Items

Check whether there are compatibility limitations between the current and target cce-controller-hpa add-on versions.

Solution

There are compatibility limitations between the current and target versions of the cce-controller-hpa add-on. To address this, install an add-on that provides metrics APIs, like metrics-server, in the cluster.

Install the metrics add-on in the cluster and try again.

1.6.6.8 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.

There are no version compatibility differences when v1.27 is upgraded to v1.28.

1.6.6.9 HTTPS Load Balancer Certificate Consistency

Check Items

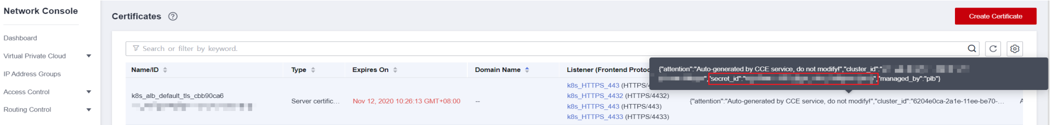
Check whether the certificate used by an HTTPS load balancer has been modified on ELB.

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 1-8 Viewing a certificate



Name/ID	Type	Expires On	Domain Name	Listener (Frontend Protoc	
k8s_alb_default_tls_cbb90ca6 [redacted]	Server certific...	Nov 12, 2020 10:26:13 GMT+08:00	--	k8s_HTTPS_443 (HTTPS/ k8s_HTTPS_4432 (HTTPS/4432) k8s_HTTPS_443 (HTTPS/443) k8s_HTTPS_4433 (HTTPS/4433)	{"attention":"Auto-generated by CCE service, do not modify!","cluster_id":"6204e0ca-2a1e-11ee-be70-...

{"attention":"Auto-generated by CCE service, do not modify!","cluster_id":"[redacted]","secret_id":"[redacted]","managed_by":"plb"}

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 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.

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 **Services & Ingresses**. Click the **Ingresses** tab, locate the row containing the ingress that uses the certificate, and choose **More** > **Update** in the **Operation** column. If multiple ingresses are using this certificate, update the certificate for all of these ingresses. To check which ingresses are using a certificate, use the **secretName** parameter in **spec.tls** of the ingress YAML files.

   Run the following kubectl command to obtain the ingresses using a certificate (replace *<secret_name>* with the actual value):

   ```
   kubectl get ingress --all-namespaces -o jsonpath='{range .items[*]}{"namespace:"}
   {.metadata.namespace}{" name:"}{.metadata.name}{" tls:"}{.spec.tls[*]}{"\n"}{end}' | grep
   <secret_name>
   ```

2. 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.
3. On the **ConfigMaps and Secrets** page, delete the target secret. Before the deletion, back up data.

----End

2 Workloads

2.1 Creating a Workload

2.1.1 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 **Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **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 **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 **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 **Configuring the Container Lifecycle**.
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **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 **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 **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 **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 **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 **Configuring 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 **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 the parameters, see **Table 2-1**.

Table 2-1 Deployment YAML parameters

Parameter	Description	Mandatory
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**.	Yes
kind	Type of a created object.	Yes

Parameter	Description	Mandatory
metadata	Metadata of a resource object.	Yes
name	Name of the Deployment.	Yes
spec	Detailed description of the Deployment.	Yes
replicas	Number of pods.	Yes
selector	Determines container pods that can be managed by the Deployment.	Yes
strategy	Upgrade mode. Possible values: ● RollingUpdate ● ReplaceUpdate By default, rolling update is used.	No
template	Detailed description of a created container pod.	Yes
metadata	Metadata.	Yes
labels	**metadata.labels**: Container labels.	No
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.	Yes
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.	No

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 **Service**.

----End

Documentation

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

2.1.2 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 **Headless Service**.

Prerequisites

- A cluster is available. For details about how to create a cluster, see **Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **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 **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 **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 **Configuring the Container Lifecycle**.
- (Optional) **Health Check**: Set the liveness probe, ready probe, and startup probe as required. For details, see **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 **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 **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 **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 **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 **Configuring 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 **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 **Service**.

----End

2.1.3 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 **Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **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 **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 **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 **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 **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 **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 **Configuring Labels and Annotations**.
- **Job Settings**
  - **Parallel Pods**: Maximum number of pods that can run in parallel during job execution. The value cannot be greater than the total number of pods in the job.

  - **Timeout (s)**: Once a job reaches this time, the job status becomes failed and all pods in this job will be deleted. If you leave this parameter blank, the job will never time out.
  - Completion Mode
    - **Non-indexed**: A job is considered complete when all the pods are successfully executed. Each pod completion is homologous to each other.
    - **Indexed**: Each pod gets an associated completion index from 0 to the number of pods minus 1. The job is considered complete when every pod allocated with an index is successfully executed. For an indexed job, pods are named in the format of $(job-name)-$(index).
  - **Suspend Job**: By default, a job is executed immediately after being created. The job's execution will be suspended if you enable this option, and resumed after you disable it.

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 2-2 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 2,000th 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
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803
4825342117067982148086513282306647093844609550582231725359408128481117450284102701938521
1055596446229489549303819644288109756659334461284756482337867831652712019091456485669234
6034861045432664821339360726024914127372458700660631558817488152092096282925409171536436
7892590360011330530548820466521384146951941511609433057270365759591953092186117381932611
7931051185480744623799627495673518857527248912279381830119491298336733624406566430860213
9494639522473719070217986094370277053921717629317675238467481846766940513200056812714526
3560827785771342757789609173637178721468440901224953430146549585371050792279689258923542
0199561121290219608640344181598136297747713099605187072113499999983729780499510597317328
1609631859502445945534690830264252230825334468503526193118817101000313783875288658753320
8381420617177669147303598253490428755468731159562863882353787593751957781857780532171226
8066130019278766111959092164201989380952572010654858632788659361533818279682303019520353
0185296899577362259941389124972177528347913151557485724245415069595082953311686172785588
9075098381754637464939319255060400927701671139009848824012858361603563707660104710181942
9555961989467678374494482553797747268471040475346462080466842590694912933136770289891521
0475216205696602405803815019351125338243003558764024749647326391419927260426992279678235
4781636009341721641219924586315030286182974555706749838505494588586926995690927210797509
3029553211653449872027559602364806654991198818347977535663698074265425278625518184175746
7289097777279380008164706001614524919217321721477235014144197356854816136115735255213347
5741849468438523323907394143334547762416862518983569485562099219222184272550254256887671
7904946016534668049886272327917860857843838279679766814541009538837863609506800642251252
0511739298489608412848862694560424196528502221066118630674427862203919494504712371378696
0956364371917287467764657573962413890865832645995813390478027590 1
```

----End

Related Operations

After a one-off job is created, you can perform operations described in **Table 2-3**.

Table 2-3 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.

2.1.4 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 **Buying a CCE Autopilot Cluster**.
- VPC endpoints for accessing SWR and OBS have been configured. For details, see **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 **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 **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 **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 **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 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 **Configuring 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 2-4**.

Table 2-4 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.

2.2 Configuring a Workload

2.2.1 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.

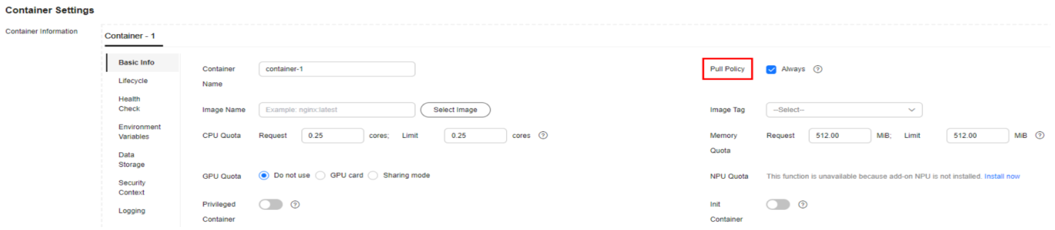
By default, **imagePullPolicy** is set to **IfNotPresent**, indicating that if the image exists on the node, the existing image is used. If the image does not exist on the node, the image is pulled from the image repository.

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 2-1 Configuring an update policy



Container Name: container-1
Image Name: Example: nginx:latest
CPU Quota: Request 0.25 cores; Limit 0.25 cores
GPU Quota: ● Do not use ○ GPU card ○ Sharing mode
Privileged Container
Pull Policy: ☑ Always
Image Tag: --Select--
Memory Quota: Request 512.00 MiB; Limit 512.00 MiB
NPU Quota: This function is unavailable because add-on NPU is not installed. Install now
Init Container

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.

2.2.2 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**.

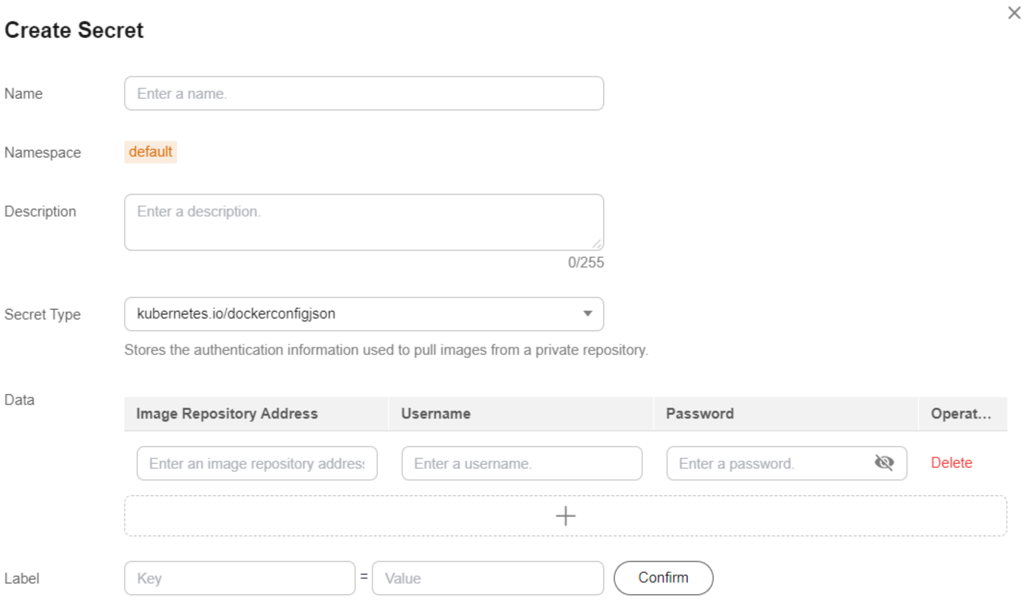
Using the Console

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 **Creating a Secret**.

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

Figure 2-2 Creating a secret



Create Secret

Name: Enter a name.

Namespace: default

Description: Enter a description.
0/255

Secret Type: kubernetes.io/dockerconfigjson

Stores the authentication information used to pull images from a private repository.

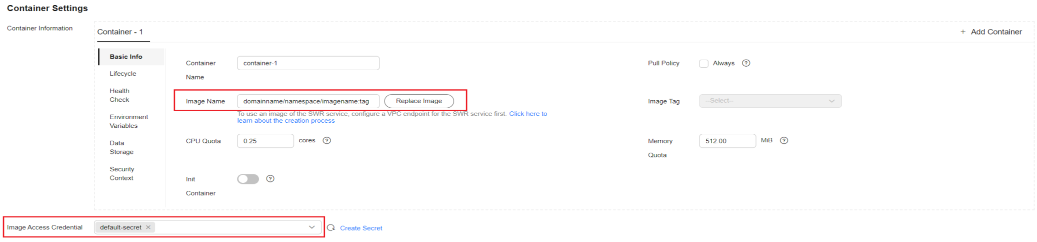
Data

Image Repository Address	Username	Password	Operat...
Enter an image repository addres:	Enter a username.	Enter a password.	Delete

Label: Key = Value

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 2-3 Private image path



Container Settings

Container Information

Container - 1

Container Name: container-1

Image Name: domainname/namespace/imagename:tag

To use an image of the SWR service, configure a VPC endpoint for the SWR service first. Click here to learn about the creation process

CPU Quota: 0.25 cores

Init Container

Pull Policy: □ Always

Image Tag: --Select--

Memory Quota: 512.00 MiB

Image Access Credential: default-secret

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

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Use kubectl to create a secret of the kubernetes.io/dockerconfigjson.

```
kubectl create secret docker-registry myregistrykey  -n default --docker-server=DOCKER_REGISTRY_SERVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL
```

In the preceding command, *myregistrykey* indicates the key name, *default* indicates the namespace where the key is located, and other parameters are as follows:

- **DOCKER_REGISTRY_SERVER**: address of a third-party image repository, for example, **www.3rdregistry.com** or **10.10.10.10:443**
- **DOCKER_USER**: account used for logging in to a third-party image repository
- **DOCKER_PASSWORD**: password used for logging in to a third-party image repository
- **DOCKER_EMAIL**: email of a third-party image repository

Step 3 Use a third-party image to create a workload.

A kubernetes.io/dockerconfigjson secret is used for authentication when you obtain a private image. The following is an example of using the myregistrykey for authentication.

```
apiVersion: v1
kind: Pod
metadata:
  name: foo
  namespace: default
spec:
  containers:
    - name: foo
      image: www.3rdregistry.com/janedoe/awesomeapp:v1
  imagePullSecrets:
    - name: myregistrykey              #Use the created secret.
```

----End

2.2.3 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.

Table 2-5 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 2-6 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 2-7 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 2-8 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 **uninstall.sh** 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
```

2.2.4 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.

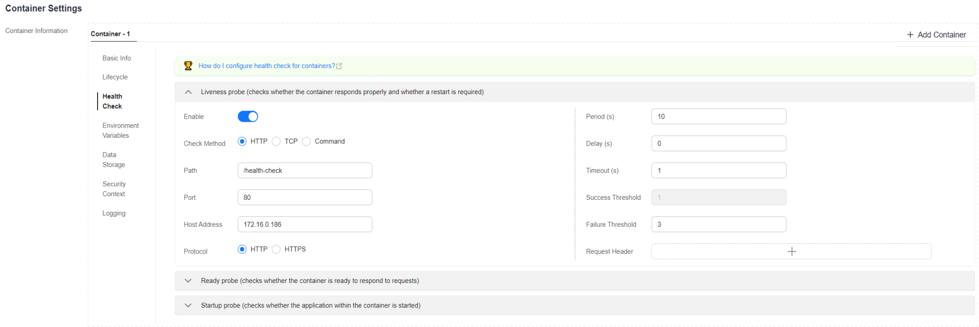
Check Method

- **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 2-4 HTTP request-based check



Liveness probe (checks whether the container responds properly and whether a restart is required)

Enable: ●

Check Method: ● HTTP ○ TCP ○ Command

Path: /health-check

Port: 80

Host Address: 172.16.0.186

Protocol: ● HTTP ○ HTTPS

Period (s): 10

Delay (s): 0

Timeout (s): 1

Success Threshold: 1

Failure Threshold: 3

Request Header: +

Ready probe (checks whether the container is ready to respond to requests)

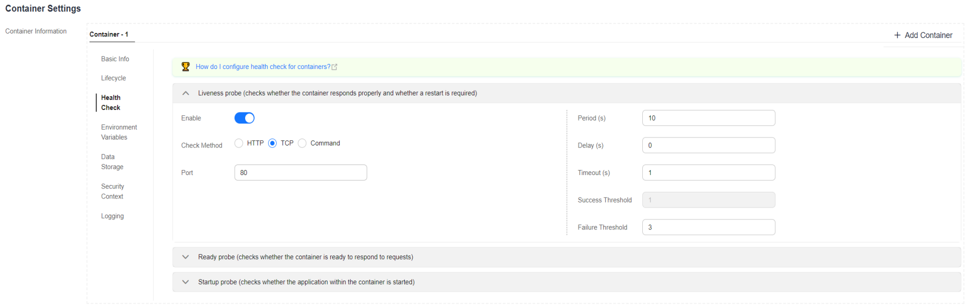
Startup probe (checks whether the application within the container is started)

- **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 2-5 TCP port-based check



Liveness probe (checks whether the container responds properly and whether a restart is required)

Enable: ●

Check Method: ○ HTTP ● TCP ○ Command

Port: 80

Period (s): 10

Delay (s): 0

Timeout (s): 1

Success Threshold: 1

Failure Threshold: 3

Ready probe (checks whether the container is ready to respond to requests)

Startup probe (checks whether the application within the container is started)

- **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.

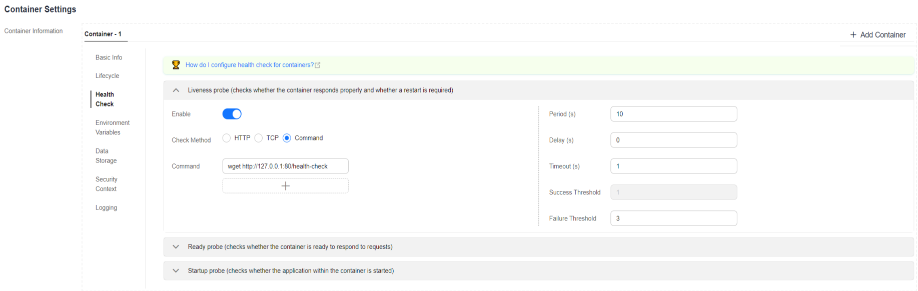
  The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

  - 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 2-6 Check using CLI



Container Settings
Container Information
Container - 1
+ Add Container
Basic Info
Lifecycle
Health Check
Environment Variables
Data Storage
Security Context
Logging
How do I configure health check for containers?
Ready probe (checks whether the container is ready to respond to requests)
Startup probe (checks whether the application within the container is started)

Liveness probe (checks whether the container responds properly and whether a restart is required)

Enable: ●

Check Method: ○ HTTP ○ TCP ● Command

Command: wget http://127.0.0.1:80/health-check

Period (s): 10

Delay (s): 0

Timeout (s): 1

Success Threshold: 1

Failure Threshold: 3

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.

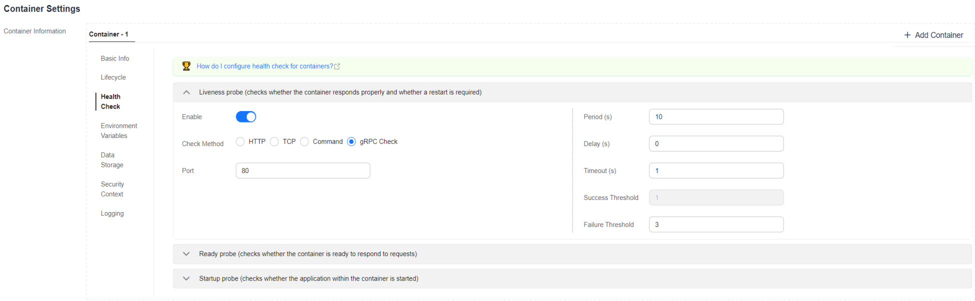
- **gRPC Check**

  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 2-7 gRPC check



Container Settings
Container Information
Container - 1
+ Add Container
Basic Info
Lifecycle
Health Check
Environment Variables
Data Storage
Security Context
Logging
How do I configure health check for containers?
Ready probe (checks whether the container is ready to respond to requests)
Startup probe (checks whether the application within the container is started)

Liveness probe (checks whether the container responds properly and whether a restart is required)

Enable: ●

Check Method: ○ HTTP ○ TCP ○ Command ● gRPC Check

Port: 80

Period (s): 10

Delay (s): 0

Timeout (s): 1

Success Threshold: 1

Failure Threshold: 3

Common Parameters

Table 2-9 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
```

2.2.5 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**: Import all key values in a ConfigMap as environment variables.
- **Added from ConfigMap key**: Import the value of a key in a ConfigMap as the value of an environment variable. As shown in Figure 2-8, if you import **configmap_value** of **configmap_key** in **configmap-example** as the value of environment variable **key1**, an environment variable named **key1** whose value is **configmap_value** is available in the container.
- **Added from secret**: Import all key values in a secret as environment variables.

- **Added from secret key**: Import the value of a key in a secret as the value of an environment variable. As shown in **Figure 2-8**, 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 2-8**, 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 2-8**, if you import the CPU limit of container-1 as the value of environment variable **key4**, an environment variable named **key4** whose value is the CPU limit of container-1 is available in the container.

Adding Environment Variables

Step 1 Log in to the CCE console.

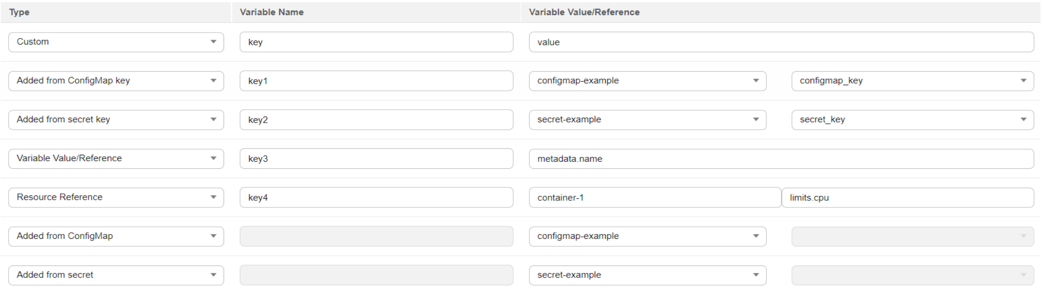
Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane, and click the **Create Workload** in the upper right corner.

Step 3 When creating a workload, modify the container information in **Container Settings** and click the **Environment Variables** tab.

Step 4 Configure environment variables.

- To add environment variables one by one, click **Adding a Variable** and configure its parameters.
- To add environment variables in batches, click **Editing Custom Variables in Batches**. Then, in the displayed dialog box, enter environment variables in the format of "Variable name=Variable or variable reference".

Figure 2-8 Configuring environment variables



Type	Variable Name	Variable Value/Reference	
Custom	key	value	
Added from ConfigMap key	key1	configmap-example	configmap_key
Added from secret key	key2	secret-example	secret_key
Variable Value/Reference	key3	metadata.name	
Resource Reference	key4	container-1	limits.cpu
Added from ConfigMap		configmap-example	
Added from secret		secret-example	

----End

YAML Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          env:
            - name: key                     # Custom
              value: value
            - name: key1                    # Added from ConfigMap key
              valueFrom:
                configMapKeyRef:
                  name: configmap-example
                  key: configmap_key
            - name: key2                    # Added from secret key
              valueFrom:
                secretKeyRef:
                  name: secret-example
                  key: secret_key
            - name: key3                    # Variable reference, which uses the field defined by a pod as the value
of the environment variable.
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: key4                    # Resource reference, which uses the field defined by a container as the
value of the environment variable.
              valueFrom:
                resourceFieldRef:
                  containerName: container1
                  resource: limits.cpu
                  divisor: 1
          envFrom:
            - configMapRef:                 # Added from ConfigMap
                name: configmap-example
            - secretRef:                    # Added from secret
                name: secret-example
      imagePullSecrets:
        - name: default-secret
```

Viewing Environment Variables

If the contents of **configmap-example** and **secret-example** are as follows:

```
$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVl              # c2VjcmV0X3ZhbHVl is the value of secret_value in Base64
mode.
```

```
kind: Secret
...
```

The environment variables in the pod are as follows:

```
$ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
env-example-695b759569-lx9jp   1/1     Running   0          17m

$ kubectl exec env-example-695b759569-lx9jp  -- printenv
/ # env
key=value                          # Custom environment variable
key1=configmap_value                  # Added from ConfigMap key
key2=secret_value                    # Added from secret key
key3=env-example-695b759569-lx9jp     # metadata.name defined by the pod
key4=1                              # limits.cpu defined by container1. The value is rounded up, in unit of cores.
configmap_key=configmap_value         # Added from ConfigMap. The key value in the original ConfigMap key is directly imported.
secret_key=secret_value              # Added from key. The key value in the original secret is directly imported.
```

2.2.6 Configuring APM

Scenario

Application Performance Management (APM) allows you to monitor Java workloads through tracing and topology. You can install APM probes to locate and analyze problems for Java workloads.

You can configure Java workload monitoring when and after a workload is created.

Prerequisites

- You have enabled the APM service. If you have not enabled the APM service, go to the APM console and enable it as prompted.
- You have created a VPC endpoint for APM. If you have not created an endpoint, create one by following the instructions on the performance management configuration page.

Precautions

- If a VPC endpoint for APM is deleted, data will fail to be uploaded to APM. If you create this endpoint again, the metrics will not be uploaded to APM automatically. In this case, you can restart the application.
- Deleting a workload that uses APM will not delete the endpoint. To delete it, go to the network console.

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 select **APM 2.0**. After the probe is enabled, APM can locate and analyze problems for Java programs.

NOTE

1. The APM 2.0 probe will be initialized in an automatically-created init container named **init-javaagent**. This init container will be allocated 0.25 CPU cores and 250 MiB of memory.
2. Adding an APM probe will add the following environment variables to all service containers: **PAAS_MONITORING_GROUP**, **JAVA_TOOL_OPTIONS**, **PAAS_CLUSTER_ID**, and **APM_ACCESS_ADDRESS**.
3. Adding an APM 2.0 probe will mount a local storage volume named **paas-apm2** to all service containers.

Step 4 Set probe-related parameters.

- To access APM in a CCE Autopilot cluster, you need to create a VPC endpoint for APM to connect to the VPC. For details about the VPC endpoint price, see **VPC Endpoint Pricing**.
- **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.
- **APM Environment**: (Optional) Enter an environment name.
- **APM Service**: Select an existing APM application.
- **Sub-service**: (Optional) Enter a sub-service of the APM application.
- **Access key**: The system automatically obtains the key information of APM. You can go to the APM console to view the key details.

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

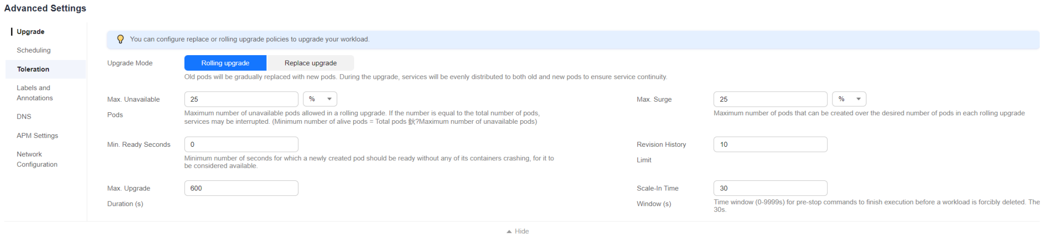
2.2.7 Configuring the Workload Upgrade Policy

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 2-9 Configuring a workload upgrade policy



Advanced Settings

You can configure replace or rolling upgrade policies to upgrade your workload.

Upgrade Mode: Rolling upgrade

Old pods will be gradually replaced with new pods. During the upgrade, services will be evenly distributed to both old and new pods to ensure service continuity.

Max. Unavailable Pods: 25 %

Maximum number of unavailable pods allowed in a rolling upgrade. If the number is equal to the total number of pods, services may be interrupted. (Minimum number of alive pods = Total pods 鈥?Maximum number of unavailable pods)

Max. Surge: 25 %

Maximum number of pods that can be created over the desired number of pods in each rolling upgrade

Min. Ready Seconds: 0

Minimum number of seconds for which a newly created pod should be ready without any of its containers crashing, for it to be considered available.

Revision History Limit: 10

Max. Upgrade Duration (s): 600

Scale-In Time Window (s): 30

Time window (0-9999s) for pre-stop commands to finish execution before a workload is forcibly deleted. The 30s.

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.
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 only supported 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).	-

Parameter	Description	Constraint
Revision History Limit (revisionHistory-Limit)	Specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of **kubectl get rs**. The configuration of each Deployment revision is stored in its ReplicaSets. Therefore, once the old ReplicaSet is deleted, you lose the ability to roll back to that revision of Deployment. By default, 10 old ReplicaSets will be kept, but the ideal value depends on the frequency and stability of the new Deployments.	-
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**.	-
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.	-

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**.

2.2.8 Configuring Labels and Annotations

Pod Annotations

CCE allows you to add annotations using YAML to enable some advanced pod functions. The following table describes the annotations you can add.

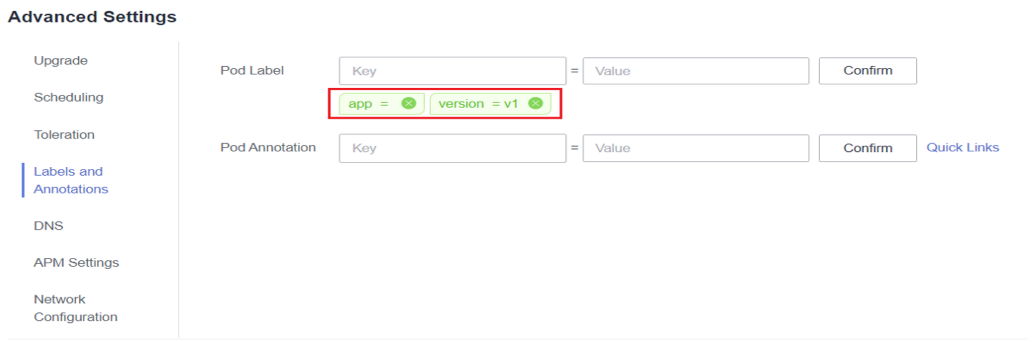
Table 2-10 Pod annotations

Function and Reference	Annotation	Example Value	Description
Configuring QoS for a Pod	kubernetes.io/ingress-bandwidth	100M	Indicates the ingress bandwidth of a pod. This annotation controls the rate at which a pod receives data to ensure that the pod can process external requests.
	kubernetes.io/egress-bandwidth	100M	Indicates the egress bandwidth of a pod. This annotation controls the rate at which a pod sends data to external systems. This affects the efficiency of communication between the pod and external services or users.
Setting AZ Affinity	node.cce.io/node-az-list	cn-east-3a,cn-east-3b	Lists the AZs for pod affinity. For a CCE Autopilot cluster, you can use workload annotations to implement AZ affinity and schedule pods to specified AZs.

Pod Labels

You can use pod labels to organize, select, and manage resources for pods to make it easy to use and maintain resources.

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.



Advanced Settings

Pod Label: app = , version = v1

Example YAML:

```
...
spec:
```

```
selector:
  matchLabels:
    app: nginx
    version: v1
template:
  metadata:
    labels:
      app: nginx
      version: v1
  spec:
    ...
```

2.2.9 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.

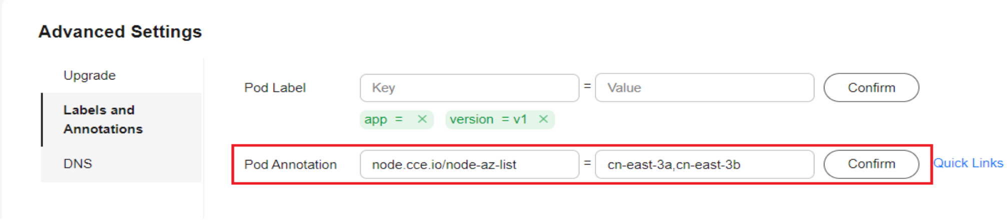
Step 2 Click the cluster name to go to the cluster console, choose **Workloads** in the navigation pane on the left, and click the **Create Workload** in the upper right corner.

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 AZ names in different regions, see **Regions and Endpoints**.

Figure 2-10 Setting AZ affinity



Pod Label: app = ×, version = v1 ×

Pod Annotation: node.cce.io/node-az-list = cn-east-3a,cn-east-3b

Step 4 Configure other workload parameters and click **Create Workload**.

----End

2.3 Logging In to a Container

Scenario

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

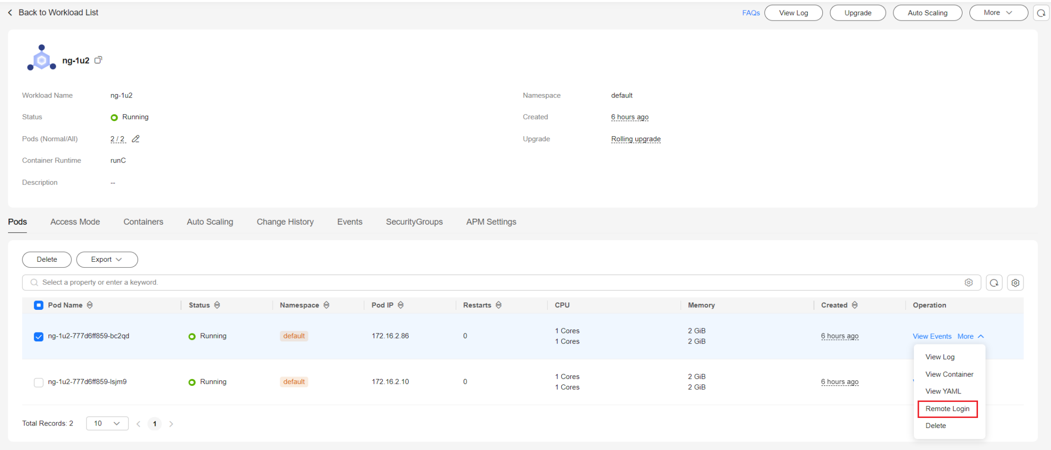
Logging In to a Container Using CloudShell

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**. 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 2-11 Accessing a container



Workload Name ng-1u2

Status Running

Pods (Normal/All) 2 / 2

Container Runtime runC

Description --

Namespace default

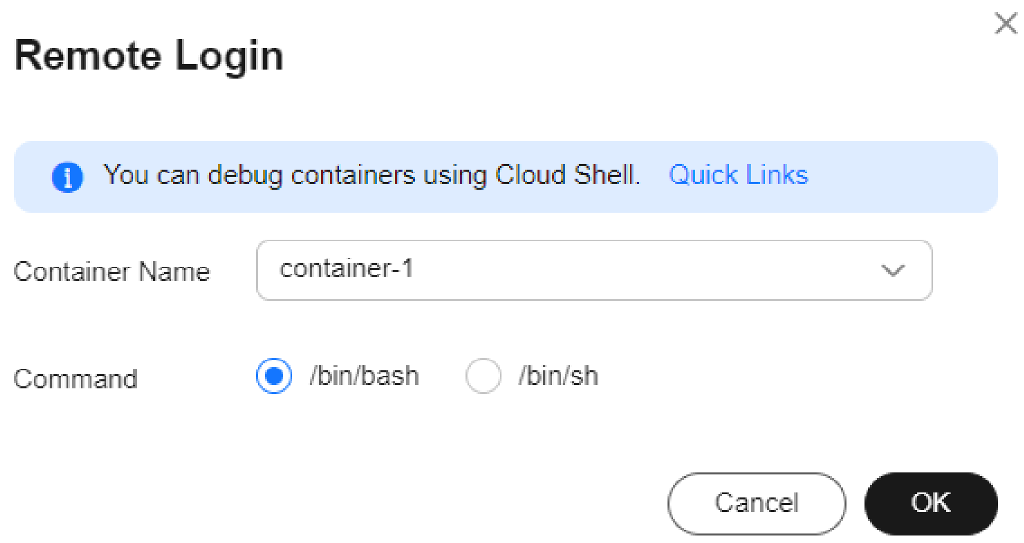
Created 6 hours ago

Upgrade Rolling upgrade

Pod Name	Status	Namespace	Pod IP	Restarts	CPU	Memory	Created	Operation
☑ ng-1u2-777d6ff859-bc2qd	Running	default	172.16.2.86	0	1 Cores 1 Cores	2 GiB 2 GiB	6 hours ago	View Events More
□ ng-1u2-777d6ff859-lsjm9	Running	default	172.16.2.10	0	1 Cores 1 Cores	2 GiB 2 GiB	6 hours ago	

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

Figure 2-12 Selecting a container and login command



Remote Login

You can debug containers using Cloud Shell. Quick Links

Container Name: container-1

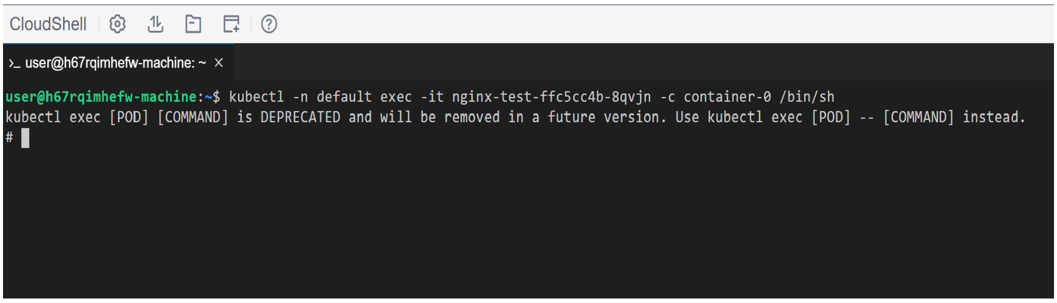
Command: ● /bin/bash ○ /bin/sh

Step 5 You will be automatically redirected to CloudShell. Then, kubectl is initialized 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 2-13 CloudShell page



```
user@h67rqimhefw-machine:~$ kubectl -n default exec -it nginx-test-ffc5cc4b-8qvjn -c container-0 /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
# 
```

----End

Logging In to a Container Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Run the following command to view the created pod:

```
kubectl get pod
```

The example output is as follows:

```
NAME                      READY   STATUS    RESTARTS     AGE
nginx-59d89cb66f-mhljr        1/1     Running   0            11m
```

Step 3 Query the container name in the pod.

```
kubectl get po nginx-59d89cb66f-mhljr -o jsonpath='{range .spec.containers[*]}{.name}{end}{"\n"}'
```

The example output is as follows:

```
container-1
```

Step 4 Run the following command to log in to the **container-1** container in the **nginx-59d89cb66f-mhljr** pod:

```
kubectl exec -it nginx-59d89cb66f-mhljr -c container-1 -- /bin/sh
```

Step 5 To exit the container, run the **exit** command.

----End

2.4 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 2-11 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 and StatefulSets 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, and jobs. This section uses a Deployment as an example to describe how to view logs.

NOTICE

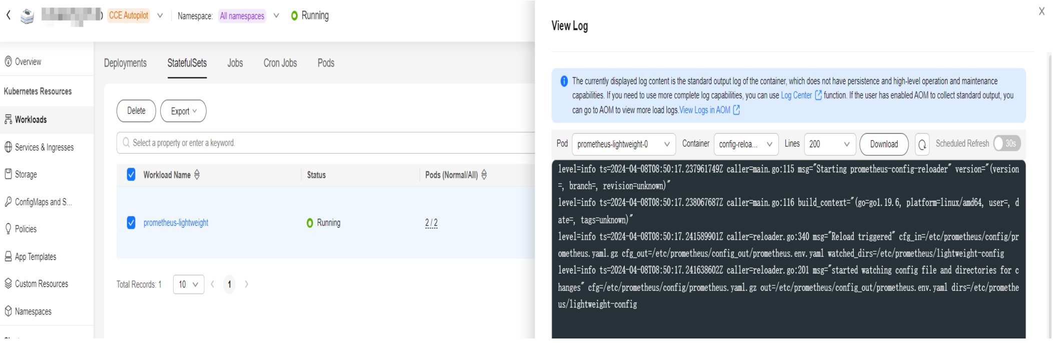
Before viewing logs, ensure that the time of the browser is the same as that on the backend server.

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 2-14 Viewing logs of a workload



The currently displayed log content is the standard output log of the container, which does not have persistence and high-level operation and maintenance capabilities. If you need to use more complete log capabilities, you can use Log Center function. If the user has enabled AOM to collect standard output, you can go to AOM to view more load logs.View Logs in AOM

```
level=info ts=2024-04-08T08:50:17.237961749Z caller=main.go:115 msg="Starting prometheus-config-reloader" version="(version
=, branch=, revision=unknown)"
level=info ts=2024-04-08T08:50:17.238067687Z caller=main.go:116 build_context="(go=go1.19.6, platform=linux/amd64, user=, d
ate=, tags=unknown)"
level=info ts=2024-04-08T08:50:17.241589901Z caller=reloader.go:340 msg="Reload triggered" cfg_in=/etc/prometheus/config/pr
ometheus.yaml.gz cfg_out=/etc/prometheus/config_out/prometheus.env.yaml watched_dirs=/etc/prometheus/lightweight-config
level=info ts=2024-04-08T08:50:17.241638602Z caller=reloader.go:201 msg="started watching config file and directories for c
hanges" cfg=/etc/prometheus/config/prometheus.yaml.gz out=/etc/prometheus/config_out/prometheus.env.yaml dirs=/etc/promethe
us/lightweight-config
```

----End

Upgrading a Workload

You quickly upgrade Deployments and StatefulSets on the 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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab. Locate the target workload and click **Upgrade** in the **Operation** column.

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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab. Locate the target workload and choose **More** > **Edit YAML** in the **Operation** column. 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)

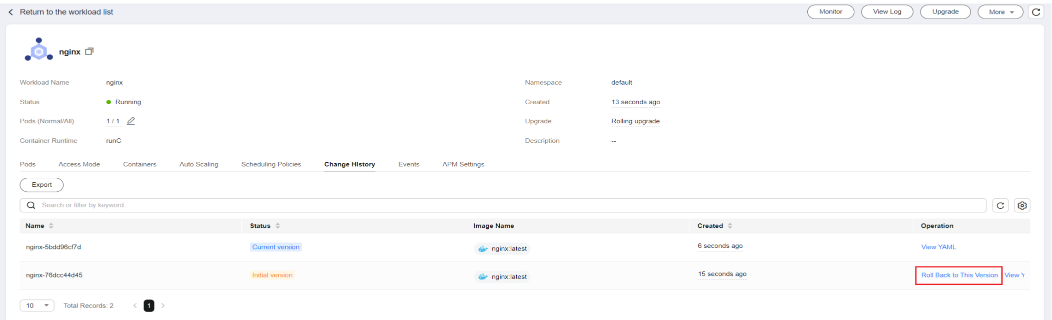
CCE records the release history of all Deployments. You can roll back a Deployment to a specified version.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab. Locate the target workload and choose **More > Roll Back** in the **Operation** column.

Step 3 Switch to the **Change History** tab. Locate the target workload and click **Roll Back to This Version**. Manually confirm the YAML file and click **OK**.

Figure 2-15 Rolling back a workload version



Workload Name nginx
Status Running
Pods (Normal/All) 1 / 1
Container Runtime runC
Namespace default
Created 13 seconds ago
Upgrade Rolling upgrade
Description --

Name	Status	Image Name	Created	Operation
nginx-5bdd96cf7d	Current version	nginx:latest	6 seconds ago	View YAML
nginx-76dcc44d45	Initial version	nginx:latest	15 seconds ago	Roll Back to This Version View Y

10 Total Records: 2 1

----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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab. Locate the target workload and choose **More** > **Redeploy** in the **Operation** column.

Step 3 In the dialog box that is displayed, click **Yes**.

----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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab. Locate the target workload and choose **More** > **Disable/Enable Upgrade** in the **Operation** column.

Step 3 In the dialog box that is displayed, click **Yes**.

----End

Managing Labels

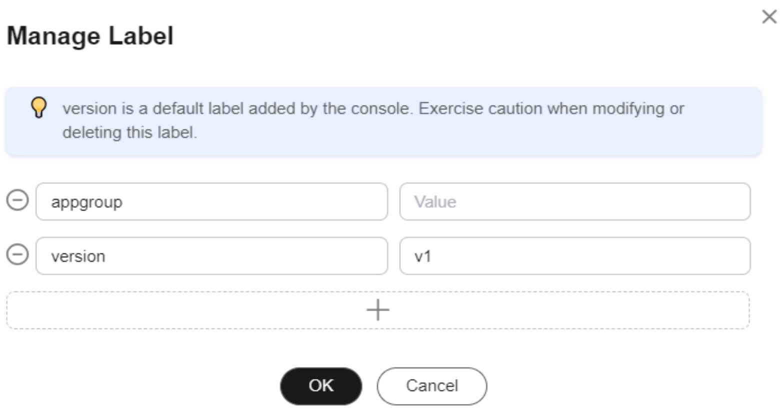
Labels are key-value pairs and can be attached to workloads. You can manage and select workloads by labels. You can add labels to multiple workloads or a specified workload.

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Click the **Deployments** tab and choose **More** > **Manage Label** in the **Operation** column of the workload.

Step 3 Click **Add**, enter a key and a value, and click **OK**.

Figure 2-16 Managing labels



Manage Label

version is a default label added by the console. Exercise caution when modifying or deleting this label.

appgroup	Value
version	v1

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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 Locate the workload you will delete and choose **More** > **Delete** in the **Operation** column.

Read the prompts carefully. A workload cannot be recovered after it is deleted.

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 and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Workloads**.

Step 2 On the **Deployments** tab, click the target workload. In the **Pods** tab, 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

2.5 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.

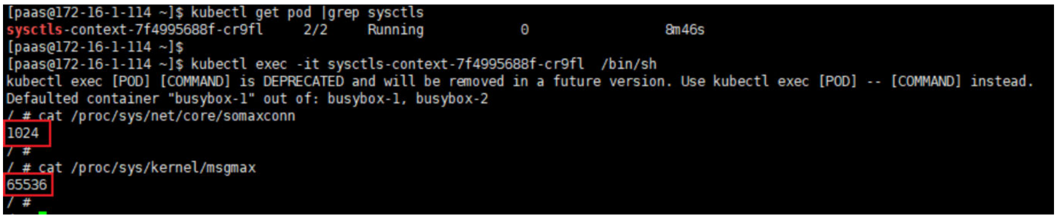
Non-secure sysctl parameters are unstable. Using non-secure sysctl parameters may cause some serious problems, such as container errors. You need to take care of the risks.

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
```



```
[paas@172-16-1-114 ~]$ kubectl get pod |grep sysctls
sysctls-context-7f4995688f-cr9fl     2/2     Running            0                 8m46s
[paas@172-16-1-114 ~]$
[paas@172-16-1-114 ~]$ kubectl exec -it sysctls-context-7f4995688f-cr9fl  /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
Defaulted container "busybox-1" out of: busybox-1, busybox-2
/ # cat /proc/sys/net/core/somaxconn
1024
/ #
/ # cat /proc/sys/kernel/msgmax
65536
/ #
```

2.6 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.

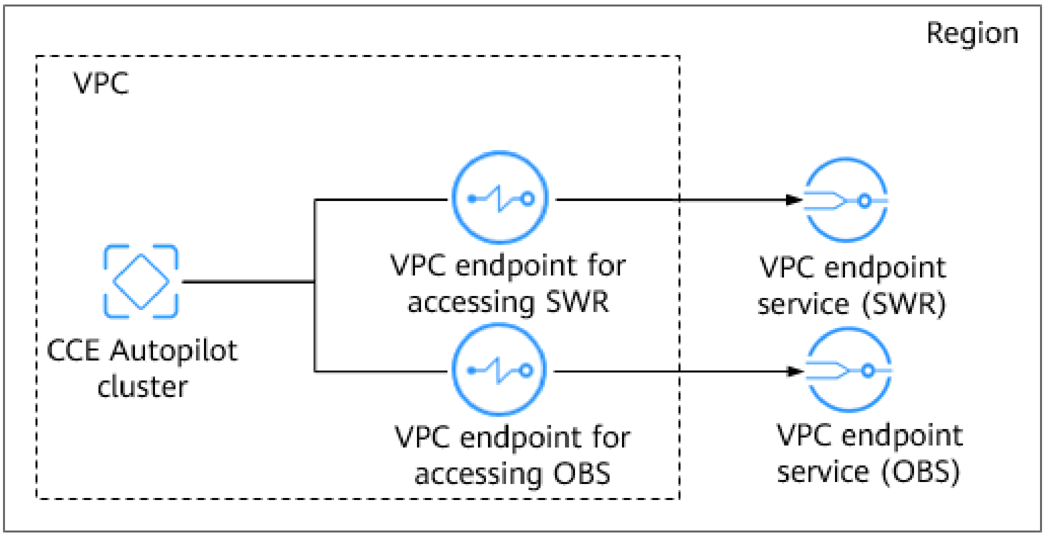
----End

2.7 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 2-17 A CCE Autopilot cluster accessing SWR and OBS


Region
VPC
CCE Autopilot cluster
VPC endpoint for accessing SWR
VPC endpoint for accessing OBS
VPC endpoint service (SWR)
VPC endpoint service (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 2-12 Parameters for creating a VPC endpoint

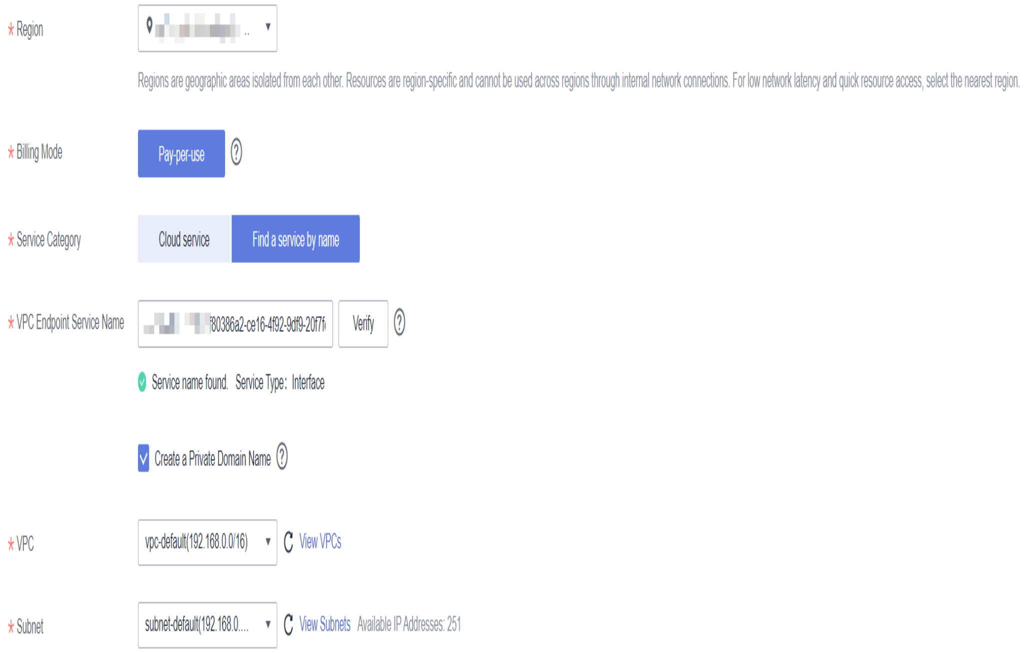
Parameter	Description
Region	Select the region where the VPC endpoint is located. The VPC endpoint must be in the same region as the cluster.
Billing Mode	Select **Pay-per-use**.
Service Category	Select **Find a service by name**.
VPC Endpoint Service Name	Enter a service name based on **Table 2-13** and click **Verify**.
VPC	Select the VPC where the cluster is located.

Parameter	Description
Subnet	Select a subnet.
IPv4 Address	By default, **Automatically assign IPv4 address** is selected. You can also select **Manually specify an IP address** as required.

Table 2-13 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
CN North-Beijing4	com.myhuaweicloud.cn-north-4.swr
AP-Bangkok	ap-southeast-2.SWR.ac7067e1-f8d1-4f5c-abe1-0f78960e5d4c
AP-Singapore	com.myhuaweicloud.ap-southeast-3.swr

Figure 2-18 Creating a VPC endpoint for SWR



* Region: [redacted] ..

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Billing Mode: Pay-per-use

* Service Category: Cloud service | Find a service by name

* VPC Endpoint Service Name: [redacted] f80386a2-ce16-4f92-9df9-20f7f Verify

Service name found. Service Type: Interface

☑ Create a Private Domain Name

* VPC: vpc-default(192.168.0.0/16) View VPCs

* Subnet: subnet-default(192.168.0.... View Subnets Available IP Addresses: 251

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 2-14 Parameters for creating a VPC endpoint

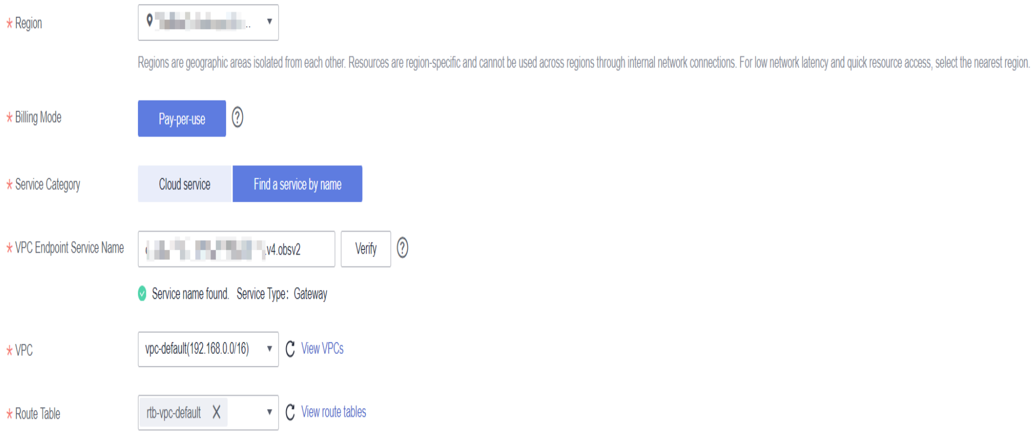
Parameter	Description
Region	Select the region where the VPC endpoint is located. The VPC endpoint must be in the same region as the cluster.
Billing Mode	Select **Pay-per-use**.
Service Category	Select **Find a service by name**.
VPC Endpoint Service Name	Enter a service name based on **Table 2-15** and click **Verify**.
VPC	Select the VPC where the cluster is located.
Route Table	Select a route table.

Table 2-15 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-south-1.com.myhuaweicloud.v4.obsv2
CN East-Shanghai1	cn-east-3.com.myhuaweicloud.v4.global.obsv2
CN North-Beijing4	cn-north-4.com.myhuaweicloud.v4.obsv2

Region	Name
AP-Bangkok	ap-southeast-2.myhuaweicloud.v4.obsv2
AP-Singapore	ap-southeast-3.com.myhuaweicloud.v4.obsv2

Figure 2-19 Creating a VPC endpoint for OBS



* Region: [redacted]

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Billing Mode: Pay-per-use

* Service Category: Cloud service | Find a service by name

* VPC Endpoint Service Name: [redacted]v4.obsv2 Verify

Service name found. Service Type: Gateway

* VPC: vpc-default(192.168.0.0/16) View VPCs

* Route Table: rtb-vpc-default View route tables

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

3 Services and Ingresses

3.1 Service

3.1.1 ClusterIP

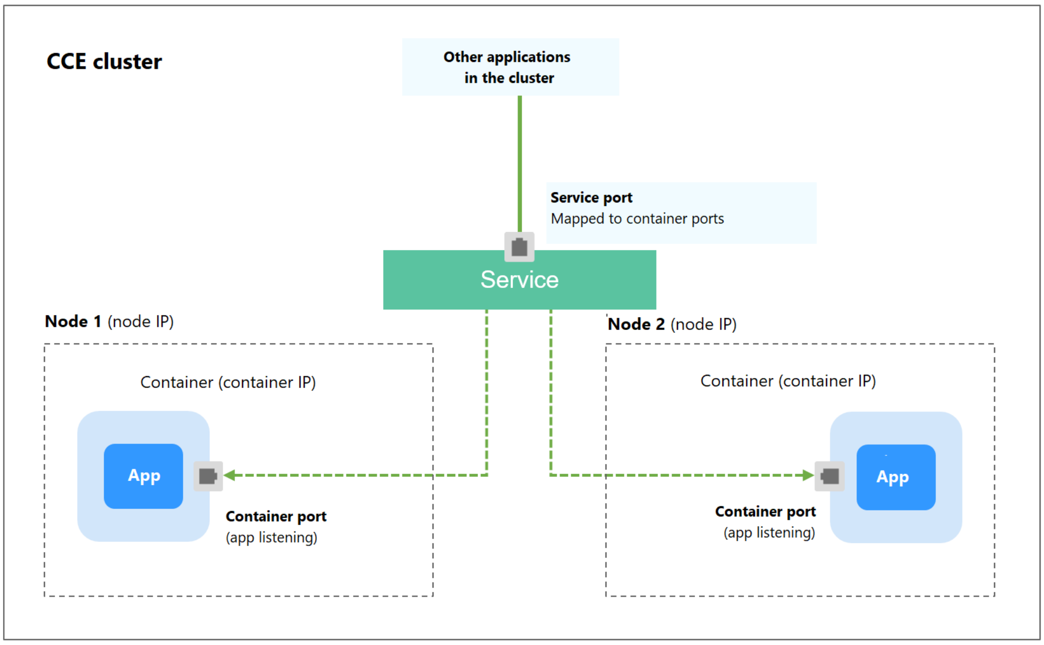
Scenario

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 3-1 shows the access channel and mappings between the access port and container ports.

Figure 3-1 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 on the left, 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**: Select the namespace that the workload belongs to.
- **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**.
- **Ports**
  - **Protocol**: Select the protocol used by the Service.
  - **Service Port**: Specify the port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: Specify the port on which the workload listens. For example, Nginx uses port 80 by default.

Step 4 Click **OK**.

----End

Adding a Service Using kubectl

You can run kubectl commands to add a Service. An Nginx workload is used 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 **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 the Service over *IP address:Port* or the domain name.

The domain name suffix can be omitted. In the same namespace, you can 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

3.1.2 LoadBalancer

3.1.2.1 Creating a LoadBalancer Service

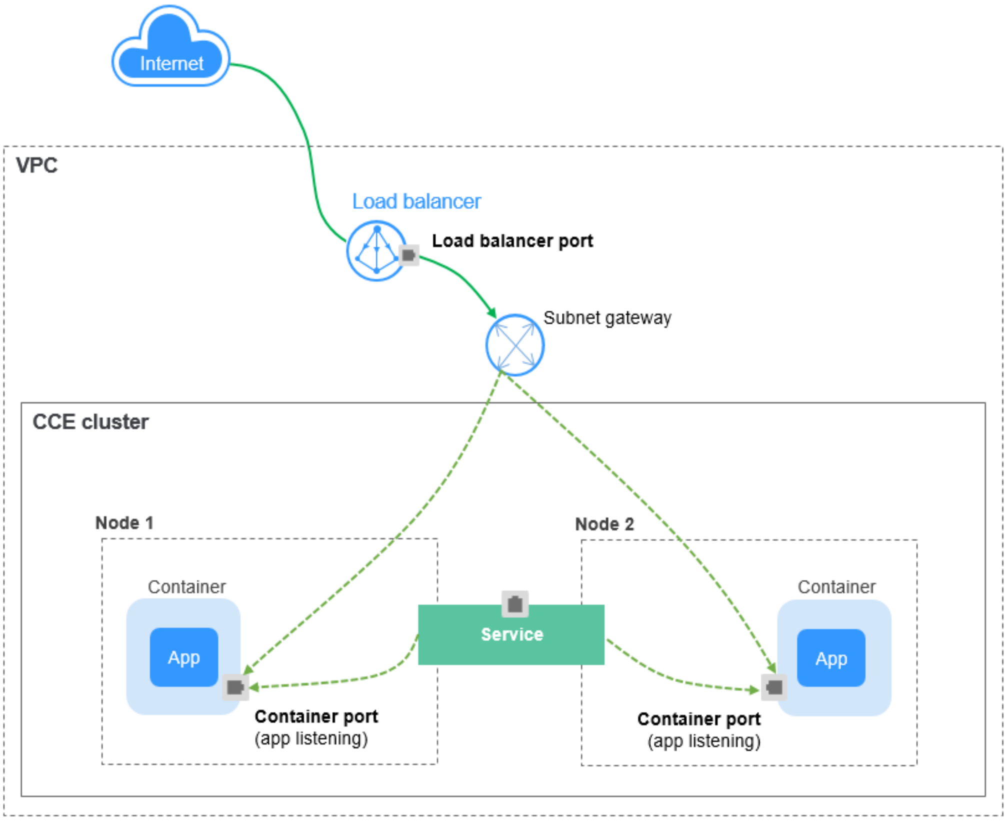
Scenario

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 3-2 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**: Select the namespace that the workload belongs to.
- **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**. A dedicated load balancer supports **Network (TCP/UDP)**, **Application (HTTP/HTTPS)**, or **Network (TCP/UDP) & Application (HTTP/HTTPS)**.

  Select either **Use existing** or **Auto create**. For more information, see **Table 3-1**.

Table 3-1 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.

Option	Description
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 used to allocate IP addresses to load balancers to receive traffic from clients. – **Backend Subnet**: This parameter is used to allocate IP addresses for load balancers to routing traffic to pods. – **Network/Application-oriented Specifications** ▪ **Elastic**: applies to fluctuating traffic, billed based on the total traffic. ▪ **Fixed**: applies to stable traffic, billed based on specifications. – **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 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. This is supported by clusters of v1.27.5-r0, v1.28.3-r0, and later versions.

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

Parameter	Description
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**: Select the protocol used by the Service.
  - **Service Port**: Specify the port used by the Service. The port number ranges from 1 to 65535.
  - **Container Port**: Specify the port on which the workload listens. For example, Nginx uses port 80 by default.
  - **Frontend Protocol**: Set the protocol of the load balancer listener for establishing a connection with the clients. For a dedicated load balancer, to use HTTP/HTTPS, the type of the load balancer must be **Application (HTTP/HTTPS)**.
  - **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 3-2**.

  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 **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

```
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 on 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 3-3**.

For more annotations and examples related to advanced features, see **Using Annotations to Configure Load Balancing**.

Table 3-3 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 load balancer. On the **Summary** tab, find and copy the ID.
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.
kubernetes.io/elb.session-affinity-mode	No	String	In source IP address-based sticky sessions, 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 3-4** 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 option 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 3-5** Object	ELB health check configuration items.

Table 3-4 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 3-5 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

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

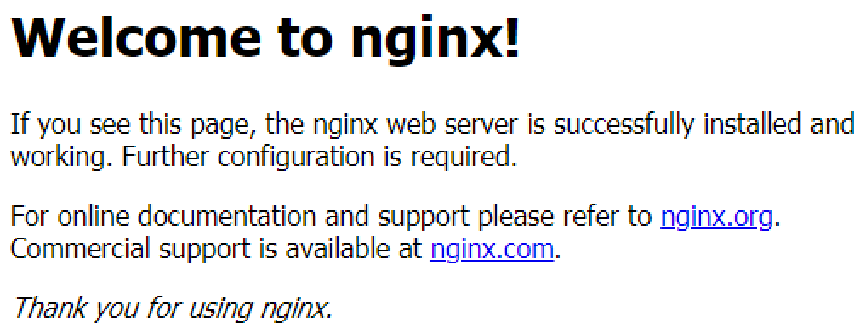
If information similar to the following is displayed, the access type has been set, and the workload is accessible.

```
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.

Nginx is accessible.

Figure 3-3 Accessing Nginx through the LoadBalancer Service



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

----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 **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

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",
      "vip_subnet_cidr_id": "*****",
      "vip_address": "**.**.**.**",
      "elb_virsubnet_ids": ["*****"],
      "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"
    }'
    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
```

This example uses annotations to implement some advanced features of load balancing, such as sticky sessions and health check. For details, see **Table 3-6**.

For more annotations and examples related to advanced features, see **Using Annotations to Configure Load Balancing**.

Table 3-6 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 for the Service. **NOTE** Automatic creation of a public network load balancer: You need to purchase an EIP for the load balancer to allow access over the public network. The load balancer can also be accessed over a private network. Automatic creation of a private network load balancer: You do not need to purchase an EIP for the load balancer. The load balancer can only be accessed over a private network. **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","available_zone": ["cn-east-3a"],"l4_flavor_name": "L4_flavor.elb.s1.small"}' - If a private network load balancer will be automatically created, set this parameter to the following value: '{"type":"inner","available_zone": ["cn-east-3a"],"l4_flavor_name": "L4_flavor.elb.s1.small"}'
kubernetes.io/elb.subnet-id	-	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	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	In source IP address-based sticky sessions, 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 3-4** Object	Sticky session timeout.

Parameter	Mandatory	Type	Description
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 option 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 3-5** Object	ELB health check configuration items.
kubernetes.io/elb.tags	No	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.

Table 3-7 elb.autocreate data structure

Parameter	Mandatory	Type	Description
name	No	String	Name of the automatically created load balancer. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed. Default: **cce-lb+service.UID**
type	No	String	Network type of the load balancer. ● **public**: public network load balancer ● **inner**: private network load balancer Default: **inner**
bandwidth_name	Yes for public network load balancers	String	Bandwidth name. The default value is **cce-bandwidth-******. The value can contain 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.

Parameter	Mandatory	Type	Description
bandwidth_chargemode	No	String	Bandwidthbilling option. ● **bandwidth**: billed by bandwidth ● **traffic**: billed by traffic Default: **bandwidth**
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2,000 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 1,000 Mbit/s. ● The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1,000 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.

Parameter	Mandatory	Type	Description
vip_address	No	String	Specifies the private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the CIDR block of the load balancer. If this parameter is not specified, an IP address will be automatically assigned from the CIDR block of the load balancer.
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.

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) as the load balancer CIDR block.
			This parameter is available only for dedicated load balancers.
			Example:
			`"elb_virsubnet_ids": [`
			`  "14567f27-8ae4-42b8-ae47-9f847a4690dd"`
			`]`

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

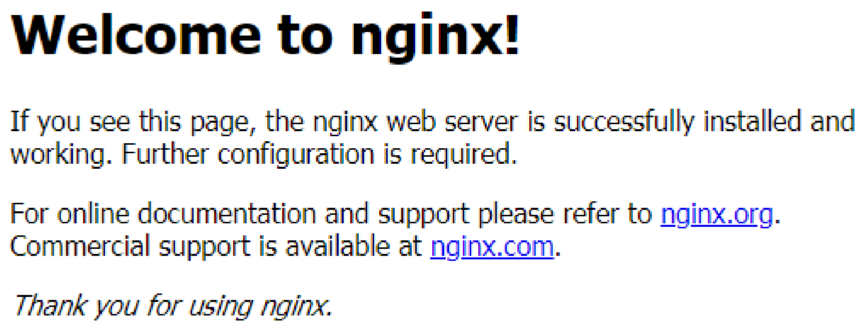
If information similar to the following is displayed, the access type has been set, and the workload is accessible.

```
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 3-4 Accessing Nginx through the LoadBalancer Service



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

----End

3.1.2.2 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**.

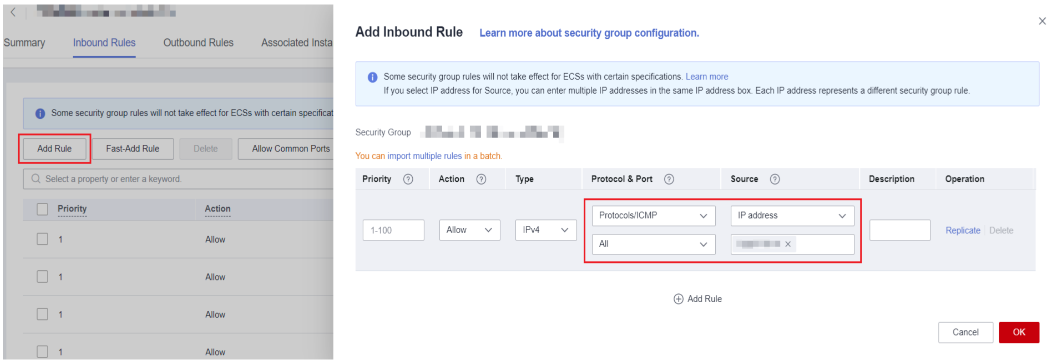
Step 2 In the security group list, locate the security group associated with the elastic network interfaces. The default security group is named *{Cluster name}-cce-eni-{Random ID}*.

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 3-5**.

- **Protocol & Port**: Select all ICMP ports.
- **Source**: Enter the backend subnet of the load balancer.

Figure 3-5 Adding a security group rule



[redacted]

Some security group rules will not take effect for ECSs with certain specificat

Priority	Action
1	Allow
1	Allow
1	Allow
1	Allow

Some security group rules will not take effect for ECSs with certain specifications. Learn more
If you select IP address for Source, you can enter multiple IP addresses in the same IP address box. Each IP address represents a different security group rule.

Security Group [redacted]

You can import multiple rules in a batch.

Priority	Action	Type	Protocol & Port	Source	Description	Operation
1-100	Allow	IPv4	Protocols/ICMP All	IP address [redacted] ×		Replicate | Delete

Step 4 Click **OK**.

----End

3.1.3 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
```

3.2 Ingresses

3.2.1 LoadBalancer Ingresses

3.2.1.1 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 **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.

  Select either **Use existing** or **Auto create**. For more information, see **Table 3-8**.

Table 3-8 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.

Option	Description
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 used to allocate IP addresses to load balancers to receive traffic from clients. – **Backend Subnet**: This parameter is used to allocate IP addresses for load balancers to routing traffic to pods. – **Network/Application-oriented Specifications** ■ **Elastic**: applies to fluctuating traffic, billed based on the total traffic. ■ **Fixed**: applies to stable traffic, billed based on specifications. – **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 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. This is supported by clusters of v1.27.5-r0, v1.28.3-r0, and later versions.

- **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 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.
  - **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 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 **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 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

  - SNI is available only when **HTTPS** is selected.
  - 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.
- **Advanced Options**

Description	Description
Transfer Listener Port Number	If this option is enabled, the listening port on the load balancer can be transferred to backend servers through the HTTP header of the packet.
Transfer Port Number in the Request	If this option is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet.
Rewrite X-Forwarded-Host	If this option is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the request and transferred to backend servers.

Description	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.
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.

- **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, gRPC, and HTTP are supported. ○ **Check Path**: This parameter is only available for HTTP and GRPC 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**.

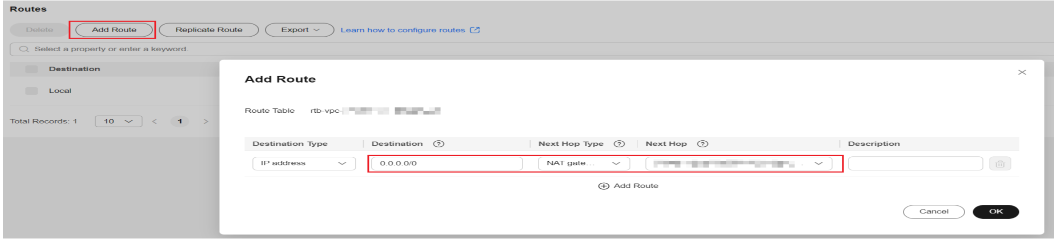
Step 4 Click **OK**. After the ingress is created, it is displayed in the ingress list.

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 3-6 ELB routing configuration



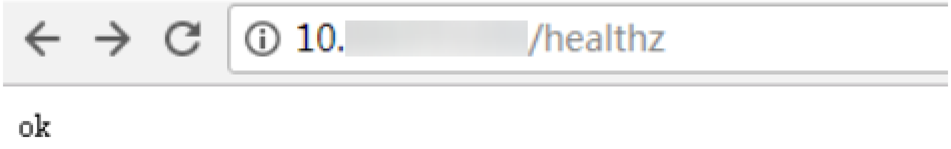
Route Table rtb-vpc-[redacted]

Destination Type	Destination	Next Hop Type	Next Hop	Description
IP address	0.0.0.0/0	NAT gate...	[redacted]	

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 3-7 Accessing the /healthz interface of defaultbackend



10.[redacted]/healthz

```
ok
```

----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?**

3.2.1.2 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 **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 **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 3-9 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	No	String	ID of the subnet where the cluster is located. The value can contain 1 to 100 characters. 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	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 3-10**. **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	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.
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
path	Yes	String	User-defined route path. All external access requests must match **host** and **path**. **NOTE** The access path added here must exist in the backend application. Otherwise, the forwarding fails. For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.

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. 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. ● **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 3-10 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	Bandwidthbilling option. ● **bandwidth**: billed by bandwidth ● **traffic**: billed by traffic Default: **bandwidth**
bandwidth_size	Yes for public network load balancers	Integer	Bandwidth size. The default value is 1 to 2,000 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 1,000 Mbit/s. ● The minimum increment is 500 Mbit/s if the allowed bandwidth exceeds 1,000 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
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.
vip_address	No	String	Specifies the private IP address of the load balancer. Only IPv4 addresses are supported. The IP address must be in the CIDR block of the load balancer. If this parameter is not specified, an IP address will be automatically assigned from the CIDR block of the load balancer.
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) as the load balancer CIDR block. This parameter is available only for dedicated load balancers. Example: `"elb_virsubnet_ids": [ "14567f27-8ae4-42b8-ae47-9f847a4690dd" ]`

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 have 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 3-11 Key parameters

Parameter	Mandatory	Type	Description
kubernetes.io/elb.id	Yes	String	Load balancer ID. 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 load balancer. On the **Summary** tab, 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.

3.2.2 Nginx Ingresses

3.2.2.1 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 **Creating a Workload**.
- A ClusterIP Service has been configured for the workload. For details about how to configure the Service, see **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 after the NGINX Ingress Controller add-on is installed in the cluster.
  - **External Protocol**: The options are **HTTP** and **HTTPS**. The default number of the listening port reserved when Nginx Ingress Controller is installed is 80 for HTTP and 443 for HTTPS. To use HTTPS, configure a server certificate.
  - **Certificate Source**: source of a certificate for encrypting and authenticating HTTPS data transmission.
    - If you select a TLS key, you must create a key certificate of the IngressTLS or kubernetes.io/tls type beforehand. For details, see **Creating a Secret**.
    - If you select the default certificate, Nginx Ingress Controller will use its default certificate for encryption and authentication.
  - **SNI**: SNI an extended protocol of TLS. SNI allows multiple TLS-compliant domain names 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 the 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

3.2.2.2 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 **Creating a Workload**.
- A ClusterIP Service has been configured for the workload. For details about how to configure the Service, see **ClusterIP**.

Procedure

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a YAML file named **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 3-12 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. 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. ● **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

3.3 Pod Network Settings

3.3.1 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 3-14**.

- 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 3-13 Annotations of an EIP with a dedicated 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-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 2,000 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 types vary by 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
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 3-14 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 by region. 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 3-13**. ● Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	-

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.

3.3.2 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 **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 3-15**.

- 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 3-15 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 3-16 Annotations of an EIP with a dedicated bandwidth

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/pod-with-eip	Yes	false	Whether to allocate an EIP with a pod and bind the EIP to the pod	**false** or **true**

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-size	No	5	Bandwidth, in Mbit/s	The value range varies depending on the region and bandwidth billing mode. For details, see the purchase page on the EIP console. For example, in the AP-Singapore region, if an EIP is billed by bandwidth, the bandwidth ranges from 1 Mbit/s to 2000 Mbit/s; if an EIP is billed by traffic, the bandwidth ranges from 1 Mbit/s to 300 Mbit/s.
yangtse.io/eip-network-type	No	5_bgp	EIP type	The type varies depending on the region. For details, see the purchase page on the EIP console. For example, the following options are available in in the AP-Singapore region: ● **5_bgp**: dynamic BGP
yangtse.io/eip-charge-mode	No	None	Billed by traffic or bandwidth **You are advised to configure this parameter.** If this parameter is left blank, no billing mode is specified. In this case, the default value of the EIP API in the region is used.	● **bandwidth**: billed by bandwidth ● **traffic**: billed by traffic

Annotation	Mandatory	Default Value	Description	Value Range
yangtse.io/eip-bandwidth-name	No	Pod name	Bandwidth name	● Enter 1 to 64 characters. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. ● Minimum length: 1 character ● Maximum length: 64 characters

Table 3-17 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 by region. 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 3-13**. - Only the **yangtse.io/eip-network-type** field can be specified concurrently, and this field is optional.	-

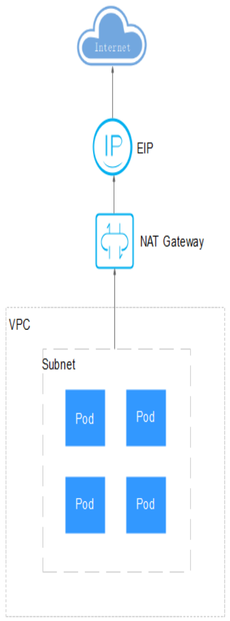
Deleting a Static EIP

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.

3.4 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 3-8** 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 3-8 SNAT



Procedure

To enable a container pod to access the Internet, perform the following steps:

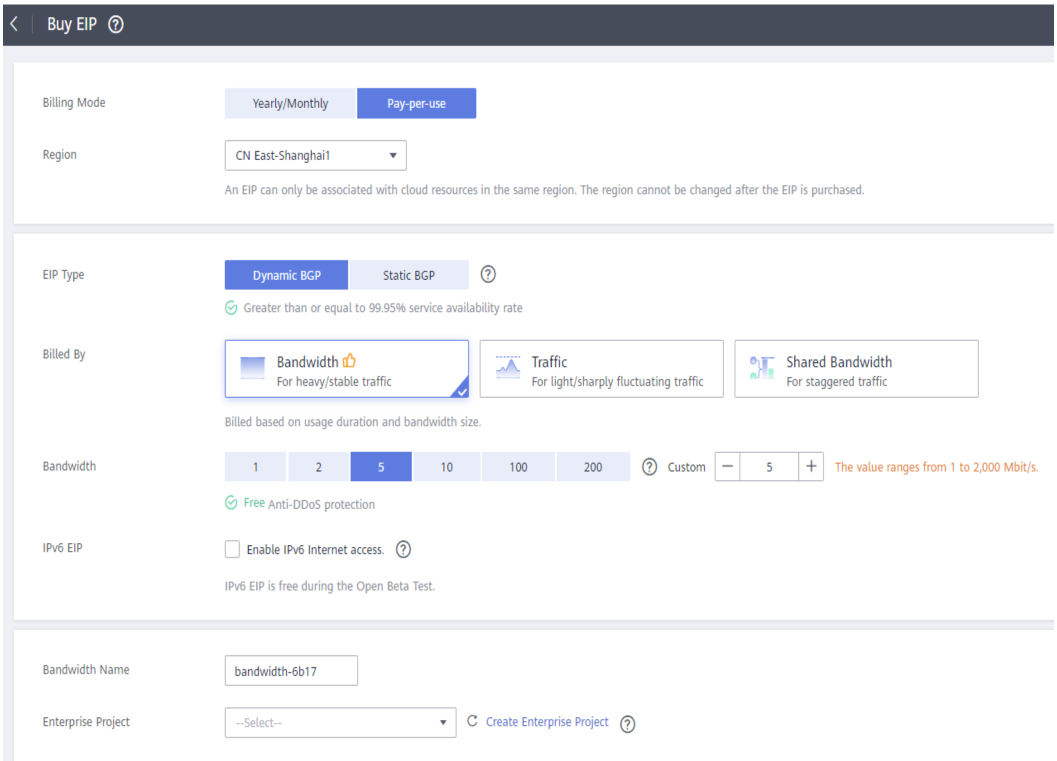
Step 1 Assign an EIP.

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 the parameters as required.

NOTE

Set **Region** to the region where container pods are located.

Figure 3-9 Buying an elastic IP address



< Buy EIP ⓘ

Billing Mode: Yearly/Monthly | **Pay-per-use**

Region: CN East-Shanghai1 ▾

An EIP can only be associated with cloud resources in the same region. The region cannot be changed after the EIP is purchased.

EIP Type: **Dynamic BGP** | Static BGP ⓘ

✓ Greater than or equal to 99.95% service availability rate

Billed By: **Bandwidth** (For heavy/stable traffic) | Traffic (For light/sharply fluctuating traffic) | Shared Bandwidth (For staggered traffic)

Billed based on usage duration and bandwidth size.

Bandwidth: 1 | 2 | **5** | 10 | 100 | 200 ⓘ Custom − 5 + The value ranges from 1 to 2,000 Mbit/s.

✓ Free Anti-DDoS protection

IPv6 EIP: □ Enable IPv6 Internet access. ⓘ

IPv6 EIP is free during the Open Beta Test.

Bandwidth Name: bandwidth-6b17

Enterprise Project: --Select-- ▾ C Create Enterprise Project ⓘ

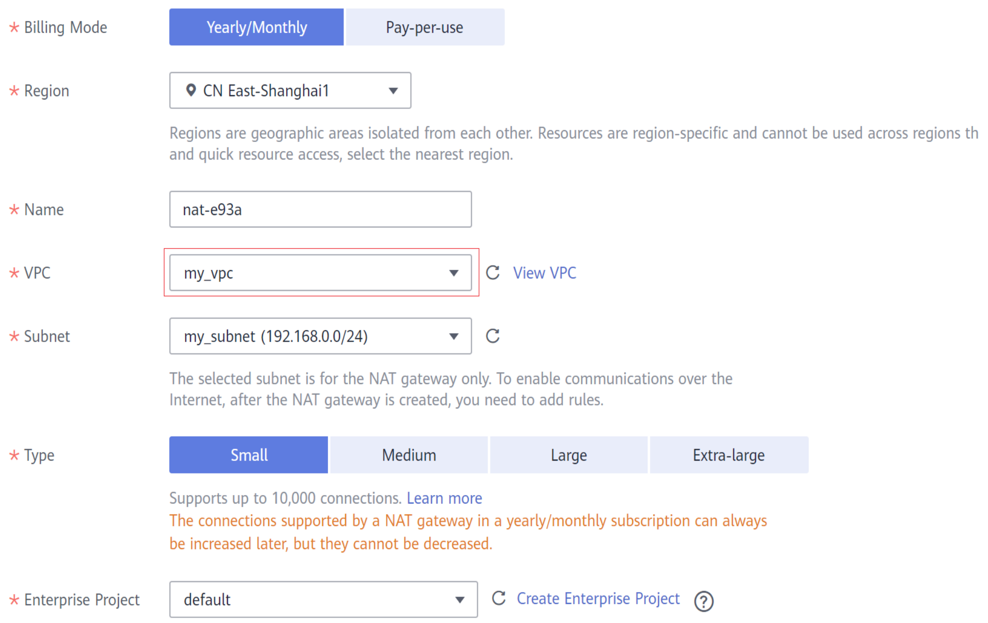
Step 2 Create a NAT gateway. For details, see **Buying a Public NAT Gateway**.

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 the parameters as required.

NOTE

Select the same VPC.

Figure 3-10 Buying a NAT gateway



* Billing Mode: Yearly/Monthly | Pay-per-use

* Region: CN East-Shanghai1

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions th
and quick resource access, select the nearest region.

* Name: nat-e93a

* VPC: my_vpc View VPC

* Subnet: my_subnet (192.168.0.0/24)

The selected subnet is for the NAT gateway only. To enable communications over the Internet, after the NAT gateway is created, you need to add rules.

* Type: Small | Medium | Large | Extra-large

Supports up to 10,000 connections. Learn more

The connections supported by a NAT gateway in a yearly/monthly subscription can always be increased later, but they cannot be decreased.

* Enterprise Project: default Create Enterprise Project

Step 3 Configure an SNAT rule and bind the EIP to the subnet. For details, see **Adding an SNAT Rule**.

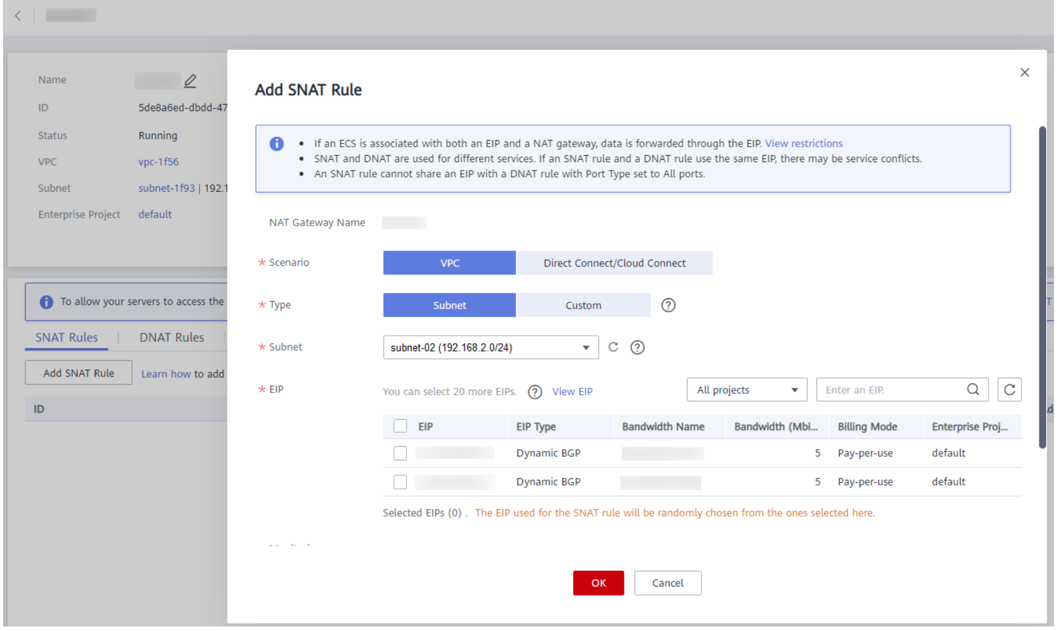
1. Log in to the management console.
2. Click 📍 in the upper left corner of the management console and select a region and a project.
3. Click ☰ at the upper left corner and choose **Networking** > **NAT Gateway** in the expanded list.
4. On the page displayed, click the name of the NAT gateway for which you want to add the SNAT rule.
5. On the **SNAT Rules** tab, click **Add SNAT Rule**.
6. Configure the 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 3-11 Adding an SNAT rule



Name

ID 5de8a6ed-dbdd-47

Status Running

VPC vpc-1f56

Subnet subnet-1f93 | 192.1

Enterprise Project default

To allow your servers to access the

SNAT Rules | DNAT Rules

Add SNAT Rule Learn how to add

ID

Add SNAT Rule

- If an ECS is associated with both an EIP and a NAT gateway, data is forwarded through the EIP. View restrictions
- SNAT and DNAT are used for different services. If an SNAT rule and a DNAT rule use the same EIP, there may be service conflicts.
- An SNAT rule cannot share an EIP with a DNAT rule with Port Type set to All ports.

NAT Gateway Name

* Scenario: VPC | Direct Connect/Cloud Connect

* Type: Subnet | Custom

* Subnet: subnet-02 (192.168.2.0/24)

* EIP: You can select 20 more EIPs. View EIP

All projects

Enter an EIP.

□ EIP	EIP Type	Bandwidth Name	Bandwidth (Mbi...	Billing Mode	Enterprise Proj...
□	Dynamic BGP		5	Pay-per-use	default
□	Dynamic BGP		5	Pay-per-use	default

Selected EIPs (0) . The EIP used for the SNAT rule will be randomly chosen from the ones selected here.

OK Cancel

After the SNAT rule is configured, workloads can access public networks from the container. Public networks can be pinged from the container.

----End

4 Storage

4.1 Scalable File Service

4.1.1 Overview

Introduction

CCE Autopilot allows you to mount a volume that is created from a Scalable File Service (SFS) file system to a container for persistent data storage. 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 analysis. For services with a massive number 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 read/write I/O latency).
- **Use cases**: Deployments and 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 Autopilot supports SFS 3.0 file systems. For details about file system types, see **File System Type**.

Table 4-1 Performance

Parameter	SFS 3.0
Maximum bandwidth	1.25 TB/s
Maximum IOPS	Million
Latency	10 ms
Maximum capacity	EB

Application Scenarios

SFS supports the following mounting modes based on application scenarios:

- **Using an Existing File System Through a Static PV**: static provisioning. You use an existing file system to create a PV and then mount the PV to the workload through a PVC. This mode applies to scenarios where the underlying storage is available or billed on a yearly/monthly basis.
- **Using an SFS File System Through a Dynamic PV**: dynamic provisioning. You do not need to create volumes in advance. Instead, you need to specify a StorageClass during PVC creation. A file system and a PV will be automatically created. This mode applies to scenarios where no underlying storage is available.

Billing

- For SFS volumes **automatically created** using StorageClass, the default billing mode is **Pay-per-use**. You are billed based on the storage capacity and duration. For SFS pricing details, see **Billing**.
- If you want to be billed on a yearly/monthly basis, **use existing SFS volumes**.

4.1.2 Using an Existing 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 and implement data persistence and sharing in workloads.

Prerequisites

- If you want to create a cluster using commands, use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.
- You have created a file system that is in the same VPC as the cluster.

Constraints

- Multiple PVs can use the same SFS or SFS Turbo file system, but there are some constraints.
  - Do not mount all PVCs/PVs that use the same SFS or SFS Turbo file system to a pod. This will result in a pod startup failure because not all

PVCs can be mounted to the pod due to the same **volumeHandle** values of these PVs.

- The **persistentVolumeReclaimPolicy** parameter in the PVs should be set to **Retain**. If any other value is used, 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. In the navigation pane on the left, choose **Storage**. Then click the **PVCs** tab. In the upper right corner, click **Create PVC**. In the displayed dialog box, configure the parameters.

Parameter	Description
PVC Type	In this example, select **SFS**.
PVC Name	Enter the PVC name, which must be unique in the same namespace.
Creation Method	– If underlying storage is available, create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV is available. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using an SFS File System Through a Dynamic PV**. In this example, select **Create new** to create 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[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 is supported.
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**.

Parameter	Description
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 avoid cascading deletion of underlying volumes.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring SFS Volume Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   In the navigation pane on the left, choose **Storage**. View the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create a workload.

1. In the navigation pane on the left, choose **Workloads**. Then click the **Deployments** tab.
2. In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

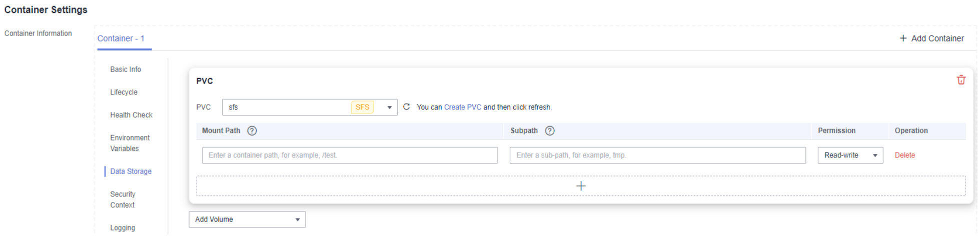
   Mount and use storage volumes, as shown in **Table 4-2**. For details about other parameters, see **Workloads**.

Table 4-2 Mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the container path that the volume will be mounted to. 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. If there are such files, they will be replaced, which will lead to a container startup 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.

Parameter	Description
Subpath	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. 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.
Permission	– **Read-only**: You can only read the data in the mounted volume. – **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 file system.



Container Settings

Container Information

Container - 1

PVC

PVC: sfs SFS

You can Create PVC and then click refresh.

Mount Path	Subpath	Permission	Operation
Enter a container path, for example, /test.	Enter a sub-path, for example, tmp.	Read-write	Delete

3. Configure other parameters and 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-sfs.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 PV is deleted while the
underlying volume is retained.
  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> # Enter the SFS volume name if SFS 3.0 is used.
```

```
    volumeAttributes:
      everest.io/share-export-location: <your_location>  # Shared path of the SFS volume.
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
      everest.io/sfs-version: sfs3.0       # SFS 3.0 is used.
  persistentVolumeReclaimPolicy: Retain    # Reclaim policy.
  storageClassName: csi-sfs              # Storage class name. csi-sfs indicates that SFS 3.0 is used.
  mountOptions: []                       # Mount options.
```

Table 4-3 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Currently, only **retain-volume-only** is supported. This parameter is valid only when the reclaim policy is set to **Delete**. If the reclaim policy is **Delete** and the 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 3.0 file system is used, enter the file system name.
everest.io/share-export-location	Yes	Shared path of the file system. A shared path is in the following format: *{your_sfs30_name}*.sfs3.*{region}*.myhuaweicloud.com:/*{your_sfs30_name}*
mountOptions	Yes	Mount options. If this parameter is not specified, the following configurations are used by default. For details, see **Configuring SFS Volume Mount Options**. mountOptions: - vers=3 - timeo=600 - nolock - hard

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	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 file sytem 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 file system will be retained. **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** 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-sfs       # Storage class name, which must be the same as that of the PV.
  volumeName: pv-sfs    # PV name.
```

Table 4-4 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 **Step 2.1**.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-sfs.yaml
```

Step 4 Create a workload.

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    # Custom volume name
          persistentVolumeClaim:
            claimName: pvc-sfs    # Name of the created PVC.
```

2. Run the following command to create a workload the SFS volume is mounted to:

```
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 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 created file 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 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 file in the **/data** path of the new pod has 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 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 described in **Table 4-5**.

Table 4-5 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. In the upper right corner, click **Create PersistentVolume**. In the displayed dialog box, configure the 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 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**: There are two options: **Retain** and **Delete**. 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 **Configuring SFS Volume Mount Options**. 2. Click **Create**.
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 (events are retained for one hour).
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.

4.1.3 Using an SFS File System Through a Dynamic PV

This section describes how to use storage classes to dynamically create PVs and PVCs and implement data persistence and sharing in workloads.

Automatically Creating an SFS 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. In the navigation pane on the left, choose **Storage**. Then click the **PVCs** tab. In the upper right corner, click **Create PVC**. In the displayed dialog box, configure the parameters.

Parameter	Description
PVC Type	In this example, select **SFS**.
PVC Name	Enter the PVC name, which must be unique in the same 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, select either **Use existing** or **Create new**. For details about static creation, see **Using an Existing File System Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The storage class of 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**.

2. Click **Create** to create a PVC and a PV.

   In the navigation pane on the left, choose **Storage**. View the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create a workload.

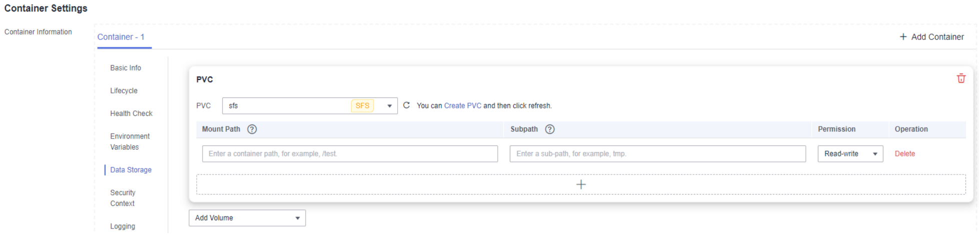
1. In the navigation pane on the left, choose **Workloads**. Then click the **Deployments** tab.
2. In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Table 4-6 describes the parameters for mounting the volume. For details about other parameters, see **Creating a Workload**.

Table 4-6 Parameters for mounting a storage volume

Parameter	Description
PVC	Select an existing SFS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the container path that the volume will be mounted to. 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. If there are such files, they will be replaced, which will lead to a container startup 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 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. 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.
Permission	– **Read-only**: You can only read the data in the mounted volume. – **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 file system.



Container Settings

PVC: sfs SFS

You can Create PVC and then click refresh.

Mount Path	Subpath	Permission	Operation
Enter a container path, for example, /test.	Enter a sub-path, for example, tmp.	Read-write	Delete

3. Configure other parameters and 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) Automatically Creating an SFS File System

Step 1 Use kubectl to connect to 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-sfs    # The StorageClass of the SFS file system
```

Table 4-7 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.
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.

Parameter	Mandatory	Description
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 a workload.

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    # Custom volume name
          persistentVolumeClaim:
            claimName: pvc-sfs-auto    # Name of the created PVC.
```

2. Run the following command to create a workload the SFS volume is mounted to:

```
kubectl apply -f web-demo.yaml
```

After the workload is created, the data in the container mount directory will be persistently stored. Verify the storage by referring to **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```
2. Run the following commands in sequence to view the files in the **/data** path of the pods:
   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```
   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:
   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```
   Expected output:
   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```
   After the deletion, the Deployment controller automatically creates a replica.
2. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```
3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:
   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```
   Expected output:
   ```
   static
   ```
   The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 **Verify data sharing.**

1. Run the following command to view the created pod:
   ```
   kubectl get pod | grep web-demo
   ```
   Expected output:
   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```
2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

```
kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
```

Check the files in the **/data** path of the pod.

```
kubectl exec web-demo-846b489584-d4d4j -- ls /data
```

Expected output:

```
share
static
```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

```
kubectl exec web-demo-846b489584-wvv5s -- ls /data
```

Expected output:

```
share
static
```

After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations described in **Table 4-8**.

Table 4-8 Related operations

Operation	Description	Procedure
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. Locate the target PVC or PV, click **View Events** in the **Operation** column to view events generated within one hour (events are retained for one hour).
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. Locate the target PVC or PV, click **View YAML** in the **Operation** column to view or download the YAML.

4.1.4 Configuring SFS Volume Mount Options

This section describes how to configure SFS 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.

SFS Volume Mount Options

CCE Autopilot presets the options described in **Table 4-9** for mounting SFS volumes.

Table 4-9 SFS volume mount options

Parameter	Value	Description
keep-original-ownership	Leave it blank.	Whether to retain the ownership of the file mount point. ● 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	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**.
sharecache/nosharecache	Leave it 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 **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 PV is deleted while the underlying
volume is retained.
  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 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-sfs              # 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 **Using an Existing 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 **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 **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

4.2 SFS Turbo

4.2.1 Overview

Introduction

CCE Autopilot 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 and StatefulSets in the ReadWriteMany mode, and jobs created for high-traffic websites, log storage, DevOps, and enterprise OA applications

SFS Turbo Performance

For details about the performance parameters of SFS Turbo, see **File System Types**.

Application Scenarios

SFS Turbo supports the following mounting modes:

- **Using an Existing SFS Turbo File System Through a Static PV**: static provisioning. You use an existing file system to create a PV and then mount the PV to the workload through a PVC.
- **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**.

4.2.2 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 **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. In the upper right corner, click **Create PVC**. In the displayed dialog box, configure the 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 is available. 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**.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring SFS Turbo Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   In the navigation pane on the left, choose **Storage**. View the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create a workload.

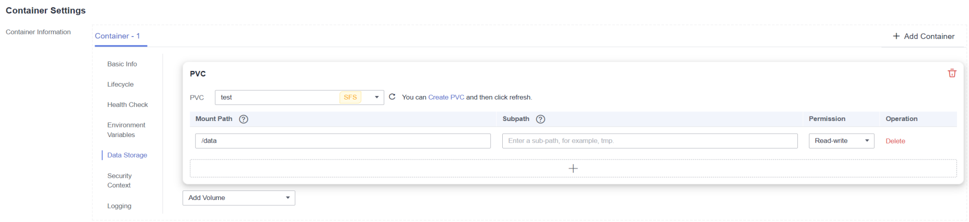
1. In the navigation pane on the left, choose **Workloads**. Then click the **Deployments** tab.
2. In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

Table 4-10 describes the parameters for mounting the volume. For details about other parameters, see **Workloads**.

Table 4-10 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 that the volume will be mounted to. 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. If there are such files, they will be replaced, which will lead to a container startup 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 volume. – **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.



PVC: test SFS

You can Create PVC and then click refresh.

Mount Path	Subpath	Permission	Operation
/data	Enter a sub-path, for example, tmp.	Read-write	Delete

3. Configure other parameters and 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 4-11 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.

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.

mountOptions	No	Mount options.
		If this parameter is not specified, the following configurations are used by default. For details, see **Configuring SFS Turbo Mount Options**.
		`mountOptions:`
		`- vers=3`
		`- timeo=600`
		`- nolock`
		`- hard`

persistentVolumeReclaimPolicy	Yes	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** 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 of the SFS Turbo volume, which must be the
```

```
same as that of the PV.
  volumeName: pv-sfsturbo    # PV name.
```

Table 4-12 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.
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 a workload.

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 that the SFS Turbo volume is mounted to:

   ```
   kubectl apply -f web-demo.yaml
   ```

   After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   Expected output:

   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```

   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```

   Expected output:

   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```

   After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```

   Expected output:

   ```
   static
   ```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 **Verify data sharing.**

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   Expected output:

   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```

   Check the files in the **/data** path of the pod.

   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```

   Expected output:

   ```
   share
   static
   ```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```

   Expected output:

   ```
   share
   static
   ```

   After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations described in **Table 4-13**.

Table 4-13 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. In the upper right corner, click **Create PersistentVolume**. In the displayed dialog box, configure the parameters. ● **Volume Type**: Select **SFS Turbo**. ● **SFS Turbo**: Click **Select SFS Turbo**. On the displayed page, 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 **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. Locate the target PVC and click **More** > **Scale-out** in the **Operation** column. 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 (events are 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.

4.2.3 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.

SFS Turbo Mount Options

CCE Autopilot presets the options described in **Table 4-14** for mounting SFS Turbo volumes.

Table 4-14 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 **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           # 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 **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

4.2.4 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/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.

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

4.3 Object Storage Service

4.3.1 Overview

Introduction

Object Storage Service (OBS) provides massive, secure, and cost-effective data storage capabilities 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 (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 VOD, gene sequencing, intelligent video surveillance, backup and archiving, and enterprise cloud boxes (web disks). You can create object storage by using the OBS console, tools, and SDKs.

OBS Specifications

OBS provides multiple storage classes to meet customers' requirements on storage performance and costs.

- Object buckets provide reliable, high-performance, secure, and budget-friendly storage for data. They have no restrictions on the quantity of files or storage capacity.
  - Standard: features low latency and high throughput. It is therefore good for storing frequently (multiple times per month) accessed files or small files (less than 1 MB). Its application scenarios include big data analytics, mobile apps, hot videos, and social apps.

  - OBS Infrequent Access: applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response. Its application scenarios include file synchronization or sharing, and enterprise-level backup. This storage class has the same durability, low latency, and high throughput as the Standard storage class, with a lower cost, but its availability is slightly lower than the Standard storage class.
- A parallel file system is a high-performance file system provided by OBS. It is designed to provide high-performance file semantics for big data scenarios. For details, see **About PFS**.

For details about OBS storage classes, see **Storage Classes**.

Performance

Every time an OBS volume is mounted to a container workload, a resident process is created in the backend. When a workload uses too many OBS volumes or reads and writes a large number of object storage files, resident processes will consume a significant amount of memory. The amount of memory required in these scenarios is listed **Table 4-15**. To ensure stable running of the workload, make sure that the number of OBS volumes used does not exceed the requested memory. For example, if the workload requests for 4 GiB of memory, the number of OBS volumes should be no more than 4.

Table 4-15 Memory required by a single object storage resident process

Test Item	Memory Usage
Long-term stable running	About 50 MiB
Concurrent write to a 10-MB file from two processes	About 110 MiB
Concurrent write to a 10-MB file from four processes	About 220 MiB
Write to a 100-GB file from a single process	About 300 MiB

Application Scenarios

OBS supports the following mounting modes based on application scenarios:

- **Using an Existing OBS Bucket Through a Static PV**: static provisioning. You use an existing OBS volume to create a PV and then mount the PV to the workload through a PVC. This mode applies to scenarios where the underlying storage is available or billed on a yearly/monthly basis.
- **Using an OBS Bucket Through a Dynamic PV**: dynamic provisioning. You do not need to create OBS volumes in advance. Instead, specify a StorageClass during PVC creation. An OBS volume and a PV will be automatically created. This mode applies to scenarios where no underlying storage is available.

Billing

- When an OBS volume is mounted, the billing mode of OBS **automatically created** using StorageClass is **pay-per-use** by default. For OBS pricing details, see **OBS Pricing Details**.
- If you want to be billed on a yearly/monthly basis, **use existing OBS volumes**.

4.3.2 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 and implement data persistence and sharing in workloads.

Prerequisites

If you want to create a cluster using commands, use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Constraints

- If OBS volumes are used, the owner group and permission of the mount point cannot be modified.
- Hard links are not supported when common buckets are mounted.
- Multiple PVs can use the same OBS volume with the following restrictions:
  - Do not mount all PVCs/PVs that use the same underlying OBS volume to a pod. This will result in a pod startup failure because not all PVCs can be mounted to the pod due to the same **volumeHandle** values of these PVs.
  - The **persistentVolumeReclaimPolicy** parameter in the PVs should be set to **Retain**. If any other value is used, 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. In the navigation pane on the left, choose **Storage**. Then click the **PVCs** tab. In the upper right corner, click **Create PVC**. In the displayed dialog box, configure the parameters.

Parameter	Description
PVC Type	In this example, select **OBS**.
OBS Endpoint	To access OBS in a CCE Autopilot cluster, you need to create an OBS endpoint.

Parameter	Description
PVC Name	Enter the PVC name, which must be unique in the same namespace.
Creation Method	– If underlying storage is available, create a storage volume or use an existing storage volume to statically create a PVC based on whether a PV is available. – If no underlying storage is available, select **Dynamically provision**. For details, see **Using an OBS Bucket Through a Dynamic PV**. In this example, select **Create new** to create 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.
OBS[b]	Click **Select OBS**. In the dialog box displayed, select the OBS storage 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 avoid cascading deletion of underlying volumes.
Access Key (AK/SK)[b]	**Custom**: Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. – **Name**: Enter a secret name. – **Namespace**: Select the namespace where the secret is located. – **Access Key (AK/SK)**: Upload a key file in .csv format. For details, see **Obtaining an Access Key**.
Mount Options[b]	Enter the mounting parameter key-value pairs. For details, see **Configuring OBS Mount Options**.

NOTE

a: The parameter is available when **Creation Method** is set to **Use existing**.

b: The parameter is available when **Creation Method** is set to **Create new**.

2. Click **Create** to create a PVC and a PV.

   In the navigation pane on the left, choose **Storage**. View the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create a workload.

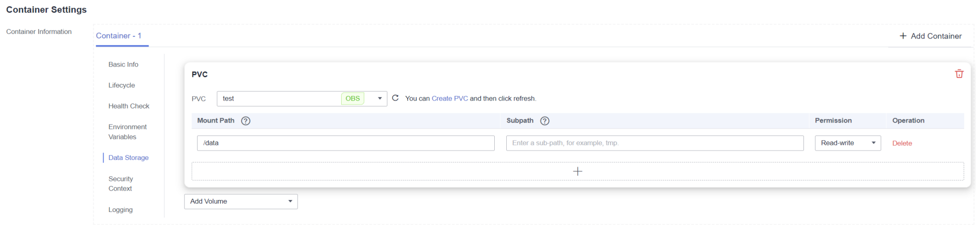
1. In the navigation pane on the left, choose **Workloads**. Then click the **Deployments** tab.
2. In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

   Mount and use storage volumes, as shown in **Table 4-16**. For details about other parameters, see **Creating a Workload**.

Table 4-16 Mounting a storage volume

Parameter	Description
PVC	Select an existing object storage volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the container path that the volume will be mounted to. 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. If there are such files, they will be replaced, which will lead to a container startup 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. **tmp**, for example, indicates 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 volume. – **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 OBS volume.



PVC: test OBS

You can Create PVC and then click refresh.

Mount Path	Subpath	Permission	Operation
/data	Enter a sub-path, for example, tmp.	Read-write	Delete

3. Configure other parameters and 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 OBS Bucket

Step 1 Use kubectl to connect to 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 PV is deleted while the underlying volume is retained.
  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 located.
    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 4-17 Key parameters

Parameter	Mandatory	Description
everest.io/reclaim-policy: retain-volume-only	No	Optional. Currently, only **retain-volume-only** is supported. If the reclaim policy is **Delete** and the 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**. – **s3fs**: Object bucket, which is mounted using s3fs.
volumeHandle	Yes	OBS volume name.
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, see **Regions and Endpoints**.

nodePublishSecretRef	No	Access key (AK/SK) used for mounting the OBS volume. You can use the AK/SK to create a secret and mount it to the PV. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.
		An example is as follows:
		`nodePublishSecretRef:`
		`  name: secret-demo`
		`  namespace: default`

mountOptions	No	Mount options. For details, see **Configuring OBS Mount Options**.

Parameter	Mandatory	Description
persistentVolumeReclaimPolicy	Yes	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 avoid cascading deletion of underlying volumes. **Delete**: – If **everest.io/reclaim-policy** is not specified, both the PV and OBS bucket 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 OBS bucket will be retained. **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** 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	The StorageClass for OBS volumes is **csi-obs**.

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.
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 4-18 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.
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 **Step 2.1**. The StorageClass for OBS volumes is **csi-obs**.
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-obs.yaml
```

Step 4 Create a workload.

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    # Custom volume name
          persistentVolumeClaim:
            claimName: pvc-obs    # Name of the created PVC.
```

2. Run the following command to create a workload that the OBS volume is mounted to:

   ```
   kubectl apply -f web-demo.yaml
   ```

   After the workload is created, you can try **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   Expected output:

   ```
   web-demo-846b489584-mjhm9   1/1     Running   0             46s
   web-demo-846b489584-wvv5s   1/1     Running   0             46s
   ```

2. Run the following commands in sequence to view the files in the **/data** path of the pods:

   ```
   kubectl exec web-demo-846b489584-mjhm9 -- ls /data
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```

   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Run the following command to create a file named **static** in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 --  touch /data/static
```

Step 3 Run the following command to check the files in the **/data** path:

```
kubectl exec web-demo-846b489584-mjhm9 -- ls /data
```

Expected output:

```
static
```

Step 4 **Verify data persistence.**

1. Run the following command to delete the pod named **web-demo-846b489584-mjhm9**:

   ```
   kubectl delete pod web-demo-846b489584-mjhm9
   ```

   Expected output:

   ```
   pod "web-demo-846b489584-mjhm9" deleted
   ```

   After the deletion, the Deployment controller automatically creates a replica.

2. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   The expected output is as follows, in which **web-demo-846b489584-d4d4j** is the newly created pod:

   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             110s
   web-demo-846b489584-wvv5s    1/1     Running   0             7m50s
   ```

3. Run the following command to check whether the files in the **/data** path of the new pod have been modified:

   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```

Expected output:

```
static
```

The **static** file is retained, indicating that the data in the file system can be stored persistently.

Step 5 **Verify data sharing.**

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep web-demo
   ```

   Expected output:

   ```
   web-demo-846b489584-d4d4j   1/1     Running   0             7m
   web-demo-846b489584-wvv5s   1/1     Running   0             13m
   ```

2. Run the following command to create a file named **share** in the **/data** path of either pod: In this example, select the pod named **web-demo-846b489584-d4d4j**.

   ```
   kubectl exec web-demo-846b489584-d4d4j --  touch /data/share
   ```

   Check the files in the **/data** path of the pod.

   ```
   kubectl exec web-demo-846b489584-d4d4j -- ls /data
   ```

   Expected output:

   ```
   share
   static
   ```

3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-846b489584-wvv5s**) as well to verify data sharing.

   ```
   kubectl exec web-demo-846b489584-wvv5s -- ls /data
   ```

   Expected output:

   ```
   share
   static
   ```

   After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

Related Operations

You can also perform the operations described in **Table 4-19**.

Table 4-19 Related operations

Operation	Description	Procedure
Creating a storage volume (PV)	You can create a PV on the CCE console.	1. In the navigation pane on the left, choose **Storage**. Then click the **PVs** tab. In the upper right corner, click **Create PersistentVolume**. In the displayed dialog box, configure the parameters. ● **Volume Type**: Select **OBS**. ● **OBS**: Click **Select OBS**. In the dialog box displayed, select the OBS storage that meets your 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**: There are two options: **Retain** and **Delete**. For details, see **PV Reclaim Policy** NOTE If multiple PVs use the same underlying storage volume, use **Retain** to avoid cascading deletion of underlying volumes. ● **Access Key (AK/SK)**: Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. ● **Mount Options**: Enter the mounting parameter key-value pairs. For details, see **Configuring OBS Mount Options**. 2. Click **Create**.
Updating an access key	You can update the access key of object storage 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 **Update Access Key**. 2. Upload a key file in .csv format. For details, see **Obtaining an Access Key**. Click **OK**.

Operation	Description	Procedure
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 (events are retained for one hour).
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.

4.3.3 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.
- Hard links are not supported when common buckets are mounted.
- OBS allows a single user to create a maximum of 100 buckets. If a large number of dynamic PVCs are created, the number of buckets may exceed the upper limit, and no more OBS buckets can be created. In this case, use OBS by calling its API or SDK and do not mount OBS buckets to workloads.

Automatically Creating an OBS Volume on the Console

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

Step 2 Dynamically create a PVC and PV.

1. In the navigation pane on the left, choose **Storage**. Then click the **PVCs** tab. In the upper right corner, click **Create PVC**. In the displayed dialog box, configure the parameters.

Parameter	Description
PVC Type	In this example, select **OBS**.

Parameter	Description
OBS Endpoint	To access OBS in a CCE Autopilot cluster, you need to create an OBS endpoint.
PVC Name	Enter the PVC name, which must be unique in the same 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, select either **Use existing** or **Create new**. For details about static creation, see **Using an Existing OBS Bucket Through a Static PV**. In this example, select **Dynamically provision**.
Storage Classes	The storage class of OBS volumes is **csi-obs**.
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.
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**.

Parameter	Description
Access Key (AK/SK)	**Custom**: Customize a secret if you want to assign different user permissions to different OBS storage devices. For details, see **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**. Only secrets with the **secret.kubernetes.io/used-by = csi** label can be selected. The secret type is cfe/secure-opaque. If no secret is available, click **Create Secret** to create one. – **Name**: Enter a secret name. – **Namespace**: Select the namespace where the secret is located. – **Access Key (AK/SK)**: Upload a key file in .csv format. For details, see **Obtaining an Access Key**.

2. Click **Create** to create a PVC and a PV.

   In the navigation pane on the left, choose **Storage**. View the created PVC and PV on the **PVCs** and **PVs** tabs, respectively.

Step 3 Create a workload.

1. In the navigation pane on the left, choose **Workloads**. Then click the **Deployments** tab.
2. In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **PVC**.

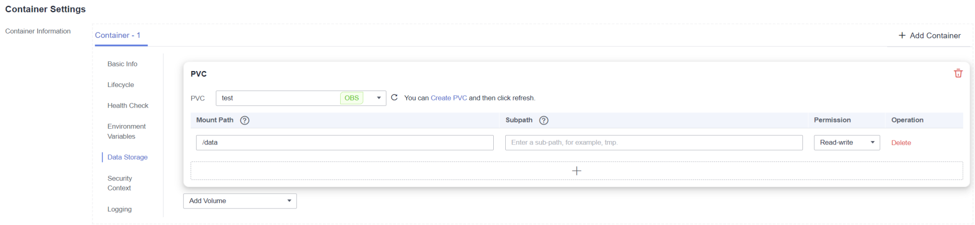
   Mount and use storage volumes. For details about the parameters, see **Table 4-20**. For other parameters, see **Creating a Workload**.

Table 4-20 Mounting a storage volume

Parameter	Description
PVC	Select an existing OBS volume.
Mount Path	Enter a mount path, for example, **/tmp**. This parameter indicates the container path that the volume will be mounted to. 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, or the files will be replaced, which will lead to a container startup 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.

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 volume. – **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 OBS volume.



Container Settings

PVC: test (OBS)

You can Create PVC and then click refresh.

Mount Path	Subpath	Permission	Operation
/data	Enter a sub-path, for example, tmp.	Read-write	Delete

3. Configure other parameters and 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) Automatically Creating an OBS Volume

Step 1 Use kubectl to connect to 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    # Object storage type.
    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.
spec:
  accessModes:
    - ReadWriteMany             # The value must be ReadWriteMany for object storage.
  resources:
```

```
  requests:
    storage: 1Gi              # OBS volume capacity.
 storageClassName: csi-obs    # The StorageClass type of OBS
```

Table 4-21 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**. – **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 **Using a Custom Access Key (AK/SK) to Mount an OBS Volume**.
csi.storage.k8s.io/node-publish-secret-namespace	No	Namespace of a custom secret.
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. The storage class name of OBS volumes is **csi-obs**.

2. Run the following command to create a PVC:

```
kubectl apply -f pvc-obs-auto.yaml
```

Step 3 Create a workload.

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    # Custom volume name
          persistentVolumeClaim:
            claimName: pvc-obs-auto    # Name of the created PVC.
```

2. Run the following command to create a workload that the OBS volume is mounted to:

   ```
   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 created file 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 file in the **/data** path of the new pod has 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 described in **Table 4-22**.

Table 4-22 Related operations

Operation	Description	Procedure
Updating an access key	Update the access key of object storage 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 **Update Access Key**. 2. Upload a key file in .csv format. For details, see **Obtaining an Access Key**. 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 (events are retained for one hour).
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.

4.3.4 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.

OBS Mount Options

When mounting an OBS volume, the Everest add-on presets the options described in **Table 4-23** and **Table 4-24** by default. The options in **Table 4-23** are mandatory. You can set other mount options if needed. For details, see **Mounting a Parallel File System**.

Table 4-23 Mandatory mount options configured by default

Parameter	Value	Description
use_ino	Leave it blank.	If enabled, obsfs allocates the **inode** number. Enabled by default in read/write mode.

Parameter	Value	Description
big_writes	Leave it blank.	If configured, the maximum size of the cache can be modified.
nonempty	Leave it blank.	Allows non-empty mount paths.
allow_other	Leave it blank.	Allows other users to access the parallel file system.
no_check_certificate	Leave it blank.	Disables server certificate verification.
sigv2	Leave it 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 4-24 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	0	Mask of the configuration file permission. For example, if the **umask** value is **022**, the directory permission (the maximum permission is **777**) is **755** (777 – 022 = 755, rwxr-xr-x).

Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in **OBS Mount Options**.

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Configure mount options in a PV. Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
    everest.io/reclaim-policy: retain-volume-only      # (Optional) The PV is deleted while the underlying volume is retained.
  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 located.
      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 **Using an Existing OBS Bucket Through a Static PV**.

----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 **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=027
```

Step 3 After the StorageClass is configured, you can use it to create a PVC. By default, the dynamically created PVs inherit the mount options configured in the StorageClass. For details, see **Using an Existing OBS Bucket Through a Static PV**.

----End

4.3.5 Using a Custom Access Key (AK/SK) to Mount an OBS Volume

Scenario

CCE supports custom access keys so IAM users can use their own custom access keys to mount an OBS volume. For details, see **How Can I Control Access to OBS?**

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.

Obtaining an Access Key

Step 1 Log in to the management 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 on the left, 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. If the value is **obsfs**, an OBS parallel file system is created and mounted.
volumeHandle	OBS bucket name.

Step 3 Create a PV.

kubectl create -f pv-example.yaml

After a PV is created, you can create a PVC and associate it with the PV.

Step 4 Create a YAML file for the PVC, for example, **pvc-example.yaml**.

Example YAML file for the PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

Step 5 Create a PVC.

kubectl create -f pvc-example.yaml

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

Mounting a Secret When Dynamically Creating an OBS Volume

When dynamically creating an OBS volume, you can use the following method to specify a secret:

Step 1 Create a YAML file for the PVC, for example, **pvc-example.yaml**.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    csi.storage.k8s.io/node-publish-secret-name: test-user
    csi.storage.k8s.io/node-publish-secret-namespace: default
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: obsfs
  name: obs-secret
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
```

Parameter	Description
csi.storage.k8s.io/node-publish-secret-name	Name of the secret
csi.storage.k8s.io/node-publish-secret-namespace	Namespace of the secret

Step 2 Create a PVC.

kubectl create -f pvc-example.yaml

After the PVC is created, you can create a workload and associate it with the PVC to create volumes.

----End

Verification

You can use a secret of an IAM user to mount an OBS volume. Assume that a workload named **obs-secret** is created, the mount path in the container is **/temp**, and the IAM user has the CCE **ReadOnlyAccess** and **Tenant Guest** permissions.

1. Query the name of the workload pod.

   kubectl get po | grep obs-secret

   Expected outputs:

   ```
   obs-secret-5cd558f76f-vxslv          1/1     Running   0          3m22s
   ```

2. Query the objects in the mount path. In this example, the query is successful.

   kubectl exec obs-secret-5cd558f76f-vxslv -- ls -l /temp/

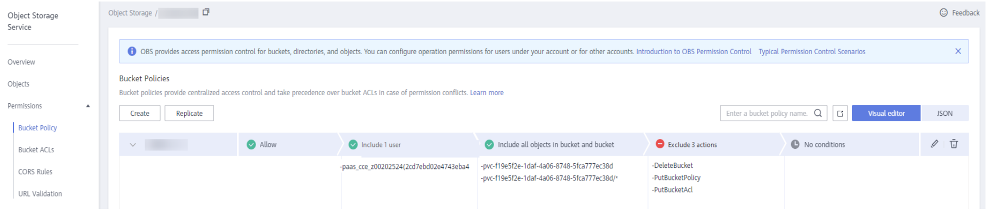
3. Write data into the mount path. In this example, the write operation failed.

   kubectl exec obs-secret-5cd558f76f-vxslv -- touch /temp/test

   Expected outputs:

```
touch: setting times of '/temp/test': No such file or directory
command terminated with exit code 1
```

4. Set the read/write permissions for the IAM user who mounted the OBS volume by referring to the bucket policy configuration.



OBS provides access permission control for buckets, directories, and objects. You can configure operation permissions for users under your account or for other accounts. Introduction to OBS Permission Control Typical Permission Control Scenarios

Bucket Policies

Bucket policies provide centralized access control and take precedence over bucket ACLs in case of permission conflicts. Learn more

[redacted]	Allow	Include 1 user	Include all objects in bucket and bucket	Exclude 3 actions	No conditions
		-paas_cce_z00202524(2cd7ebd02e4743eba4	-pvc-f19e5f2e-1daf-4a06-8748-5fca777ec38d -pvc-f19e5f2e-1daf-4a06-8748-5fca777ec38d/*	-DeleteBucket -PutBucketPolicy -PutBucketAcl	

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
```

4.4 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**. Then click the **Deployments** tab.

Step 3 In the upper right corner, click **Create Workload**. On the displayed page, click **Data Storage** in the **Container Settings** area and click **Add Volume** to select **EmptyDir**.

Step 4 **Table 4-25** describes the parameters for mounting the volume. For details about other parameters, see **Creating a Workload**.

Table 4-25 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 that the volume will be mounted to. 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. If there are such files, they will be replaced, which will lead to a container startup 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 volume. ● **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 Configure other parameters and click **Create Workload**.

----End

Using kubectl to Use a Temporary Path

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-emptydir.yaml** and edit it.

vi nginx-emptydir.yaml

Content of the YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-emptydir
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-emptydir
  template:
    metadata:
      labels:
        app: nginx-emptydir
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          volumeMounts:
            - name: vol-emptydir     # Volume name, which must be the same as the volume name in the 
volumes field.
              mountPath: /tmp        # Path to which an emptyDir volume is mounted.
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: vol-emptydir         # Volume name, which can be customized.
          emptyDir:
            medium: Memory           # emptyDir volume 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

5 Observability

5.1 Monitoring Center

5.1.1 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 AOM instances. This section describes how to enable cluster monitoring.

NOTICE

- After cluster monitoring is enabled, cluster metrics are reported to the selected AOM instance. Basic metrics are free, but custom metrics are billed based on the standard pricing of AOM. For details, see **AOM 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.

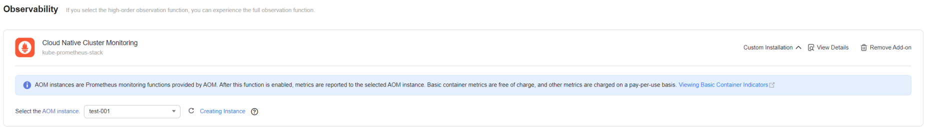
Constraints

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 5-1 Enabling cluster monitoring



Observability If you select the high-order observation function, you can experience the full observation function.

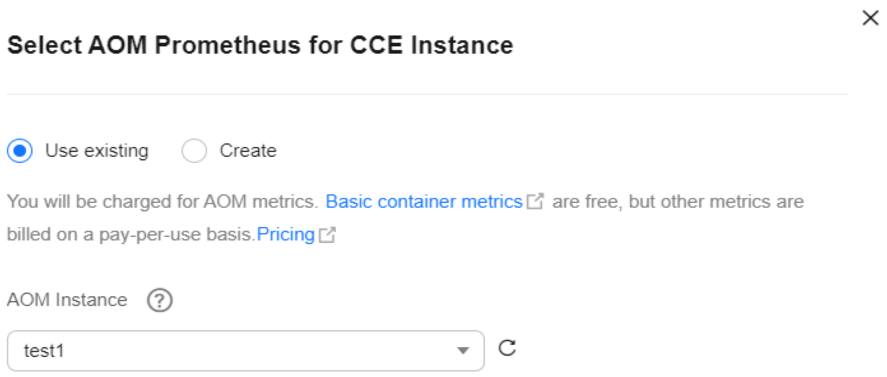
Cloud Native Cluster Monitoring
kube-prometheus-stack

AOM instances are Prometheus monitoring functions provided by AOM. After this function is enabled, metrics are reported to the selected AOM instance. Basic container metrics are free of charge, and other metrics are charged on a pay-per-use basis. Viewing Basic Container Indicators

Select the AOM instance. test-001

- **Enabling cluster monitoring on the Monitoring Center page**
  a. Click the cluster name to access the cluster console. 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 5-2 Enabling cluster monitoring



Select AOM Prometheus for CCE Instance

● Use existing ○ Create

You will be charged for AOM metrics. Basic container metrics are free, but other metrics are billed on a pay-per-use basis. Pricing

AOM Instance

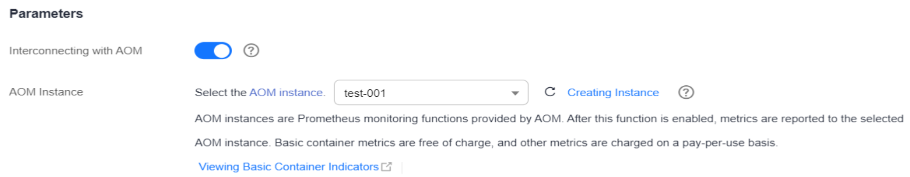
test1

  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 to access the cluster console. In the navigation pane on the left, choose **Add-ons**.
  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 5-3 Installing the Cloud Native Cluster Monitoring add-on



Parameters

Interconnecting with AOM: on

AOM Instance: Select the AOM instance. test-001 | Creating Instance

AOM instances are Prometheus monitoring functions provided by AOM. After this function is enabled, metrics are reported to the selected AOM instance. Basic container metrics are free of charge, and other metrics are charged on a pay-per-use basis.

Viewing Basic Container Indicators

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.

5.1.2 Cluster Monitoring

On the **Clusters** tab, you can view the monitoring data of each cluster from multiple dimensions, as described in **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.

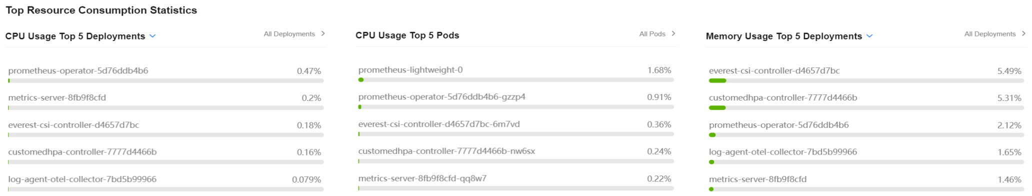
Step 2 In the navigation pane on the left, choose **Monitoring Center**. Then click **Clusters**.

----End

Resource Overview

Resource Overview displays the percentages of abnormal resources in workloads and pods and the total number of namespaces.

Figure 5-4 Resource Overview



Top Resource Consumption Statistics

CPU Usage Top 5 Deployments	All Deployments >
prometheus-operator-5d76ddb4b6	0.47%
metrics-server-8fb9f8cfd	0.2%
everest-csi-controller-d4657d7bc	0.18%
customedhpa-controller-7777d4466b	0.16%
log-agent-otel-collector-7bd5b99966	0.079%

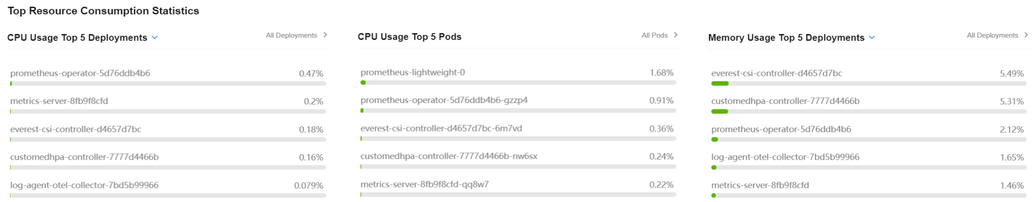
CPU Usage Top 5 Pods	All Pods >
prometheus-lightweight-0	1.68%
prometheus-operator-5d76ddb4b6-gzzp4	0.91%
everest-csi-controller-d4657d7bc-6m7vd	0.36%
customedhpa-controller-7777d4466b-nw6sx	0.24%
metrics-server-8fb9f8cfd-qq8w7	0.22%

Memory Usage Top 5 Deployments	All Deployments >
everest-csi-controller-d4657d7bc	5.49%
customedhpa-controller-7777d4466b	5.31%
prometheus-operator-5d76ddb4b6	2.12%
log-agent-otel-collector-7bd5b99966	1.65%
metrics-server-8fb9f8cfd	1.46%

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 5-5 Top Resource Consumption Statistics



Top Resource Consumption Statistics

CPU Usage Top 5 Deployments	All Deployments
prometheus-operator-5d76ddb4b6	0.47%
metrics-server-8fb9f8cfd	0.2%
everest-csi-controller-d4657d7bc	0.18%
customedhpa-controller-7777d4466b	0.16%
log-agent-otel-collector-7bd5b99966	0.079%

CPU Usage Top 5 Pods	All Pods
prometheus-lightweight-0	1.68%
prometheus-operator-5d76ddb4b6-gzzp4	0.91%
everest-csi-controller-d4657d7bc-6m7vd	0.36%
customedhpa-controller-7777d4466b-nw6sx	0.24%
metrics-server-8fb9f8cfd-qq8w7	0.22%

Memory Usage Top 5 Deployments	All Deployments
everest-csi-controller-d4657d7bc	5.49%
customedhpa-controller-7777d4466b	5.31%
prometheus-operator-5d76ddb4b6	2.12%
log-agent-otel-collector-7bd5b99966	1.65%
metrics-server-8fb9f8cfd	1.46%

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 **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.

5.1.3 Workload Monitoring

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**. Then click **Workloads**.

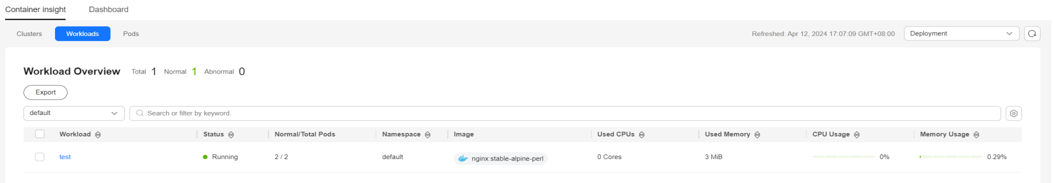
Information about all workloads is displayed. To view the monitoring data of a workload, click the workload name to go to the **Overview** tab. You can also click **Pods** or **Monitoring** to view corresponding information.

----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 5-6 Workloads



Container insight | Dashboard

Clusters | Workloads | Pods

Refreshed: Apr 12, 2024 17:07:09 GMT+08:00 | Deployment

Workload Overview Total 1 Normal 1 Abnormal 0

Export

default | Search or filter by keyword.

	Workload	Status	Normal/Total Pods	Namespace	Image	Used CPUs	Used Memory	CPU Usage	Memory Usage
□	test	● Running	2 / 2	default	nginx:stable-alpine-perl	0 Cores	3 MiB	0%	0.29%

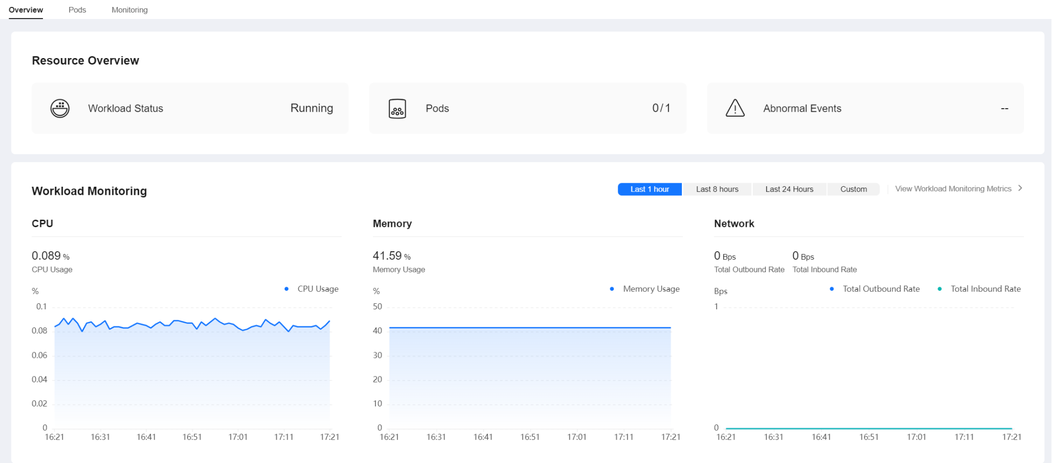
You can select a workload type in the upper right corner, or select **Workload name**, **Status**, and **Namespace** above the list to quickly locate the required workload.

You can click **Export** in the upper right corner of the list to export details of 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 5-7 Resource overview and monitoring overview



Overview | Pods | Monitoring

Resource Overview

Workload Status Running

Pods 0/1

Abnormal Events --

Workload Monitoring

Last 1 hour | Last 8 hours | Last 24 Hours | Custom | View Workload Monitoring Metrics

CPU

0.089 % CPU Usage

Memory

41.59 % Memory Usage

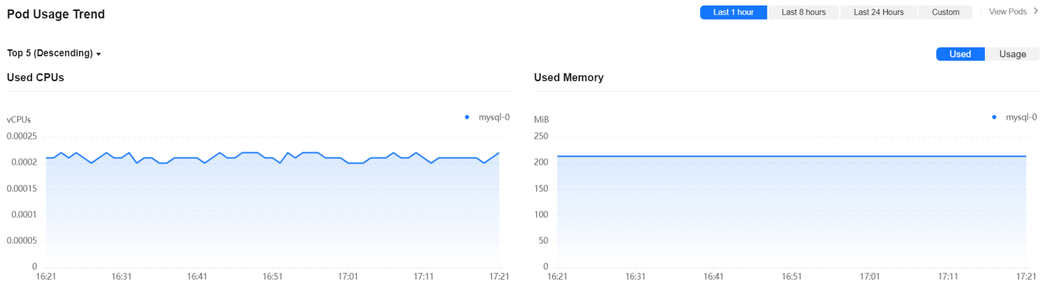
Network

0 Bps Total Outbound Rate

0 Bps Total Inbound Rate

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 5-8 Pod usage trend

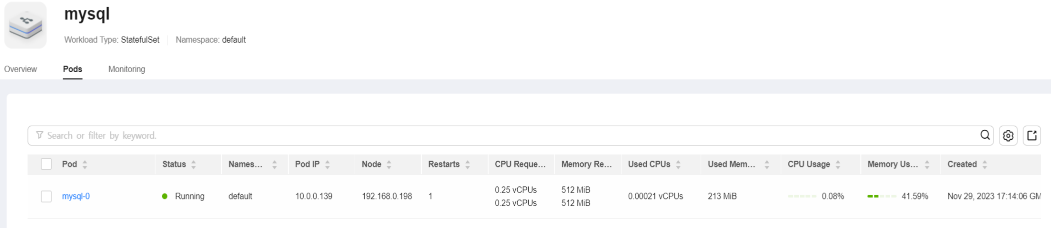


For more metrics, go to the **Monitoring** tab.

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 5-9 Pods



mysql

Workload Type: StatefulSet | Namespace: default

	Pod	Status	Names...	Pod IP	Node	Restarts	CPU Reque...	Memory Re...	Used CPUs	Used Mem...	CPU Usage	Memory Us...	Created
□	mysql-0	● Running	default	10.0.0.139	192.168.0.198	1	0.25 vCPUs 0.25 vCPUs	512 MiB 512 MiB	0.00021 vCPUs	213 MiB	0.08%	41.59%	Nov 29, 2023 17:14:06 GM

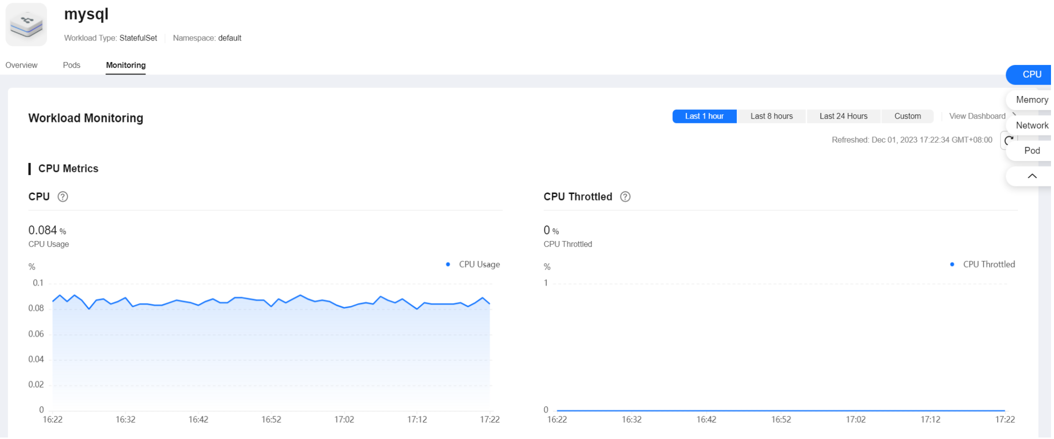
You can find the desired pod by name, status, namespace, IP address, or node. You can click **Export** in the upper right corner of the list to export details of all pods or the selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

You can click the name of a pod to view its monitoring data. For more information, see **Pod Monitoring**.

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 data, click **View Dashboard** to access the **Dashboard** page. For details, see **Using Dashboard**.

Figure 5-10 Workload monitoring



mysql

Workload Type: StatefulSet | Namespace: default

Refreshed: Dec 01, 2023 17:22:34 GMT+08:00

- **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.

5.1.4 Pod Monitoring

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 **Pods**.

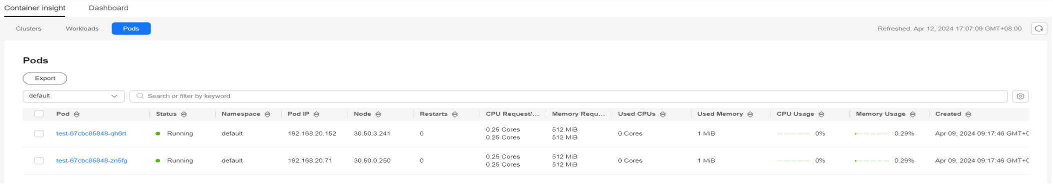
Information about all pods is displayed. To view the monitoring data of a pod, click the pod name to go to the **Overview** tab. You can also click the **Containers** or **Monitoring** tab to view the corresponding information.

----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 5-11 Pods



Container insight
Dashboard
Clusters
Workloads
Pods
Refreshed: Apr 12, 2024 17:07:09 GMT+08:00
Pods
Export
default
Search or filter by keyword.

	Pod	Status	Namespace	Pod IP	Node	Restarts	CPU Request/...	Memory Requ...	Used CPUs	Used Memory	CPU Usage	Memory Usage	Created
□	test-67cbc85848-qh6rt	● Running	default	192.168.20.152	30.50.3.241	0	0.25 Cores 0.25 Cores	512 MiB 512 MiB	0 Cores	1 MiB	0%	0.29%	Apr 09, 2024 09:17:46 GMT+(
□	test-67cbc85848-zn5fg	● Running	default	192.168.20.71	30.50.0.250	0	0.25 Cores 0.25 Cores	512 MiB 512 MiB	0 Cores	1 MiB	0%	0.29%	Apr 09, 2024 09:17:46 GMT+(

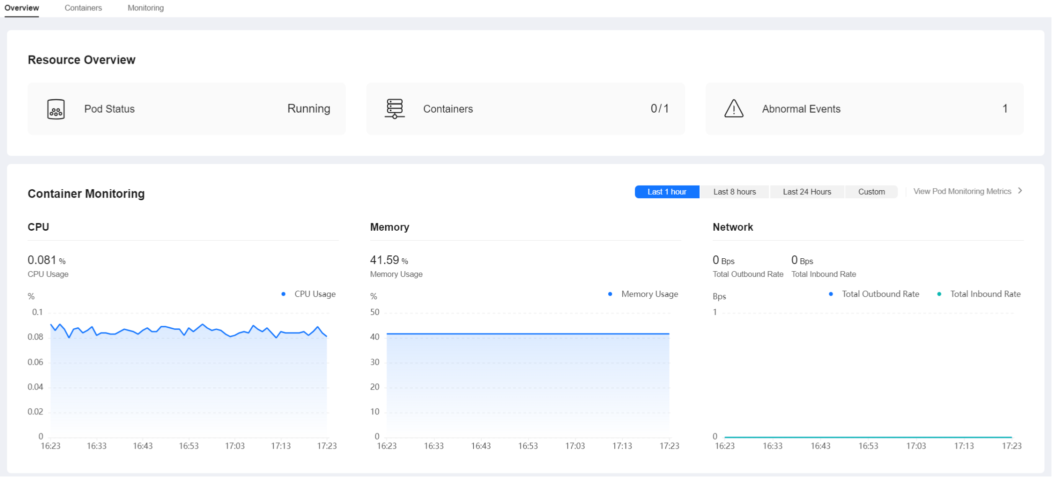
You can select a namespace in the upper right corner, or select **Pod**, **Status**, **Namespace**, **Pod IP**, and **Node** above the list to quickly locate the required pod.

You can click **Export** in the upper right corner of the list to export details 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 in the last hour, including the CPU usage, memory usage, and network inbound/outbound rate.

Figure 5-12 Resource overview and monitoring overview



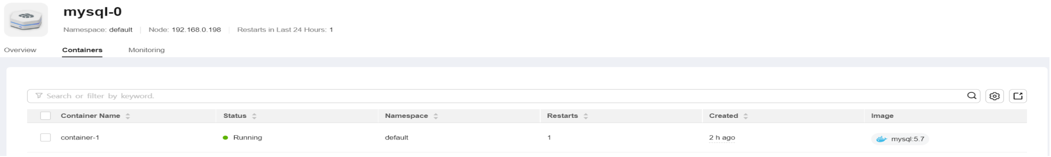
The **Overview** tab also shows the container usage trend. You can switch the metrics in the upper right corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each container in the pod. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper left 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 5-13 Containers


mysql-0
Namespace: default | Node: 192.168.0.198 | Restarts in Last 24 Hours: 1
Overview
Containers
Monitoring
Search or filter by keyword.

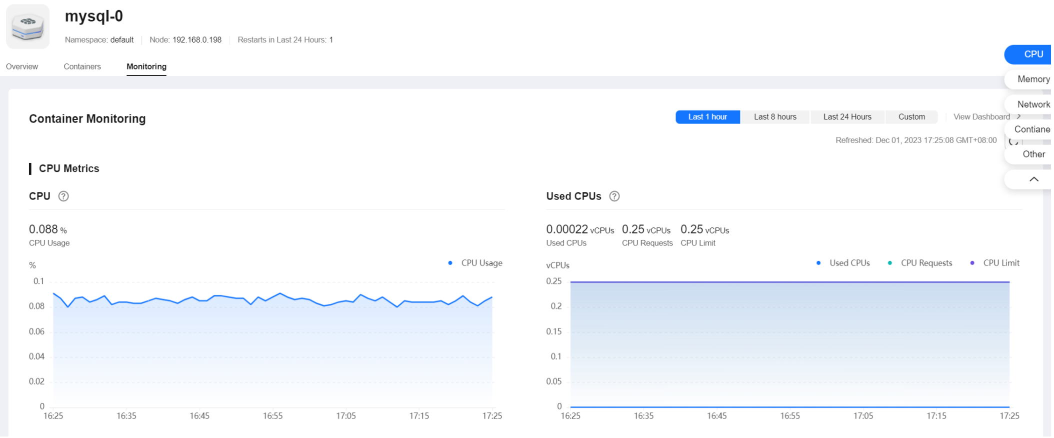
□	Container Name	Status	Namespace	Restarts	Created	Image
□	container-1	● Running	default	1	2 h ago	mysql:5.7

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 details 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 **Using Dashboard**.

Figure 5-14 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.

5.1.5 Dashboard

5.1.5.1 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

5.2 Logging

5.2.1 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. It 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 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

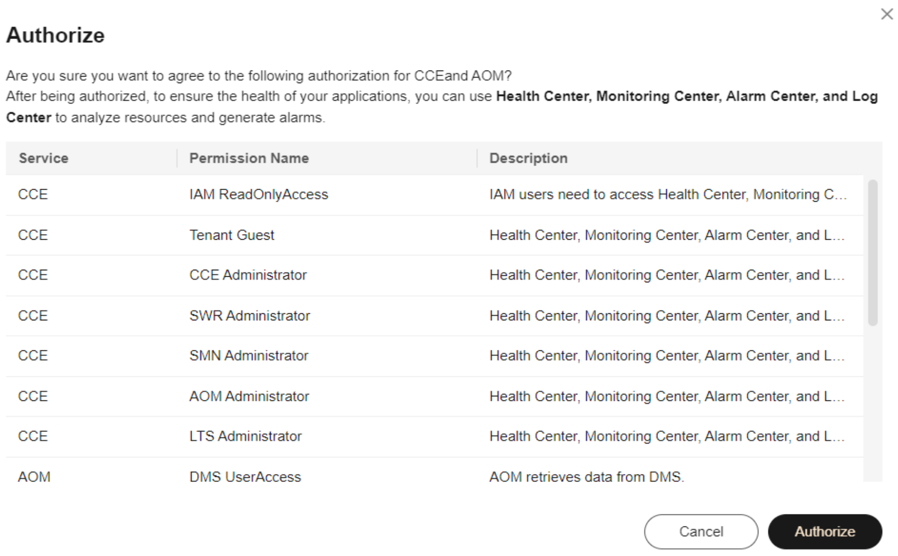
1. Log in to the CCE console.
2. In the upper right corner, click **Buy Cluster**.
3. In the **Select Add-on** step, select **Cloud Native Logging**.
4. Click **Next: Add-on Configuration** in the lower right corner and select the required logs.
   - Container logs: A log collection policy named **default-stdout** 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**.

Figure 5-15 Adding authorization



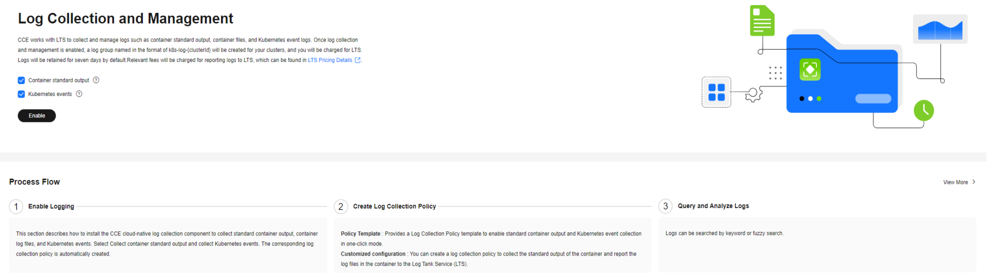
Authorize

Are you sure you want to agree to the following authorization for CCEand AOM?
After being authorized, to ensure the health of your applications, you can use **Health Center, Monitoring Center, Alarm Center, and Log Center** to analyze resources and generate alarms.

Service	Permission Name	Description
CCE	IAM ReadOnlyAccess	IAM users need to access Health Center, Monitoring C...
CCE	Tenant Guest	Health Center, Monitoring Center, Alarm Center, and L...
CCE	CCE Administrator	Health Center, Monitoring Center, Alarm Center, and L...
CCE	SWR Administrator	Health Center, Monitoring Center, Alarm Center, and L...
CCE	SMN Administrator	Health Center, Monitoring Center, Alarm Center, and L...
CCE	AOM Administrator	Health Center, Monitoring Center, Alarm Center, and L...
CCE	LTS Administrator	Health Center, Monitoring Center, Alarm Center, and L...
AOM	DMS UserAccess	AOM retrieves data from DMS.

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.

Figure 5-16 Enabling log collection



Log Collection and Management

CCE works with LTS to collect and manage logs such as container standard output, container files, and Kubernetes event logs. Once log collection and management is enabled, a log group named in the format of k8s-log-{clusterId} will be created for your clusters, and you will be charged for LTS. Logs will be retained for seven days by default.Relevant fees will be charged for reporting logs to LTS, which can be found in LTS Pricing Details.

☑ Container standard output

☑ Kubernetes events

Process Flow

1 Enable Logging

This section describes how to install the CCE cloud-native log collection component to collect standard container output, container log files, and Kubernetes events. Select Collect container standard output and collect Kubernetes events. The corresponding log collection policy is automatically created.

2 Create Log Collection Policy

Policy Template : Provides a Log Collection Policy template to enable standard container output and Kubernetes event collection in one-click mode.
Customized configuration : You can create a log collection policy to collect the standard output of the container and report the log files in the container to the Log Tank Service (LTS).

3 Query and Analyze Logs

Logs can be searched by keyword or fuzzy search.

Step 2 View and configure log collection policies.

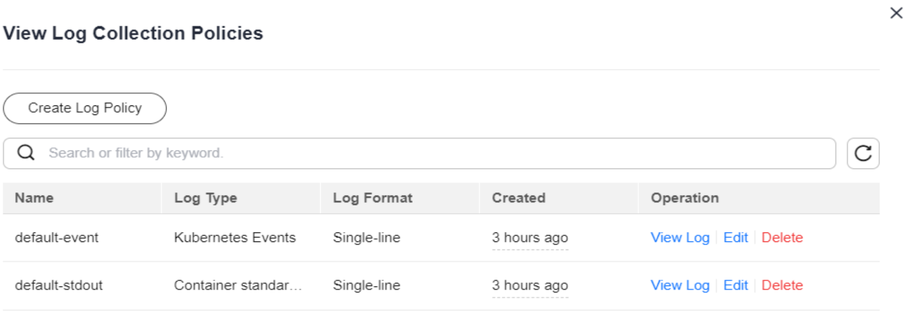
1. On the CCE console, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.

   If **Container standard output** and **Kubernetes events** are selected during the 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**: collects standard output. Default log group: **k8s-logs-{*Cluster ID*}**; Default log stream: **stdout-{*Cluster ID*}**

- **default-event**: collects Kubernetes events. Default log group: **k8s-logs-{*Cluster ID*}**; Default log stream: **event-{*Cluster ID*}**

Figure 5-17 Viewing log collection policies



View Log Collection Policies

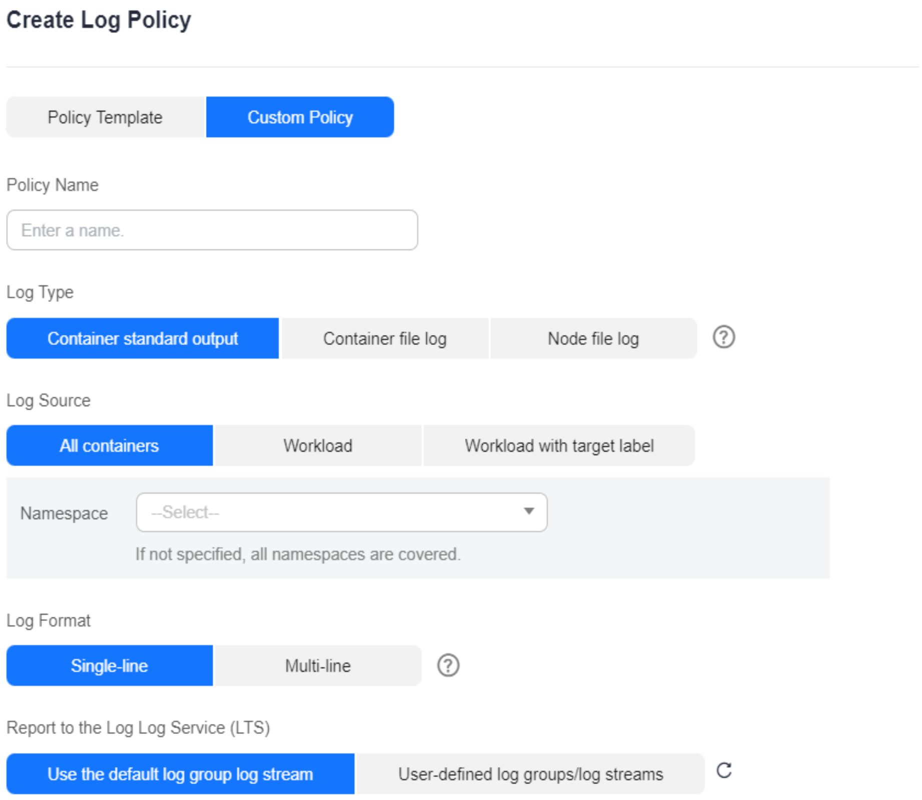
Create Log Policy

Search or filter by keyword.

Name	Log Type	Log Format	Created	Operation
default-event	Kubernetes Events	Single-line	3 hours ago	View Log | Edit | Delete
default-stdout	Container standar...	Single-line	3 hours ago	View Log | Edit | Delete

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 a default log collection policy.
   - **Custom Policy**: You can use this option to create a log collection policy.

Figure 5-18 Custom policy



Create Log Policy

Policy Template | Custom Policy (selected)

Policy Name: Enter a name.

Log Type: Container standard output (selected) | Container file log | Node file log

Log Source: All containers (selected) | Workload | Workload with target label

Namespace: --Select--

If not specified, all namespaces are covered.

Log Format: Single-line (selected) | Multi-line

Report to the Log Log Service (LTS): Use the default log group log stream (selected) | User-defined log groups/log streams

Table 5-1 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.

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)`

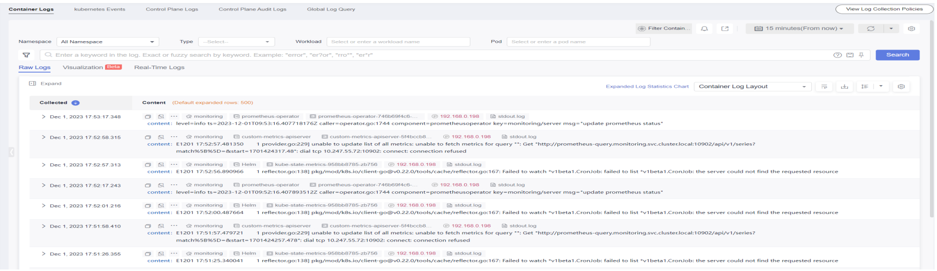
Parameter	Description
Report to the Log Tank Service (LTS)	This parameter is used to configure the log group and log stream for log reporting. – Default log groups/log streams: The default log group (**k8s-log-***{Cluster ID}*) and default log stream (**stdout-***{Cluster ID}*) are automatically selected. – Custom log groups/log streams: You can select any log group and log stream. ■ 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**. ■ 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.

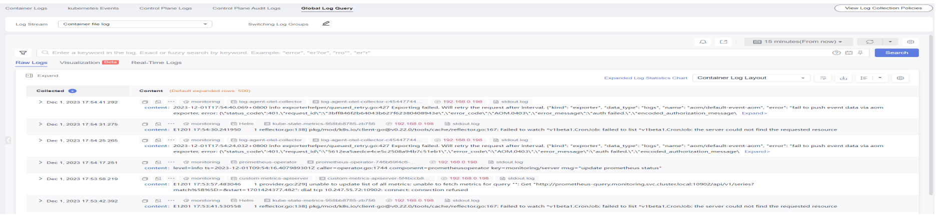
Figure 5-19 Querying container logs



```
Dec 1, 2023 17:53:17.348   monitoring  prometheus-operator  prometheus-operator-746b69f4c6-...  192.168.0.198  stdout.log
content: level=info ts=2023-12-01T09:53:16.407718176Z caller=operator.go:1744 component=prometheusoperator key=monitoring/server msg="update prometheus status"
Dec 1, 2023 17:52:58.315   monitoring  custom-metrics-apiserver  custom-metrics-apiserver-5f4bccb8...  192.168.0.198  stdout.log
content: E1201 17:52:57.481350       1 provider.go:229] unable to update list of all metrics: unable to fetch metrics for query "": Get "http://prometheus-query.monitoring.svc.cluster.local:10902/api/v1/series?
         match%5B%5D=&start=1701424317.48": dial tcp 10.247.55.72:10902: connect: connection refused
Dec 1, 2023 17:52:57.313   monitoring  Helm  kube-state-metrics-958bb8785-zb756  192.168.0.198  stdout.log
content: E1201 17:52:56.890966       1 reflector.go:138] pkg/mod/k8s.io/client-go@v0.22.0/tools/cache/reflector.go:167: Failed to watch *v1beta1.CronJob: failed to list *v1beta1.CronJob: the server could not find the requested resource
Dec 1, 2023 17:52:17.243   monitoring  prometheus-operator  prometheus-operator-746b69f4c6-...  192.168.0.198  stdout.log
content: level=info ts=2023-12-01T09:52:16.407893512Z caller=operator.go:1744 component=prometheusoperator key=monitoring/server msg="update prometheus status"
Dec 1, 2023 17:52:01.216   monitoring  Helm  kube-state-metrics-958bb8785-zb756  192.168.0.198  stdout.log
content: E1201 17:52:00.487664       1 reflector.go:138] pkg/mod/k8s.io/client-go@v0.22.0/tools/cache/reflector.go:167: Failed to watch *v1beta1.CronJob: failed to list *v1beta1.CronJob: the server could not find the requested resource
Dec 1, 2023 17:51:58.410   monitoring  custom-metrics-apiserver  custom-metrics-apiserver-5f4bccb8...  192.168.0.198  stdout.log
content: E1201 17:51:57.479721       1 provider.go:229] unable to update list of all metrics: unable to fetch metrics for query "": Get "http://prometheus-query.monitoring.svc.cluster.local:10902/api/v1/series?
         match%5B%5D=&start=1701424257.478": dial tcp 10.247.55.72:10902: connect: connection refused
Dec 1, 2023 17:51:26.355   monitoring  Helm  kube-state-metrics-958bb8785-zb756  192.168.0.198  stdout.log
content: E1201 17:51:25.340041       1 reflector.go:138] pkg/mod/k8s.io/client-go@v0.22.0/tools/cache/reflector.go:167: Failed to watch *v1beta1.CronJob: failed to list *v1beta1.CronJob: the server could not find the requested resource
```

- **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.

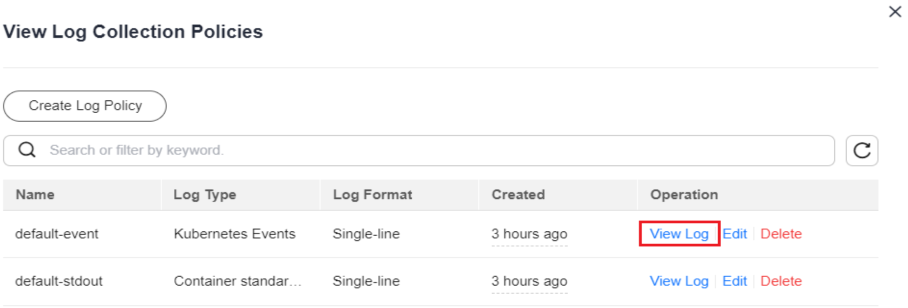
Figure 5-20 Global log query



```
> Dec 1, 2023 17:54:41.292   monitoring  log-agent-otel-collector  log-agent-otel-collector-c45447744...  192.168.0.198  stdout.log
content: 2023-12-01T17:54:40.069+0800 info exporterhelper/queued_retry.go:427 Exporting failed. Will retry the request after interval. ("kind": "exporter", "data_type": "logs", "name": "aom/default-event-aom", "error": "fail to push event data via aom exporter, error: {\"status_code\":401,\"request_id\":\"3bff846f2b64043b627f62380408943e\",\"error_code\":\"AOM.0403\",\"error_message\":\"auth failed.\",\"encoded_authorization_message\  Expand>
> Dec 1, 2023 17:54:31.275   monitoring  Helm  kube-state-metrics-958bb8785-zb756  192.168.0.198  stdout.log
content: E1201 17:54:30.241950     1 reflector.go:138] pkg/mod/k8s.io/client-go@v0.22.0/tools/cache/reflector.go:167: Failed to watch *v1beta1.CronJob: failed to list *v1beta1.CronJob: the server could not find the requested resource
> Dec 1, 2023 17:54:25.265   monitoring  log-agent-otel-collector  log-agent-otel-collector-c45447744...  192.168.0.198  stdout.log
content: 2023-12-01T17:54:24.032+0800 info exporterhelper/queued_retry.go:427 Exporting failed. Will retry the request after interval. ("kind": "exporter", "data_type": "logs", "name": "aom/default-event-aom", "error": "fail to push event data via aom exporter, error: {\"status_code\":401,\"request_id\":\"5612ea5aedce4ce5c2508a94b7c51eb1\",\"error_code\":\"AOM.0403\",\"error_message\":\"auth failed.\",\"encoded_authorization_message\  Expand>
> Dec 1, 2023 17:54:17.251   monitoring  prometheus-operator  prometheus-operator-746b69f4c6-...  192.168.0.198  stdout.log
content: level=info ts=2023-12-01T09:54:16.407989301Z caller=operator.go:1744 component=prometheusoperator key=monitoring/server msg="update prometheus status"
> Dec 1, 2023 17:53:58.219   monitoring  custom-metrics-apiserver  custom-metrics-apiserver-5f4bccb8...  192.168.0.198  stdout.log
content: E1201 17:53:57.483046     1 provider.go:229] unable to update list of all metrics: unable to fetch metrics for query "": Get "http://prometheus-query.monitoring.svc.cluster.local:10902/api/v1/series?match%5B%5D=&start=1701424377.482": dial tcp 10.247.55.72:10902: connect: connection refused
> Dec 1, 2023 17:53:42.392   monitoring  Helm  kube-state-metrics-958bb8785-zb756  192.168.0.198  stdout.log
content: E1201 17:53:41.530558     1 reflector.go:138] pkg/mod/k8s.io/client-go@v0.22.0/tools/cache/reflector.go:167: Failed to watch *v1beta1.CronJob: failed to list *v1beta1.CronJob: the server could not find the requested resource
```

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 5-21 Viewing logs



Name	Log Type	Log Format	Created	Operation
default-event	Kubernetes Events	Single-line	3 hours ago	View Log Edit Delete
default-stdout	Container standar...	Single-line	3 hours ago	View Log Edit Delete

----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
  ```

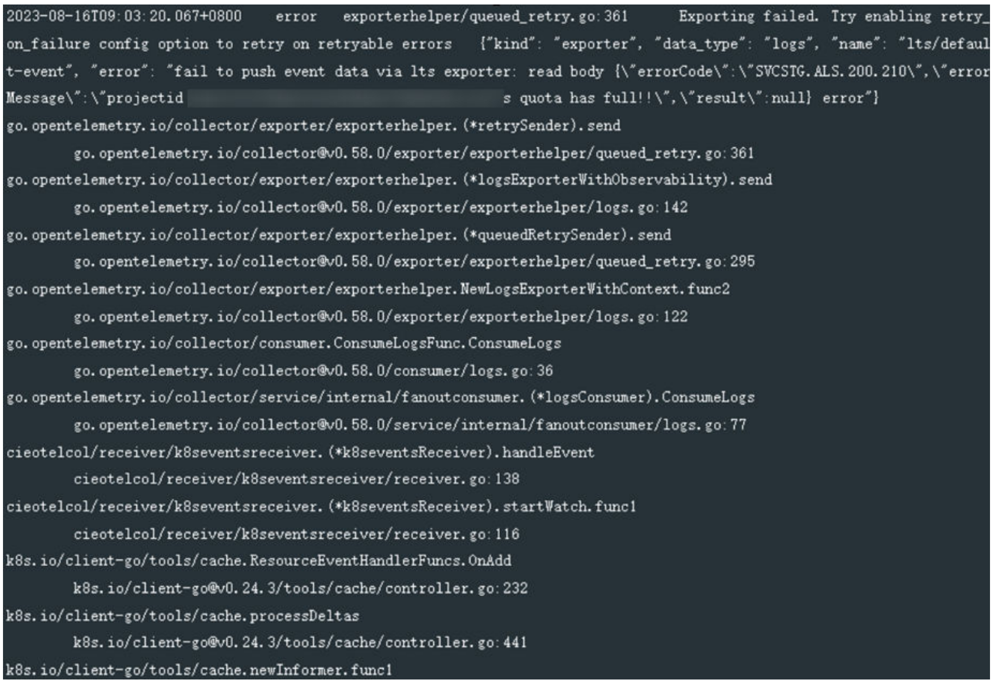
- **"Failed to create log group, the number of log groups exceeds the quota" is reported in the standard output log of log-operator.**

  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.**

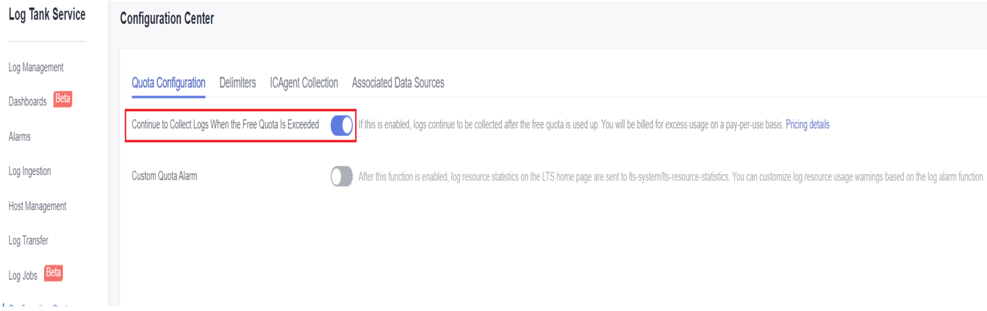


```
2023-08-16T09:03:20.067+0800    error   exporterhelper/queued_retry.go:361      Exporting failed. Try enabling retry_
on_failure config option to retry on retryable errors   {"kind": "exporter", "data_type": "logs", "name": "lts/defaul
t-event", "error": "fail to push event data via lts exporter: read body {\"errorCode\":\"SVCSTG.ALS.200.210\",\"error
Message\":\"projectid [redacted] s quota has full!!\",\"result\":null} error"}
go.opentelemetry.io/collector/exporter/exporterhelper.(*retrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:361
go.opentelemetry.io/collector/exporter/exporterhelper.(*logsExporterWithObservability).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:142
go.opentelemetry.io/collector/exporter/exporterhelper.(*queuedRetrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:295
go.opentelemetry.io/collector/exporter/exporterhelper.NewLogsExporterWithContext.func2
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:122
go.opentelemetry.io/collector/consumer.ConsumeLogsFunc.ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/consumer/logs.go:36
go.opentelemetry.io/collector/service/internal/fanoutconsumer.(*logsConsumer).ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/service/internal/fanoutconsumer/logs.go:77
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).handleEvent
        cieotelcol/receiver/k8seventsreceiver/receiver.go:138
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).startWatch.func1
        cieotelcol/receiver/k8seventsreceiver/receiver.go:116
k8s.io/client-go/tools/cache.ResourceEventHandlerFuncs.OnAdd
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:232
k8s.io/client-go/tools/cache.processDeltas
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:441
k8s.io/client-go/tools/cache.newInformer.func1
```

Solution:

LTS provides a free log quota. If the quota is used up, you will be charged for the excess log usage. If an error message is displayed, the free quota has been used up. To continue collecting logs, log in to the LTS console, choose **Configuration Center** in the navigation pane on the left, and enable **Continue to Collect Logs When the Free Quota Is Exceeded**.

Figure 5-22 Quota configuration



Continue to Collect Logs When the Free Quota Is Exceeded ● If this is enabled, logs continue to be collected after the free quota is used up. You will be billed for excess usage on a pay-per-use basis. Pricing details

Custom Quota Alarm ○ After this function is enabled, log resource statistics on the LTS home page are sent to lts-system/lts-resource-statistics. You can customize log resource usage warnings based on the log alarm function.

- **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**.

5.2.2 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.

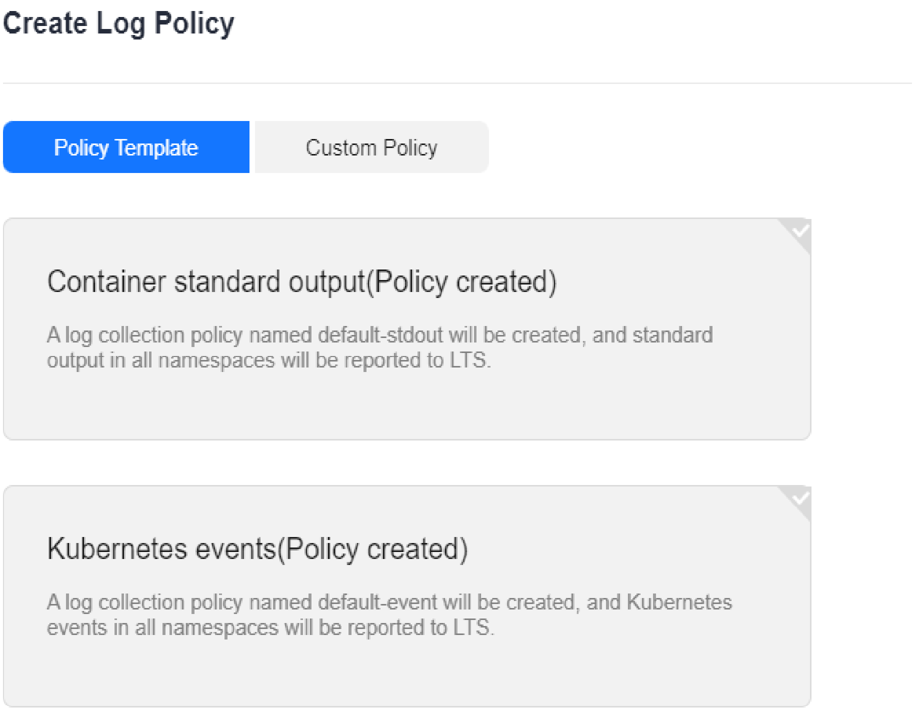
You can select **Kubernetes events** when installing the Cloud Native Logging add-on. A default log collection policy will be created, and all events collected will be reported to LTS. For details about how to install the add-on, see **Collecting Logs**.

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.



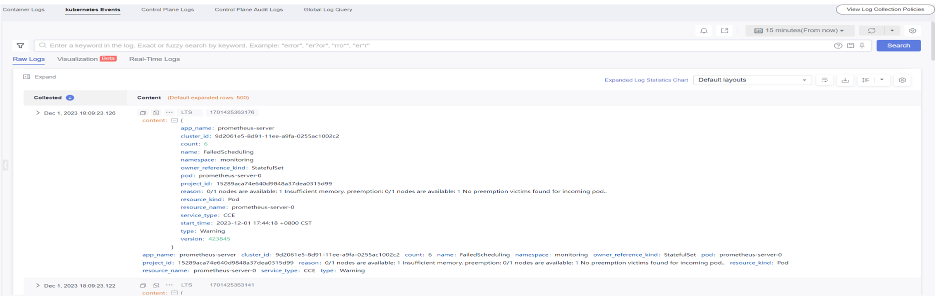
Container standard output(Policy created)

A log collection policy named default-stdout will be created, and standard output in all namespaces will be reported to LTS.

Kubernetes events(Policy created)

A log collection policy named default-event will be created, and Kubernetes events in all namespaces will be reported to LTS.

4. On the **Logging** page, select the log stream configured in the log collection policy to view the events reported to LTS.



Container Logs
kubernetes Events
Control Plane Logs
Control Plane Audit Logs
Global Log Query
View Log Collection Policies
15 minutes(From now)
Enter a keyword in the log. Exact or fuzzy search by keyword. Example: "error", "er?or", "rro*", "er*r"
Search
Raw Logs
Visualization Beta
Real-Time Logs
Expand
Expanded Log Statistics Chart
Default layouts
Collected
Content (Default expanded rows: 500)

```
> Dec 1, 2023 18:09:23.126   LTS   1701425363176
content: {
        app_name: prometheus-server
        cluster_id: 9d2061e5-8d91-11ee-a9fa-0255ac1002c2
        count: 6
        name: FailedScheduling
        namespace: monitoring
        owner_reference_kind: StatefulSet
        pod: prometheus-server-0
        project_id: 15289aca74e640d9848a37dea0315d99
        reason: 0/1 nodes are available: 1 Insufficient memory. preemption: 0/1 nodes are available: 1 No preemption victims found for incoming pod..
        resource_kind: Pod
        resource_name: prometheus-server-0
        service_type: CCE
        start_time: 2023-12-01 17:44:18 +0800 CST
        type: Warning
        version: 423845
    }
app_name: prometheus-server  cluster_id: 9d2061e5-8d91-11ee-a9fa-0255ac1002c2  count: 6  name: FailedScheduling  namespace: monitoring  owner_reference_kind: StatefulSet  pod: prometheus-server-0
project_id: 15289aca74e640d9848a37dea0315d99  reason: 0/1 nodes are available: 1 Insufficient memory. preemption: 0/1 nodes are available: 1 No preemption victims found for incoming pod..  resource_kind: Pod
resource_name: prometheus-server-0  service_type: CCE  type: Warning
> Dec 1, 2023 18:09:23.122   LTS   1701425363141
content: {
```

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

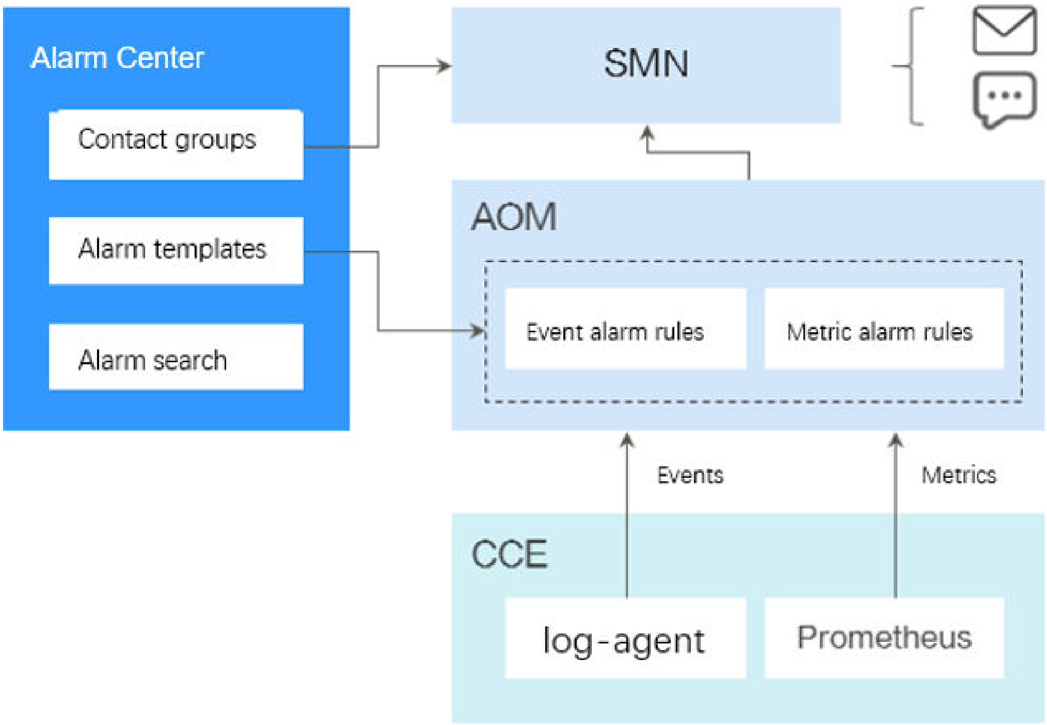
5.3 Alarm Center

5.3.1 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 Center that allows you to quickly configure and view common alarms (such as resource usage alarms).

Figure 5-23 Alarm Center architecture



- Alarm Center

  Based on the alarm capabilities of AOM, Alarm Center provides quick alarm search and configuration for clusters. You can use Alarm Center to configure common alarm rules with just a few clicks.

- AOM

  AOM is a one-stop, multi-dimensional O&M management platform for monitoring and alarm reporting of cloud applications.

- Simple Message Notification

  Simple Message Notification (SMN) connects cloud applications. After events or alarms are triggered, SMN will send notifications. In cloud native scenarios, alarms triggered by AOM are sent by SMS message, email, or HTTP message configured on SMN.

5.3.2 Configuring Alarms in Alarm Center

By using AOM, Alarm Center can promptly detect cluster faults and generate alarms for service stability. Alarm Center provides built-in alarm rules, which can free you from manually configuring alarm rules on AOM. These rules are established based on the extensive cluster O&M experience of our Huawei Cloud container team and can cover container service exceptions, key metric alarms of basic cluster resources, and metric alarms of applications in a cluster to meet your routine O&M requirements.

Constraints

Only Huawei Cloud accounts, HUAWEI IDs, or IAM users with CCE administrator or FullAccess permissions can perform all operations using Alarm Center. IAM users with the CCE ReadOnlyAccess permission can only view all resources.

Enabling Alarm Center

Step 1 Click the cluster name to access the cluster console. In the navigation pane on the left, choose **Alarm Center**.

Step 2 On the **Alarm Rules** tab, click **Enable Alarm Center**. In the window that slides out from the right, select one or more contact groups to manage subscription terminals 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 Center only after the Cloud Native Cluster Monitoring add-on is installed and the AOM Prometheus instance is interconnected. For details about how to enable Monitoring Center, see **Enabling Cluster Monitoring**.

Event alarms in **Table 5-2** can be reported only when Kubernetes event collection is enabled in Logging. For details, see **Collecting Kubernetes Events**.

----End

Configuring Alarm Rules

After Alarm Center is enabled for 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 In the navigation pane on the left, choose **Alarm Center**. Then, click the **Alarm Rules** tab and configure and manage alarm rules.

By default, Alarm Center generates alarm rules for containers. The rules are intended for alarms including event alarms and metric alarms for exceptions. Alarm rules are classified into several sets. You can associate an alarm rule set with multiple contact groups and enable or disable alarm items. An alarm rule set consists of multiple alarm rules. An alarm rule corresponds to the check items for a single exception. **Table 5-2** lists default alarm rules.

----End

Table 5-2 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 terminals. You can configure contact groups to manage terminals 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 terminals 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 In the navigation pane on the left, choose **Alarm Center**. Then, click the **Default Contact Groups** 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 terminal. For example, if you set **Terminal Type** to **Email** and specify a display name, the name you specified will be displayed as the alarm message sender. If no alarm message display name is specified, the sender will be **username@example.com**. The alarm message display name can be changed after a contact group is created.
- **Add Subscription Terminal**: Add one or more terminals to receive alarm messages. The terminal 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 terminal is in the **Unconfirmed** state. Send a subscription request to the terminal to verify its validity.

Step 6 Click **Request Confirmation** in the **Operation** column to send a subscription request to the terminal. After the terminal receives and confirms the request, the subscription terminal status 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.

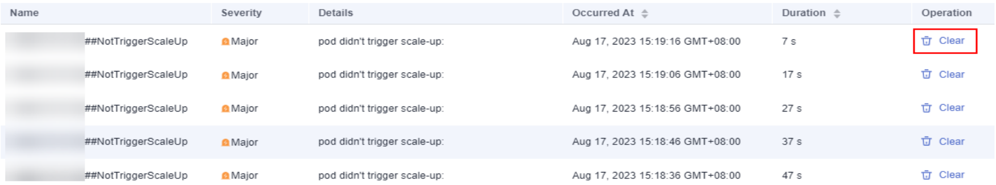
Step 2 On the cluster list page, click the name of the target cluster to go to the details page.

Step 3 In the navigation pane on the left, choose **Alarm Center**. Then, click the **Alarms** tab.

By default, all alarms to be cleared are displayed in the list. You can query alarms by alarm keyword, alarm severity, or alarm time. In addition, you can view the distribution of alarms that meet the specified criteria in different periods.

If 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 5-24 Querying alarms



Name	Severity	Details	Occurred At	Duration	Operation
[redacted]##NotTriggerScaleUp	Major	pod didn't trigger scale-up:	Aug 17, 2023 15:19:16 GMT+08:00	7 s	Clear
[redacted]##NotTriggerScaleUp	Major	pod didn't trigger scale-up:	Aug 17, 2023 15:19:06 GMT+08:00	17 s	Clear
[redacted]##NotTriggerScaleUp	Major	pod didn't trigger scale-up:	Aug 17, 2023 15:18:56 GMT+08:00	27 s	Clear
[redacted]##NotTriggerScaleUp	Major	pod didn't trigger scale-up:	Aug 17, 2023 15:18:46 GMT+08:00	37 s	Clear
[redacted]##NotTriggerScaleUp	Major	pod didn't trigger scale-up:	Aug 17, 2023 15:18:36 GMT+08:00	47 s	Clear

----End

5.3.3 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 **Enabling Cluster Monitoring**.

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

Step 2 In the navigation pane on the left, choose **Alarm Center**. Then, choose **Alarm Rules** > **Custom Alarm Rules** and click **Create Alarm Rule**.

Step 3 Configure the alarm rule parameters.

- **Rule Type**: Select **Metric alarm**.

- **Alarm Template**: If you select **No template**, you need to configure the parameters in **Rule Details**. You can also set this parameter to **Use template** to quickly define a PromQL-based alarm rule or modify an existing template.
- **Rule Details**: Configure the parameters listed in the following table.

Parameter	Description	Example Value
Rule Name	Enter the name of the alarm rule.	CoreDNS memory usage higher than 80%
(Optional) Description	Describe the alarm rule.	Check whether the memory usage of CoreDNS is higher than 80%.
Alarm Rule (PromQL)	Enter a Prometheus query statement. For details about how to compile Prometheus query statements, see **Query Examples**.	The following is an example statement for generating an alarm when the maximum memory usage of CoreDNS is higher than 80%: `(sum(container_memory_working_set_bytes{image!="", container!="POD",namespace="kube-system",container="coredns"}) BY (cluster_name, node,container, pod, namespace, cluster) / sum(container_spec_memory_limit_bytes{namespace="kube-system", container="coredns"} > 0) BY (cluster_name, node, container, pod , namespace, cluster) * 100) > 80`
Severity	Select **Critical**, **Major**, **Minor**, or **Warning**.	Critical
Duration	Select an alarm duration from the drop-down list. The default value is **1 minute**.	1 minute
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 **Collecting Logs**.

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 **Alarm Center**. Then, choose **Alarm Rules** > **Custom Alarm Rules** and click **Create Alarm Rule**.

Step 3 Configure the alarm rule parameters.

- Rule Type: Select **Event alarm**. Common events include Kubernetes events and cloud service events.
- **Rule Details**: Configure the parameters listed in the following table.

Parameter	Description	Example Value
Rule Name	Enter the name of the alarm rule.	ReplicaSet quantity change
(Optional) Description	Describe the alarm rule.	The number of ReplicaSets changes more than three times within 5 minutes.
Event Name	Enter the event name based on the actual Kubernetes event or cloud service event.	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

6 Namespaces

6.1 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**. Then click **Create Namespace** in the upper right corner.

Step 3 Configure the parameters based on **Table 6-1**.

Table 6-1 Parameters for creating a namespace

Parameter	Description
Name	Unique name of the created namespace.
Description	Description about the namespace.
Quota Management	Resource quotas can limit the 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 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
```

6.2 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.

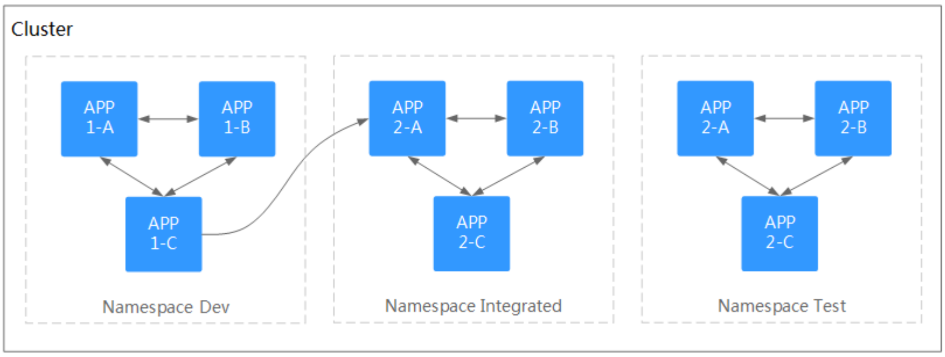
    Resources cannot be shared among different clusters. In addition, services in different environments can access each other only through load balancing.

  - 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.

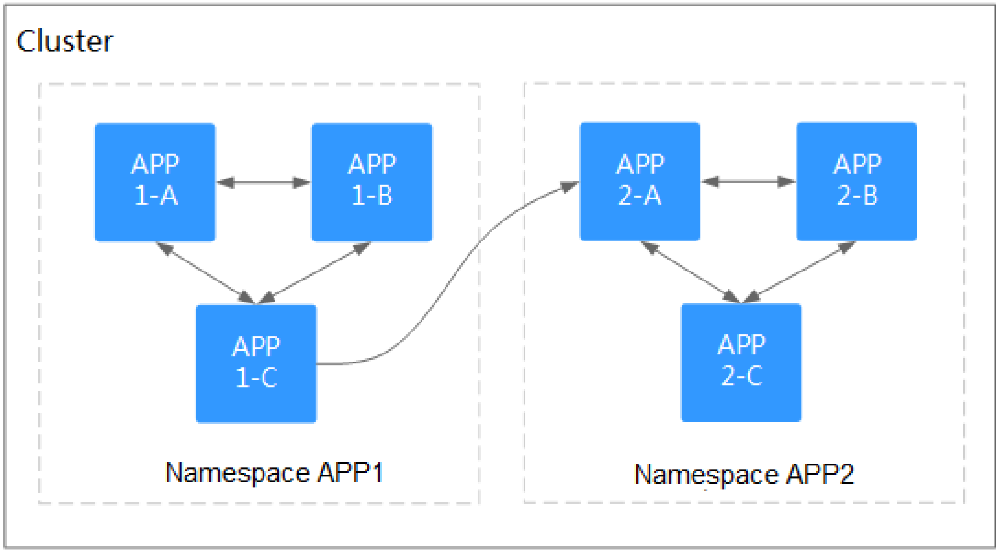
Figure 6-1 One namespace for one environment



- **Isolating namespaces by application**

  You are advised to use this method if a large number of workloads are deployed in the same environment. For example, in the following figure, different namespaces (APP1 and APP2) are created to logically manage workloads as different groups. Workloads in the same namespace access each other using the Service name, and workloads in different namespaces access each other using the Service name or namespace name.

Figure 6-2 Grouping workloads into different namespaces



Managing Namespace Labels

Step 1 Log in to the CCE console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Namespaces**.

Step 2 Locate the row containing the target namespace and choose **More** > **Manage Label** in the **Operation** column.

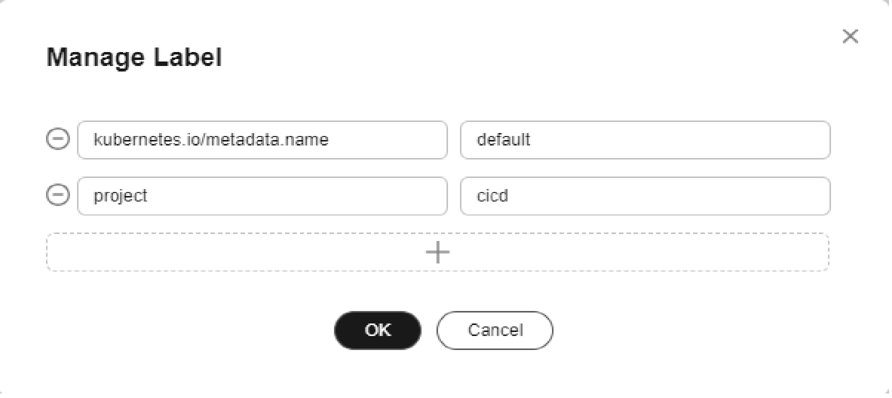
Step 3 In the dialog box that is displayed, the existing labels of the namespace are displayed. Modify the labels as needed.

- 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 6-3 Adding or deleting a namespace label


Manage Label

kubernetes.io/metadata.name	default
project	cicd

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 In the navigation pane on the left, choose **Namespaces**. Locate the target namespace and choose **More** > **Delete** in the **Operation** column.

Follow the prompts to delete the namespace. The default namespaces cannot be deleted.

----End

6.3 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

7 ConfigMaps and Secrets

7.1 Creating a ConfigMap

Scenario

A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. After creating ConfigMaps, you can use them as files or environment variables in a containerized workload.

ConfigMaps allow you to decouple configuration files from container images to enhance the portability of workloads.

Benefits of ConfigMaps:

- Manage configurations of different environments and services.
- Deploy workloads in different environments. Multiple versions are supported for configuration files so that you can update and roll back workloads easily.
- Quickly import configurations in the form of files to containers.

Notes and Constraints

- The size of a ConfigMap resource file cannot exceed 1 MB.
- ConfigMaps cannot be used in **static pods**.

Procedure

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

Step 2 Choose **ConfigMaps and Secrets** in the navigation pane and click **Create ConfigMap** in the upper right corner.

Step 3 Configure parameters.

Table 7-1 Parameters for creating a ConfigMap

Parameter	Description
Name	Name of the ConfigMap you create, which must be unique in a namespace.
Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of the ConfigMap.
Data	Data of a ConfigMap, in the key-value pair format. Click + to add data. The value can be in string, JSON, or YAML format.
Label	Label of the ConfigMap. Enter a key-value pair and click **Confirm**.

Step 4 Click **OK**.

The new ConfigMap is displayed in the ConfigMap list.

----End

Creating a ConfigMap Using kubectl

Step 1 Use kubectl to access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **cce-configmap.yaml** and edit it.

vi cce-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cce-configmap
data:
  SPECIAL_LEVEL: Hello
  SPECIAL_TYPE: CCE
```

Table 7-2 Key parameters

Parameter	Description
apiVersion	The value is fixed at **v1**.
kind	The value is fixed at **ConfigMap**.
metadata.name	ConfigMap name, which can be customized.
data	ConfigMap data. The value must be key-value pairs.

Step 3 Run the following commands to create a ConfigMap.

kubectl create -f cce-configmap.yaml

Run the following commands to view the created ConfigMap:

kubectl get cm

```
NAME            DATA        AGE
cce-configmap     3         7m
```

----End

Related Operations

After creating a ConfigMap, you can update or delete it as described in **Table 7-3**.

Table 7-3 Related operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target ConfigMap resides to edit its YAML file.
Updating a ConfigMap	1. Select the name of the ConfigMap to be updated and click **Update**. 2. Modify the secret data. For more information, see **Table 7-1**. 3. Click **OK**.
Deleting a ConfigMap	Select the configuration you want to delete and click **Delete**. Follow the prompts to delete the ConfigMap.

7.2 Using a ConfigMap

You can use a ConfigMap to set environment variables or command line parameters, or mount a ConfigMap as a data volume.

- **Configuring Environment Variables for a Workload**
- **Configuring Command Line Parameters**
- **Mounting a ConfigMap Volume to a Workload**

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, it must be in the same cluster and namespace as the workload.
- When a ConfigMap mounted as a data volume is updated, Kubernetes updates the data in the data volume at the same time.

  If a ConfigMap is mounted as a data volume using **subPath**, data in the data volume cannot be automatically updated when the ConfigMap is updated.
- When a ConfigMap is used as an environment variable, data in the environment variable cannot be automatically updated when the ConfigMap is updated. To update the data, restart the pod.

Configuring Environment Variables for a Workload

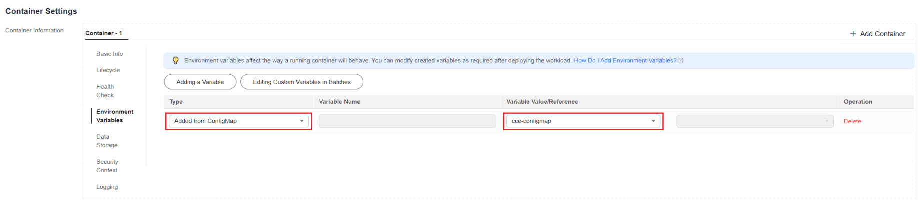
Using the console

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

Step 2 In the navigation pane on the left, choose **Workloads**. In the upper right corner of the displayed page, click **Create Workload**.

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.

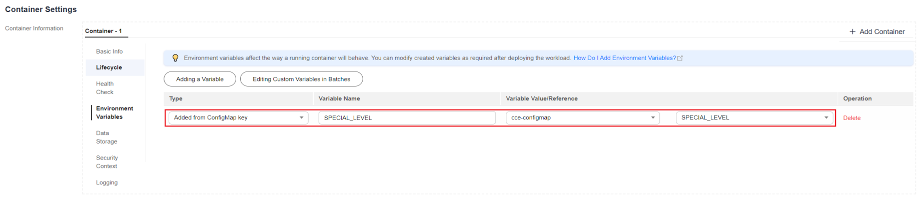


Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do I Add Environment Variables?

Type	Variable Name	Variable Value/Reference		Operation
Added from ConfigMap		cce-configmap		Delete

- **Added from ConfigMap key**: Import a key in a ConfigMap as the value of an environment variable.
  - **Variable Name**: Name of the environment variable. 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.



Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do I Add Environment Variables?

Type	Variable Name	Variable Value/Reference		Operation
Added from ConfigMap key	SPECIAL_LEVEL	cce-configmap	SPECIAL_LEVEL	Delete

Step 3 Configure other workload parameters and click **Create Workload**.

When the workload runs normally, **log in to the container** and run the following statement to check whether the ConfigMap has been set as an environment variable for the workload:

```
printenv SPECIAL_LEVEL
```

The following is an example output:

```
Hello
```

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

The content of the YAML file is as follows:

- **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 an environment variable for the workload.
        - name: SPECIAL_LEVEL           # Name of the environment variable for 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).

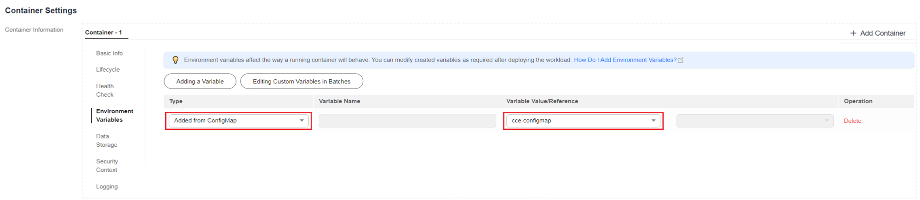
Using the console

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

Step 2 In the navigation pane on the left, choose **Workloads**. In the upper right corner of the displayed page, click **Create Workload**.

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.



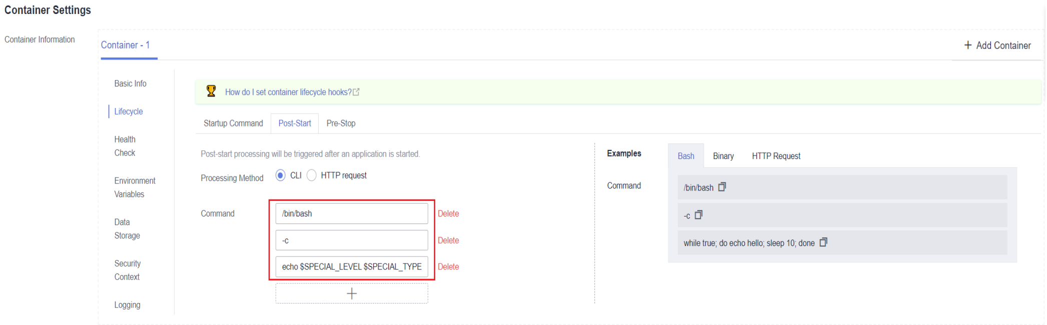
Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do I Add Environment Variables?

Type	Variable Name	Variable Value/Reference		Operation
Added from ConfigMap		cce-configmap		Delete

Step 3 Click **Lifecycle** in the **Container Settings** area, click the **Post-Start** tab on the right, and configure 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
```



How do I set container lifecycle hooks?

Post-start processing will be triggered after an application is started.

Processing Method ● CLI ○ HTTP request

Command

```
/bin/bash
-c
echo $SPECIAL_LEVEL $SPECIAL_TYPE
```

Command

```
/bin/bash
-c
while true; do echo hello; sleep 10; done
```

Step 4 Configure other workload parameters and click **Create Workload**.

When the workload runs normally, log in to the container and run the following statement to check whether the ConfigMap has been set as an environment variable for the workload:

```
cat /usr/share/nginx/html/index.html
```

The following is an example output:

```
Hello CCE
```

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

In this 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 Wait until the workload runs normally. The following content will be added to the **/usr/share/nginx/html/index.html** file of 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 Volume to a Workload

A ConfigMap can be mounted as a volume to a specified container path. The workload code is separated from the configuration files. ConfigMap volumes are used to store workload configuration parameters. You need to create ConfigMaps in advance by following the instructions in **Creating a ConfigMap**.

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 on the left, choose **Workloads**. In the upper right corner of the displayed page, click **Create Workload**.

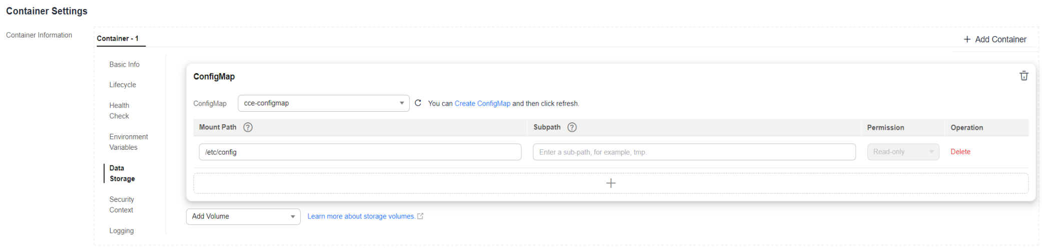
When creating a workload, click **Data Storage** in the **Container Settings** area. Click **Add Volume** and select **ConfigMap** from the drop-down list.

Step 3 Configure the parameters for mounting a ConfigMap volume based on **Table 7-4**.

Table 7-4 Parameters for mounting a ConfigMap volume

Parameter	Description
ConfigMap	Select the desired ConfigMap. A ConfigMap must be created beforehand. For details, see **Creating a ConfigMap**.
Mount Path	Enter a mount 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 indicates the container path that the volume will be mounted to. 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, or the files will be replaced, which will lead to a container startup 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 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 using a subpath will not be updated along with the ConfigMap.
Permission	Read-only permission. The data volume in the path is read-only.

Figure 7-1 Mounting a ConfigMap volume



Container Settings

ConfigMap: cce-configmap

You can Create ConfigMap and then click refresh.

Mount Path	Subpath	Permission	Operation
/etc/config	Enter a sub-path, for example, tmp.	Read-only	Delete

Step 4 Click **Create Workload**.

When the workload runs normally, the **SPECIAL_LEVEL** and **SPECIAL_TYPE** files will be generated in the **/etc/config** directory in this example. The content of file **SPECIAL_LEVEL** is **Hello**, and that of **SPECIAL_TYPE** is **CCE**.

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 **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-configmap.yaml** and edit it.

vi nginx-configmap.yaml

In this 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 the volume 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 When the workload runs normally, the **SPECIAL_LEVEL** and **SPECIAL_TYPE** files will be generated in the **/etc/config** directory in this example. The content of file **SPECIAL_LEVEL** is **Hello**, and that of **SPECIAL_TYPE** is **CCE**.

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

7.3 Creating a Secret

Scenario

A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined. After creating secrets, you can use them as files or environment variables in a containerized workload.

Notes and Constraints

Secrets cannot be used in **static pods**.

Procedure

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

Step 2 Choose **ConfigMaps and Secrets** in the navigation pane, click the **Secrets** tab, and click **Create Secret** in the upper right corner.

Step 3 Configure parameters.

Table 7-5 Parameters for creating a secret

Parameter	Description
Name	Name of the secret you create, which must be unique.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of a secret.
Type	Type of the secret you create. ● Opaque: common secret. ● kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository. ● **kubernetes.io/tls**: Kubernetes TLS secret, which is used to store the certificate required by layer-7 load balancing Services. For details about examples of the kubernetes.io/tls secret and its description, see **TLS secrets**. ● **IngressTLS**: TLS secret provided by CCE to store the certificate required by layer-7 load balancing Services. ● Other: another type of secret, which is specified manually.

Parameter	Description
Secret Data	Workload secret data can be used in containers. ● If **Secret Type** is **Opaque**, click +. In the dialog box displayed, enter a key-value pair and select **Auto Base64 Encoding**. ● If **Secret Type** is **kubernetes.io/dockerconfigjson**, enter the account and password for logging in to the private image repository. ● If **Secret Type** is **kubernetes.io/tls** or **IngressTLS**, upload the certificate file and private key file. **NOTE** – A certificate is a self-signed or CA-signed credential used for identity authentication. – A certificate request is a request for a signature with a private key.
Secret Label	Label of the secret. Enter a key-value pair and click **Confirm**.

Step 4 Click **OK**.

The new secret is displayed in the key list.

----End

Secret Resource File Configuration Example

This section describes configuration examples of secret resource description files.

- Opaque type

  The **secret.yaml** file is defined as shown below. The **data** field is filled in as a key-value pair, and the **value** field must be encoded using Base64. For details about the Base64 encoding method, see **Base64 Encoding**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be encoded using Base64.
type: Opaque
```

- kubernetes.io/dockerconfigjson type

  The **secret.yaml** file is defined as shown below. The value of **.dockerconfigjson** must be encoded using Base64. For details, see **Base64 Encoding**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          #Secret name
  namespace: default      #Namespace. The default value is default.
data:
  .dockerconfigjson: eyJh*****    # Content encoded using Base64.
type: kubernetes.io/dockerconfigjson
```

To obtain the **.dockerconfigjson** content, perform the following steps:

a. Obtain the following login information of the image repository.

   - Image repository address: The section uses *address* as an example. Replace it with the actual address.
   - Username: The section uses *username* as an example. Replace it with the actual username.
   - Password: The section uses *password* as an example. Replace it with the actual password.

b. Use Base64 to encode the key-value pair *username:password* and fill the encoded content in 3.

```
echo -n "username:password" | base64
```

Command output:

```
dXNlcm5hbWU6cGFzc3dvcmQ=
```

c. Use Base64 to encode the following JSON content:

```
echo -n '{"auths":{"address":
{"username":"username","password":"password","auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}'
| base64
```

Command output:

```
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 access the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create and edit the Base64-encoded **cce-secret.yaml** file.

```
# echo -n "content to be encoded" | base64
******
```

vi cce-secret.yaml

The following YAML file uses the Opaque type as an example. For details about other types, see **Secret Resource File Configuration Example**.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  <your_key>: <your_value>  # Enter a key-value pair. The value must be encoded using Base64.
```

Step 3 Create a secret.

kubectl create -f cce-secret.yaml

You can query the secret after creation.

kubectl get secret -n default

----End

Related Operations

After creating a secret, you can update or delete it as described in **Table 7-6**.

NOTE

The secret list contains system secret resources that can be queried only. The system secret resources cannot be updated or deleted.

Table 7-6 Related Operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target secret resides to edit its YAML file.
Updating a secret	1. Select the name of the secret to be updated and click **Update**. 2. Modify the secret data. For more information, see **Table 7-5**. 3. Click **OK**.
Deleting a secret	Select the secret you want to delete and click **Delete**. Follow the prompts to delete the secret.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete** above the secret list. 3. Follow the prompts to delete the secrets.

Base64 Encoding

To Base64-encode a string, run the **echo -n *content to be encoded* | base64** command. The following is an example:

```
root@ubuntu:~# echo -n "content to be encoded" | base64
******
```

7.4 Using a Secret

After secrets are created, they can be mounted as data volumes or exposed as environment variables for a container.

NOTICE

Do not perform any operation on the following secrets (for details, see **Cluster Secrets**):

- Secrets under kube-system
- default-secret and paas.elb in any namespace default-secret is used to pull private images from SWR, and paas.elb is used to connect the services in the namespace to ELB.

- **Configuring Environment Variables for a Workload**
- **Configuring a Data Volume for 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 encoded using Base64.
  password: ******  #The value must be encoded using Base64.
```

NOTICE

- When a secret is used in a pod, it must be in the same cluster and namespace as the pod.
- When a secret mounted as a data volume is updated, Kubernetes updates the data in the data volume at the same time.

  However, when a secret mounted using **subPath** is updated, Kubernetes cannot automatically update the data in the data volume.

Configuring Environment Variables for a Workload

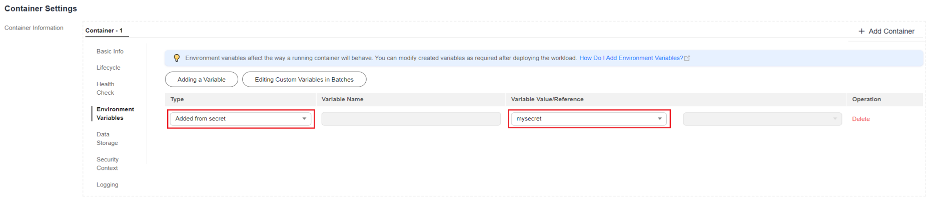
Using the console

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

Step 2 In the navigation pane on the left, choose **Workloads**. In the upper right corner of the displayed page, click **Create Workload**.

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.

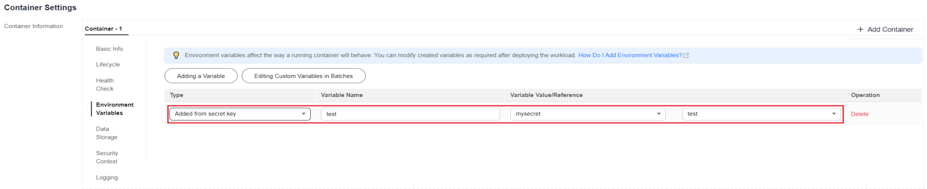


Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do I Add Environment Variables?

Type	Variable Name	Variable Value/Reference		Operation
Added from secret		mysecret		Delete

- **Added from secret key**: Import the value of a key in a secret as the value of an environment variable.
  - **Variable Name**: Name of the environment variable. 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.



Environment variables affect the way a running container will behave. You can modify created variables as required after deploying the workload. How Do I Add Environment Variables?

Type	Variable Name	Variable Value/Reference		Operation
Added from secret key	test	mysecret	test	Delete

Step 3 Configure other workload parameters and click **Create Workload**.

When the workload runs normally, **log in to the container** and run the following statement to check whether the secret has been set as an environment variable for 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 for the workload.

----End

Using kubectl

Step 1 Use kubectl to connect to the cluster. For details, see **Connecting to a Cluster Using kubectl**.

Step 2 Create a file named **nginx-secret.yaml** and edit it.

vi nginx-secret.yaml

The content of the YAML file is as follows:

- **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 an environment variable for the workload.
        - name: SECRET_USERNAME          # Name of the environment variable for 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 for the workload.

----End

Configuring a Data Volume for a Workload

You can mount a secret as a volume to the specified container path. The content of a secret is user-defined. You need to create a secret in advance. For details, see **Creating a Secret**.

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 on the left, click **Workloads**. Then 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.

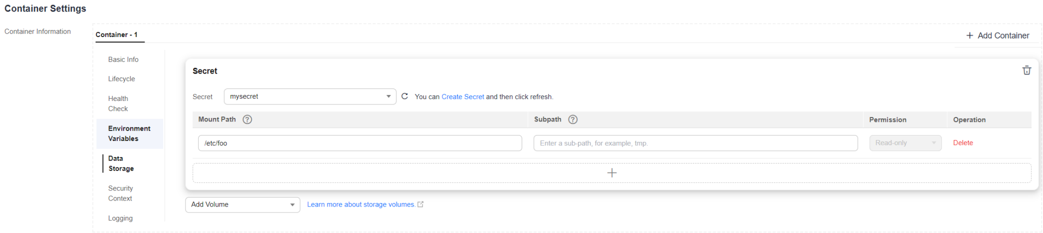
Step 3 Configure parameters for mounting a secret volume based on **Table 7-7**.

Table 7-7 Parameters for mounting a secret volume

Parameter	Description
Secret	Select the desired secret. A secret must be created beforehand. For details, see **Creating a Secret**.
Mount Path	Enter a mount 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 indicates the container path that the volume will be mounted to. 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, or the files will be replaced, which will lead to a container startup 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.

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 secret. If the subpath is a key-value pair that does not exist, the data import does not take effect. ● The data imported using a subpath will not be updated along with the secret.
Permission	Read-only permission. The data volume in the path is read-only.

Figure 7-2 Mounting a secret volume



Secret: mysecret

You can Create Secret and then click refresh.

Mount Path	Subpath	Permission	Operation
/etc/foo	Enter a sub-path, for example, tmp.	Read-only	Delete

Step 4 Click **Create Workload**.

When the workload runs normally, the **username** and **password** files will be generated in the **/etc/foo** directory in this example. The file content is the 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 **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 the secret volume 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. In the following example, if the **password** key 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, or the volume will not be 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 the secret volume 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 Wait until the workload runs normally. The **username** and **password** files will be generated in the **/etc/foo** directory.

1. Run the following command to view the created pod:

   ```
   kubectl get pod | grep nginx-secret
   ```

   Expected output:

   ```
   nginx-secret-***   1/1     Running   0              2m18s
   ```

2. Run the following command to view the **username** or **password** file in the pod:

   ```
   kubectl exec nginx-secret-*** -- cat /etc/foo/username
   ```

   The expected output is the same as the content in the secret.

----End

7.5 Cluster Secrets

By default, CCE creates the following secrets in each namespace:

- default-secret
- paas.elb
- default-token-*xxxxx* (*xxxxx* is a random number.)

The functions of these secrets are described as follows.

default-secret

The type of **default-secret** is **kubernetes.io/dockerconfigjson**. The data is the credential for logging in to the SWR image repository and is used to pull images from SWR. To pull an image from SWR when creating a workload on CCE, set **imagePullSecrets** to **default-secret**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:alpine
    name: container-0
    resources:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
  imagePullSecrets:
  - name: default-secret
```

The data of **default-secret** is updated periodically, and the current data will expire after a certain period of time. You can run the **describe** command to view the expiration time in default-secret.

NOTICE

Use default-secret directly instead of copying the secret content to create a new one. The credential in the copied secret will expire and the image cannot be pulled.

```
$ kubectl describe secret default-secret
Name:         default-secret
Namespace:    default
Labels:       secret-generated-by=cce
Annotations:  temporary-ak-sk-expires-at: 2021-11-26 20:55:31.380909 +0000 UTC

Type:  kubernetes.io/dockerconfigjson

Data
====
.dockerconfigjson:  347 bytes
```

paas.elb

The **paas.elb** data stores a temporary AK/SK that is used when a node is created or a load balancer is automatically created. The **paas.elb** data is updated periodically and has a specific time limit before it expires.

In practice, you will not directly use **paas.elb**. Do not delete it, as doing so will result in the failure of creating a node or load balancer. will fail.

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>
```

8 Auto Scaling

8.1 Scaling a Workload

8.1.1 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).

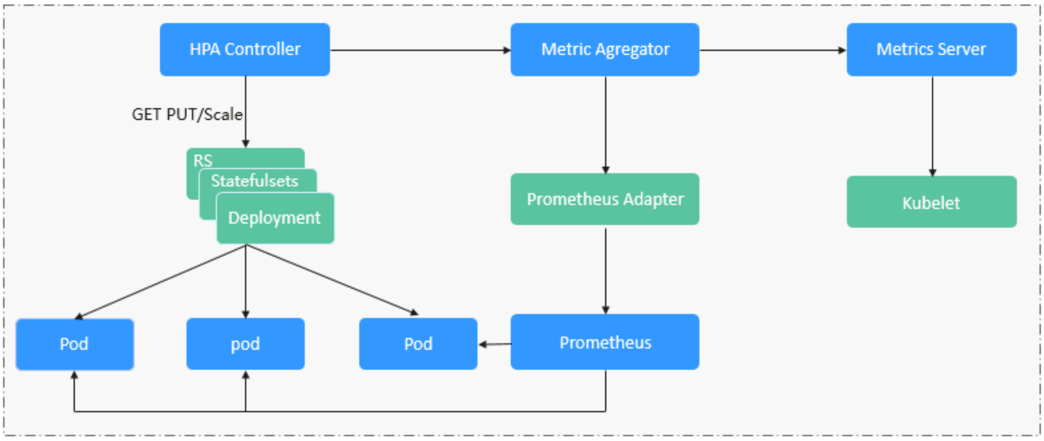
A prerequisite for auto scaling is that your container running data can be collected, such as the numbers of cluster nodes and pods, and CPU and memory usages of containers. Kubernetes does not have built-in monitoring capabilities, but you can use extensions like **Prometheus** and **Metrics Server** to monitor and collect data.

- **Prometheus** is an open-source monitoring and alarming framework that can collect multiple types of metrics. Prometheus has been a standard monitoring solution of Kubernetes.
- **Metrics Server** is a cluster-wide aggregator of resource utilization data. Metrics Server collects metrics from the Summary API exposed by kubelet. These metrics are set for core Kubernetes resources, such as pods, nodes, containers, and Services. Metrics Server provides a set of standard APIs for external systems to collect these metrics.

Horizontal Pod Autoscaler (HPA) can work with Metrics Server to implement auto scaling based on CPU and memory usages. It can also work with Prometheus to implement auto scaling based on custom metrics.

Figure 8-1 shows how HPA works.

Figure 8-1 HPA working process



Two core modules of HPA:

- Data Source Monitoring

  The community provided only CPU- and memory-based HPA at the early stage. With the population of Kubernetes and Prometheus, developers need more custom metrics or monitoring information at the access layer for their own applications, for example, the QPS of the load balancer and the number of online users of the website. In response, the community defines a set of standard metric APIs to provide services externally through these aggregated APIs.

  - **metrics.k8s.io** provides monitoring metrics related to the CPU and memory of pods and nodes.
  - **custom.metrics.k8s.io** provides custom monitoring metrics related to Kubernetes objects.
  - **external.metrics.k8s.io** provides metrics that come from external systems and are irrelevant to any Kubernetes resource metrics.

- Scaling Decision-Making Algorithms

  The HPA controller calculates the scaling ratio based on the current metric values and desired metric values using the following formula:

  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.

8.1.2 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:

- **Kubernetes Metrics Server**: provides basic resource usage metrics, such as container CPU and memory usage.
- **Creating an HPA Policy Using Custom Metrics**: Custom metrics need to be aggregated to the Kubernetes API server for auto scaling.

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 8-1 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.

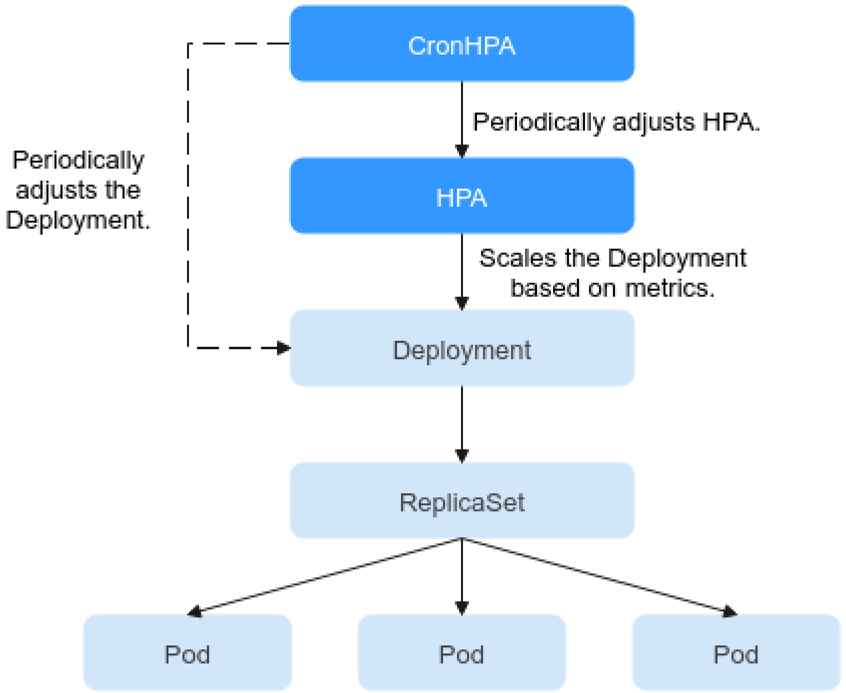
Parameter	Description
Scaling Behavior	● **Default**: Scale workloads using the Kubernetes default behavior. For details, see **Default Behavior**. ● **Custom**: Scale workloads using custom policies such as stabilization window, steps, and priorities. Unspecified parameters use the values recommended 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 only supported in clusters v1.15 or later.**

Step 5 Click **Create**.

----End

8.1.3 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
Periodically adjusts HPA.
HPA
Scales the Deployment based on metrics.
Periodically adjusts the Deployment.
Deployment
ReplicaSet
Pod
Pod
Pod

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.

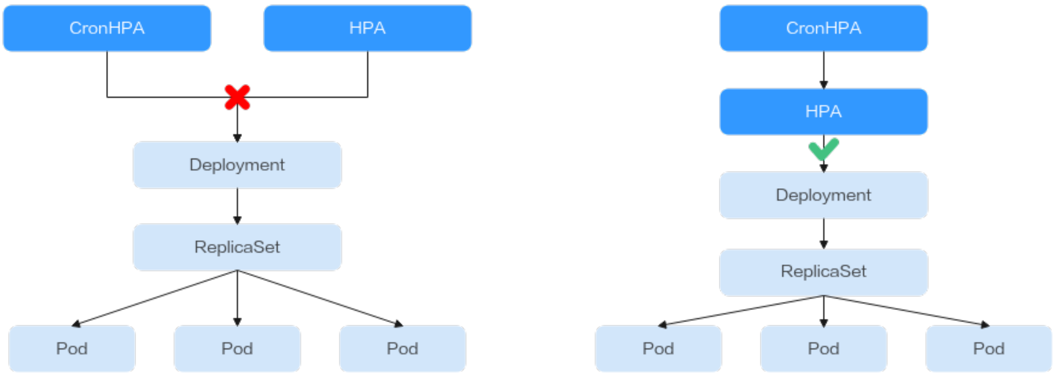
Prerequisites

The **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 8-2 CronHPA scaling scenarios

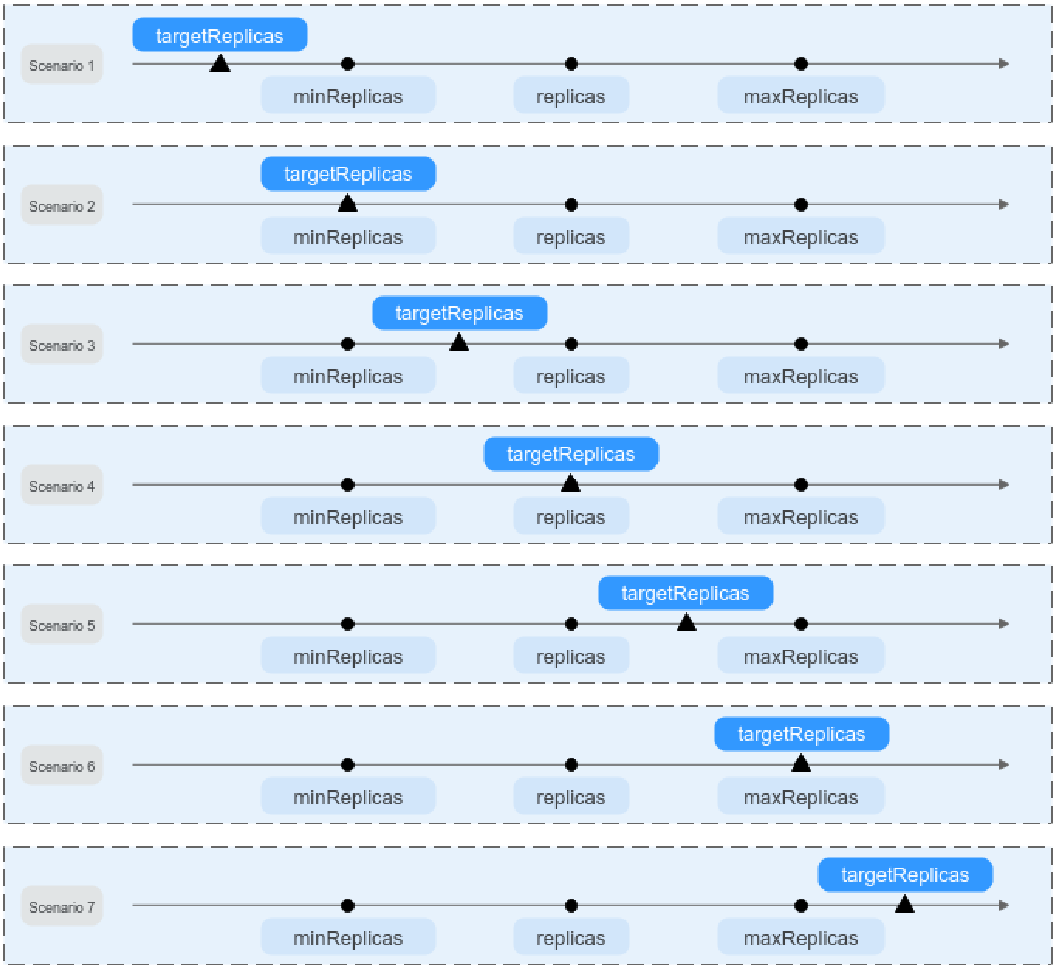


Figure 8-2 shows possible scaling scenarios. The following examples detail how CronHPA modifies the number of pods in HPAs.

Table 8-2 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**.

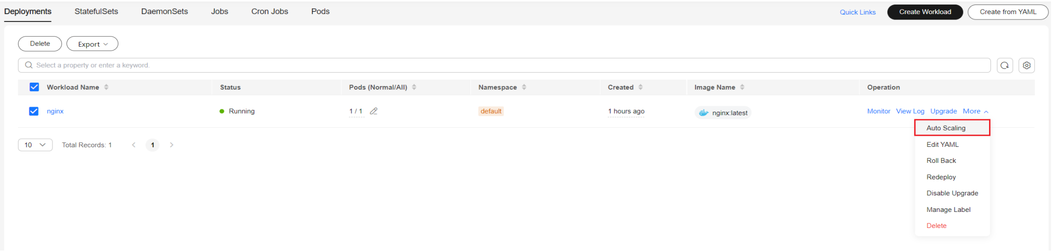
Scenario	Scenario Description	CronHPA (targetReplicas)	Deployment (replicas)	HPA (minReplicas/maxReplicas)	Result	Operation Description
7	minReplicas ≤ replicas ≤ maxReplicas < **targetReplicas**	11	5	5/10	HPA: 11/11 Deployments: 11	When the value of **targetReplicas** is greater than that of **maxReplicas**: ● 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 8-3 Scaling a workload



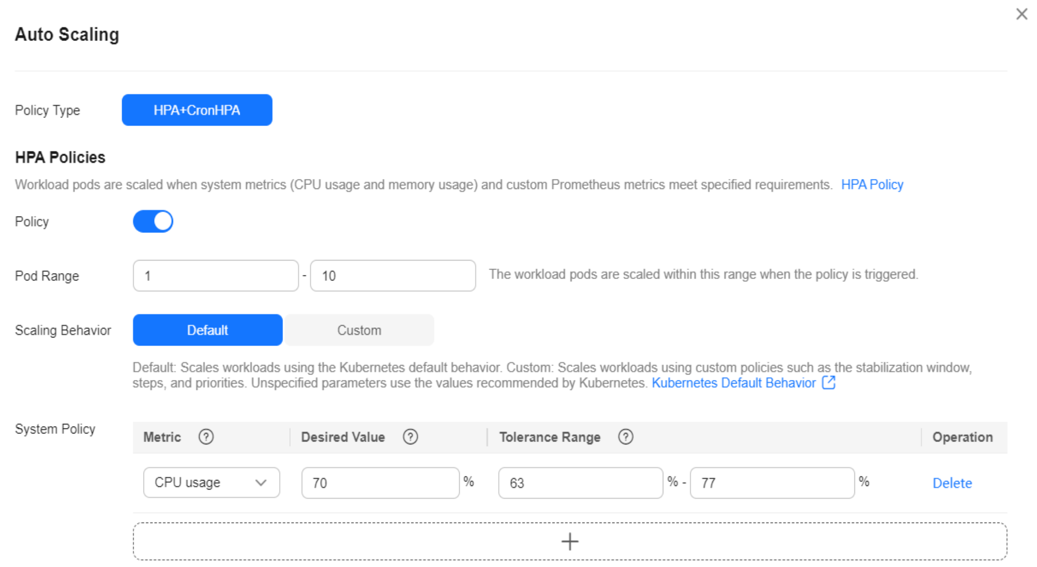
☑ Workload Name	Status	Pods (Normal/All)	Namespace	Created	Image Name	Operation
☑ nginx	● Running	1 / 1	default	1 hours ago	nginx:latest	Monitor View Log Upgrade More

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 **HPA Policies**.

Figure 8-4 Enabling the HPA policy



Auto Scaling

Policy Type: HPA+CronHPA

HPA Policies

Workload pods are scaled when system metrics (CPU usage and memory usage) and custom Prometheus metrics meet specified requirements. HPA Policy

Policy: ●

Pod Range: 1 - 10 The workload pods are scaled within this range when the policy is triggered.

Scaling Behavior: Default | Custom

Default: Scales workloads using the Kubernetes default behavior. Custom: Scales workloads using custom policies such as the stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. Kubernetes Default Behavior

System Policy

Metric	Desired Value	Tolerance Range	Operation
CPU usage	70 %	63 % - 77 %	Delete

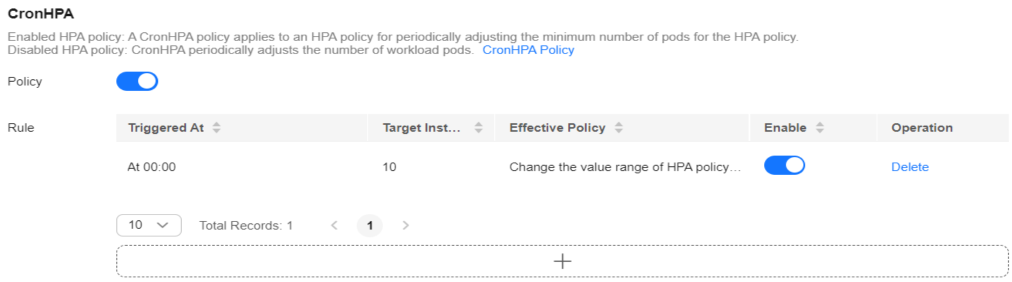
Table 8-3 HPA policy parameters

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 8-5 Enabling the CronHPA policy



CronHPA

Enabled HPA policy: A CronHPA policy applies to an HPA policy for periodically adjusting the minimum number of pods for the HPA policy.
Disabled HPA policy: CronHPA periodically adjusts the number of workload pods. CronHPA Policy

Policy ●

Rule

Triggered At	Target Inst...	Effective Policy	Enable	Operation
At 00:00	10	Change the value range of HPA policy...	●	Delete

10 Total Records: 1 < 1 >

+

Table 8-4 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 8-2**.
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 8-5 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

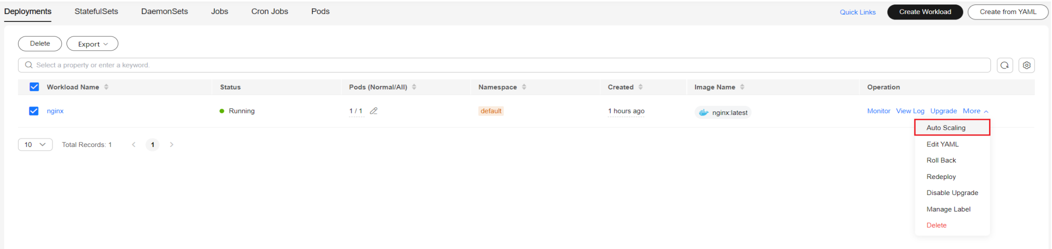
CronHPA adjusts the associated Deployments separately to periodically adjust the number of Deployment pods. The method is as follows:

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 8-6 Scaling a workload



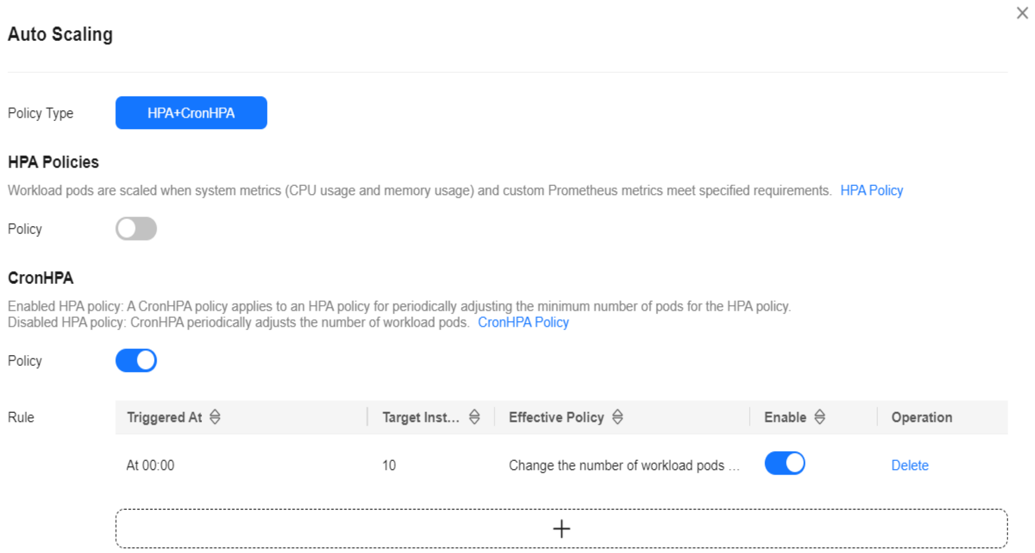
Workload Name	Status	Pods (Normal/All)	Namespace	Created	Image Name	Operation
nginx	Running	1 / 1	default	1 hours ago	nginx:latest	Monitor View Log Upgrade More

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 8-7 Using CronHPA to adjust the number of workload pods



Policy Type HPA+CronHPA

HPA Policies

Workload pods are scaled when system metrics (CPU usage and memory usage) and custom Prometheus metrics meet specified requirements. HPA Policy

Policy

CronHPA

Enabled HPA policy: A CronHPA policy applies to an HPA policy for periodically adjusting the minimum number of pods for the HPA policy.
Disabled HPA policy: CronHPA periodically adjusts the number of workload pods. CronHPA Policy

Policy

Rule

Triggered At	Target Inst...	Effective Policy	Enable	Operation
At 00:00	10	Change the number of workload pods ...		Delete

Table 8-6 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
```

8.1.4 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 8-7 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 8-1**.

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

9 Add-ons

9.1 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 9-1 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 9-2 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 9-3 describes common plugins. For details, see **Plugins**.
	Example:
	`{`
	`    "servers": [`
	`        {`
	`         "plugins": [`
	`            {`
	`                "name": "bind",`
	`                "parameters": "{$POD_IP}"`
	`            },`
	`            {`
	`                "name": "cache",`
	`                "parameters": 30`
	`            },`

Parameter	Description

```
                {
                    "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 9-3 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**.
errors	Errors are logged to stdout. For details, see **errors**.

Plugin Name	Description
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 9-4 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

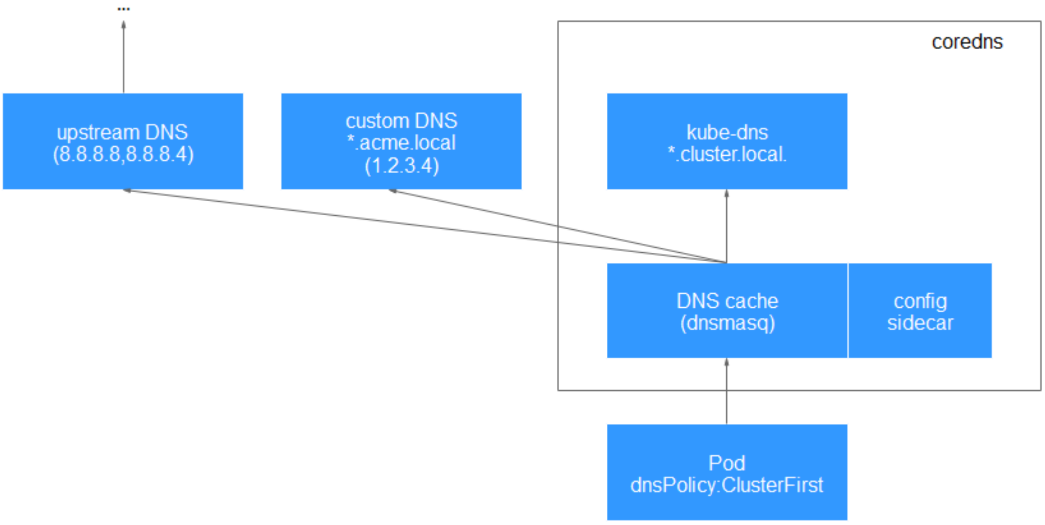
Without stub domain configurations: Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

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 9-1 Routing



9.2 CCE Container Storage (Everest)

Introduction

Everest is a cloud native container storage add-on that enables Kubernetes clusters to access cloud storage services through CSI.

NOTE

In clusters of v1.27.5-r0, v1.28.3-r0, and later versions, this add-on is automatically configured by the system and does not need to be manually installed or updated.

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 **CCE Container Storage (Everest)** on the right and click **Edit**.

Step 2 On the **Edit Add-on** page, modify the specifications.

Table 9-5 Everest specifications

Parameter	Description
Instances	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.

Parameter	Description
Containers	CPU and memory quotas of the containers for running the add-on. The add-on contains the everest-csi-controller and everest-csi-driver components. For details, see **Components**.

Step 3 Modify the parameters.

Table 9-6 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.
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 GiB, it can be overcommitted to 180 GiB.
project_id	ID of the project to which a cluster belongs

NOTE

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 Click **Install**.

----End

Components

Table 9-7 Everest components

Component	Description	Resource Type
everest-csi-controller	Used to create, delete, snapshot, expand, attach, and detach storage volumes.	Deployment
everest-csi-driver	Used to mount and unmount PVs and resize file systems.	DaemonSet

9.3 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 **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 9-8 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 9-9 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

9.4 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 Monitoring Center so that you can view monitoring data and configure alarms in Monitoring Center.

Open source community: **https://github.com/prometheus/prometheus**

Permissions

The node-exporter component of the Cloud Native Cluster Monitoring add-on needs to read the Docker info data from the **/var/run/docker.sock** directory on the host for monitoring the Docker disk space.

The following permission is required for running node-exporter:

- cap_dac_override: reads the Docker info data.

Installing the Add-on

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.

- **Interconnecting with AOM**: Prometheus data will be reported to AOM. If this option is enabled, you need to select an AOM instance. Basic metrics are free, but custom metrics are billed based on the standard pricing of AOM. For details, see **AOM Pricing Details**. To interconnect with AOM, you must have certain permissions. Only Huawei Cloud accounts, HUAWEI IDs, and IAM users in the **admin** user group can perform this operation.
- **User-defined indicator HPA**: Application metrics are automatically collected in the form of service discovery. If this option is enabled, you need to add related configurations to the target application. For details, see **Creating an HPA Policy Using Custom Metrics**.

Step 4 Click **Install**.

After the add-on is installed, you may need to perform the following operations:

If you want to use custom metrics to create an HPA policy, you need to aggregate the custom metrics collected by Prometheus to the API server. For details, see **Creating an HPA Policy Using Custom Metrics**.

----End

Components

All Kubernetes resources created during kube-prometheus-stack add-on installation are created in the namespace named **monitoring**.

Table 9-10 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-lightweight (workload name: prometheus-lightweight)	A Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.	StatefulSet

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. **NOTE** If the components run in multiple pods, only metrics in one pod are collected.	Deployment
nodeExporter (workload name: node-exporter)	Deployed on each node to collect node monitoring data.	Pod
adapter (workload name: custom-metrics-apiserver)	Aggregates custom metrics to the native Kubernetes API Server. This is closely related to custom metric HPA. The adapter component needs to be installed only when custom metric HPA is enabled.	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 Metrics API, create the following APIService object:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl create -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If information similar to the following is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                                    CPU(cores)   MEMORY(bytes)
......
```

```
custom-metrics-apiserver-d4f556ff9-l2j2m               38m          44Mi
......
```

NOTICE

To uninstall the add-on, run the following kubectl command and delete the APIService object first or the add-on cannot be installed due to residual APIService resources.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Creating an HPA Policy Using Custom Metrics

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 **ConfigMaps and Secrets**. Then click **Create from YAML**.

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 collection rule:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: user-adapter-config
  namespace: monitoring
data:
  config.yaml: |-
    rules:
    - seriesQuery: 'container_network_receive_bytes_total{namespace!="",pod!=""}' # Original metrics
required for scale-out (metrics from kubelet)
      seriesFilters: []
      resources:
        overrides:  # Specifies pod and namespace resources.
          namespace:
            resource: namespace
          pod:
            resource: pod
      name:
        matches: container_(.*)_total
        as: "pod_${1}_per_second"  # Metric alias
      metricsQuery: sum(rate(<<.Series>>{<<.LabelMatchers>>}[30s])) by (<<.GroupBy>>) # Metric change
rate of all containers for a workload within 30s
```

NOTE

In the preceding example, the aggregation time is 30s. If this time is set to a value less than 15s (a collection period), the metrics may be inaccurate.

Step 3 In the navigation pane, choose **Add-ons**. Locate the Cloud Native Cluster Monitoring add-on, click **Edit**, and enable custom metric HPA.

Click **Install**.

NOTE

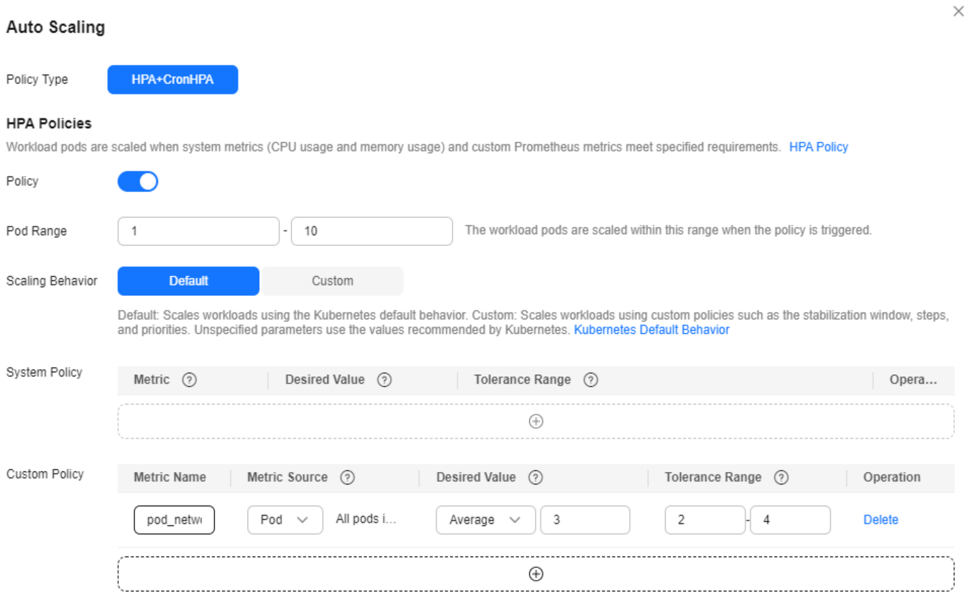
You need to create collection rules described in **Step 2** and then enable the custom metric HPA to trigger add-on redeployment for custom metric collection to take effect.

Step 4 In the navigation pane on the left, choose **Workloads**. Locate the workload for which you want to create an HPA policy and click **Auto Scaling**. In the **Custom Policy** area, you can create an auto scaling policy based on the metrics in **Step 2**.

NOTE

After a workload is created, create an HPA policy unless all pods for that workload are ready and the metrics of the first collection period are collected.

Figure 9-2 Creating an HPA policy



Auto Scaling

Policy Type: HPA+CronHPA

HPA Policies

Workload pods are scaled when system metrics (CPU usage and memory usage) and custom Prometheus metrics meet specified requirements. HPA Policy

Policy: ●

Pod Range: 1 - 10 The workload pods are scaled within this range when the policy is triggered.

Scaling Behavior: Default | Custom

Default: Scales workloads using the Kubernetes default behavior. Custom: Scales workloads using custom policies such as the stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. Kubernetes Default Behavior

System Policy

Metric	Desired Value	Tolerance Range	Opera...

Custom Policy

Metric Name	Metric Source	Desired Value	Tolerance Range	Operation
pod_netw	Pod All pods i...	Average 3	2 - 4	Delete

----End

9.5 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 all abnormal Kubernetes events and some normal Kubernetes events to AOM. For details about how to collect logs, see **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, up to 10,000 single-line logs can be collected per second, and up to 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 9-11 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 container components, whose specifications can be adjusted as required: ● fluent-bit: indicates the 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 9-12 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

9.6 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

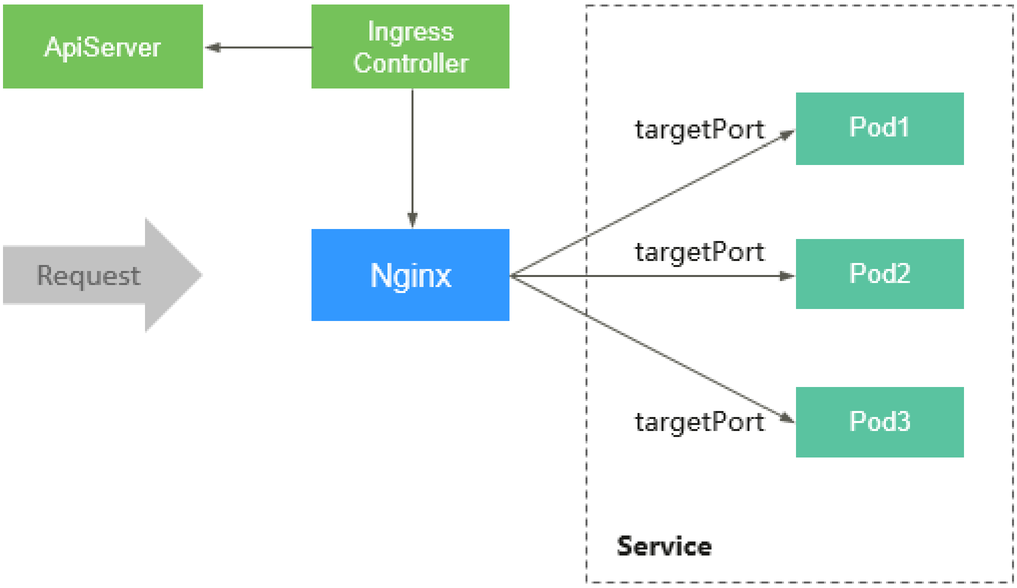
- When installing the NGINX Ingress Controller, you can specify Nginx parameters. These parameters take effect globally and are contained in the **nginx.conf** file. You can search for the parameters in **ConfigMaps**. If the parameters are not included in ConfigMaps, the configurations will not take 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 9-3** 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 9-3 How the NGINX Ingress Controller works



Constraints

- Dedicated load balancers with private IP addresses bound and used for network load balancing (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 **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 9-13 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 9-14 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

9.7 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.

- **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 Using Custom Metrics**: Custom metrics need to be aggregated to the Kubernetes API server for auto scaling.

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 **CCE Advanced HPA** on the right and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 9-15 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 9-16 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

10 Helm Chart

10.1 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.

Resource	Restriction Item	Description	Recommended Alternative Solution
	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	-
	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. The node affinity or the **nodeSelector** field does not take effect in CCE Autopilot clusters.	● 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.
	Pod affinity and anti-affinity	Ineffective	You do not need to set this parameter.
	allowPrivilegeEscalation (whether privilege escalation is allowed)	Not supported	Keep the default settings.

Resource	Restriction Item	Description	Recommended Alternative Solution
	RuntimeClassName	This parameter does not need to be configured. When RuntimeClassName is specified by an application (except pods), the value is automatically changed to **runc** supported by the system.	You do not need to set this parameter.
	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.
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.

Resource	Restriction Item	Description	Recommended Alternative Solution
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.

10.2 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 10-1**, the parameters marked with * are mandatory.

Table 10-1 Parameters in the directory structure of a chart

Parameter	Description
* templates	Stores all templates.
* 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, 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**. 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 10-2**.

Table 10-2 Installation parameters

Parameter	Description
Instance	Unique name of the chart release.
Namespace	Namespace to which the workload will be deployed.
Select Version	Version of a chart.
Configuration File	You can import and replace the **values.yaml** file or directly edit the chart parameters online. **NOTE** An imported **values.yaml** file must comply with YAML specifications, that is, KEY:VALUE format. The fields in the file are not restricted. The key value of the imported values.yaml must be the same as that of the selected chart package. Otherwise, the values.yaml does not take effect. That is, the key cannot be changed. 1. Click **Select File**. 2. Select the corresponding **values.yaml** file and click **Open**.

Step 4 Click **Install**.

On the **Releases** tab, 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**. 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**. 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**. 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