Cloud Container Engine

User Guide

Issue 01

Date 2021-02-28

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Contents

1 What Is Cloud Container Engine?

Cloud Container Engine (CCE) provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

CCE is deeply integrated with cloud services, including high-performance computing (ECS), network (VPC/EIP/ELB), and storage (EVS/OBS/SFS) services. It supports heterogeneous computing architectures such as GPU, Arm, and FPGA. By using multi-AZ and multi-region disaster recovery, CCE ensures high availability of Kubernetes clusters.

You can use CCE by using the **console**, **kubectl**, or APIs. Before using the CCE service, learn about the concepts related to Kubernetes. For details, see **https://kubernetes.io/docs/concepts/**.

- Junior users: You are advised to perform operations such as creating a cluster or workload by using the **console**.
- Advanced users: If you have experience in using kubectl, you are advised to use the **kubectl** and APIs to perform operations. For details, see **Kubernetes API** and **kubectl CLI**.

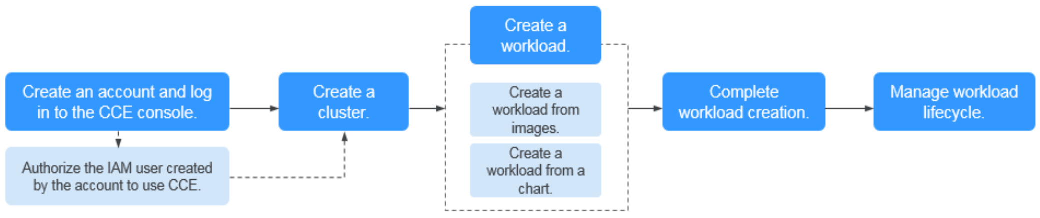
2 Cloud Container Engine (CCE) Introduction

This section provides instructions for getting started with CCE.

Procedure

Complete the following tasks to get started with CCE.

Figure 2-1 Procedure for getting started with CCE


Create an account and log in to the CCE console.
Authorize the IAM user created by the account to use CCE.
Create a cluster.
Create a workload.
Create a workload from images.
Create a workload from a chart.
Complete workload creation.
Manage workload lifecycle.

Step 1 **Authorize an IAM user to use CCE.**

The cloud accounts have the permissions to use CCE. However, IAM users created by the accounts do not have the permissions.

Step 2 **Create a cluster.**

For details, see **Creating a Hybrid Cluster**.

Step 3 **Create a workload from images or a chart.**

Select an existing image/chart or create a new one.

- For details on how to create a workload from an image, see **Workload Management**.
- For details on how to create a workload from a chart, see **Chart Management**.

Step 4 **View workload status and logs. Upgrade, scale, and monitor the workload.**

For details, see **Managing a Workload**.

----End

FAQs

1. **Is CCE suitable for users who are not familiar with Kubernetes?**

   Yes. The CCE console is easy-to-use, and the *Getting Started* guide helps you quickly understand and use CCE.

2. **Is CCE suitable for users who have little experience in building images?**

   Yes. You can select images from **Third-party Images** and **Shared Images** pages on the CCE console. The **My Images** page displays only the images created by you. For details, see **Workload Management**.

3. **How do I create a workload using CCE?**

   Create a cluster and then create a workload in the cluster.

4. **How do I create a workload accessible to public networks?**

   CCE provides different types of Services for workload access in diverse scenarios. Currently, CCE provides two access types to expose a workload to public networks: NodePort and LoadBalancer. For details, see **Network Management**.

5. **How can I allow multiple workloads in the same cluster to access each other?**

   Select the access type ClusterIP, which allows workloads in the same cluster to use their cluster-internal domain names to access each other.

   Cluster-internal domain names are in the format of <Self-defined service name>.<Workload's namespace>.svc.cluster.local:<Port number>. For example, nginx.default.svc.cluster.local:80.

   Example:

   Assume that workload A needs to access workload B in the same cluster. Then, you can create a ClusterIP Service for workload B. After the ClusterIP Service is created, workload B is reachable at <Self-defined service name>.<Workload B's namespace>.svc.cluster.local:<Port number>.

3 Product Bulletin

3.1 Risky Operations on Cluster Nodes

Precautions for Using a Cluster

- When performing operations such as creating, deleting, and scaling clusters, do not change user permissions on the Identity and Access Management (IAM) console. Otherwise, these operations may fail.
- Canal, the CNI plug-ins used by CCE nodes, uses a CIDR block as the CIDR block of the container network. This CIDR block can be configured during cluster creation and defaults to 172.16.0.0/16. The Docker service creates a docker0 bridge by default. The default docker0 address is 172.17.0.1. When creating a cluster, ensure that the CIDR block of the VPC in the cluster is different from those of the container network and the docker0 bridge. If VPC peering connections are used, ensure that the CIDR block of the peer VPC is different from those of the container network and the docker0 bridge.
- For clusters of Kubernetes v1.15, the DNS server of nodes in the cluster uses the DNS server in the VPC subnet. The CoreDNS address of Kubernetes is not added. Ensure that the DNS address in the subnet exists and is configurable.
- For clusters of Kubernetes v1.17, the network of a node is a single network plane. In the multi-network plane scenario, if you bind a new NIC to the ECS, you need to configure the NIC information on the node and restart the NIC after the binding.
- Do not modify the security groups, Elastic Volume Service (EVS) disks, and other resources created by CCE. Otherwise, clusters may not function properly. The resources created by CCE are labeled "cce", for example, "cce-evs-jwh9pcl7-****".
- When a node is added, the DNS server in the subnet must be able to resolve the domain name of the corresponding service. Otherwise, the node cannot be installed properly.

Precautions for Using a Node

Some of the node resources will be used to run necessary Kubernetes system components and resources to make the node as part of your cluster. Therefore, the amount of your node resources differs from that of node allocatable resources in Kubernetes. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components.

To ensure node stability, a certain amount of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

NOTE

You are advised not to install private software or modify the operating system (OS) configuration on a cluster node. This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.

Risky Operations on Nodes

After logging in to a node created by CCE, do not perform the following operations. Otherwise, the node will become unready.

Table 3-1 Operations that will cause nodes to become unready

No.	Operation	Impact	Solution
1	Reinstalling the operating system using the original image or another image	The node will become unavailable.	Delete the node and buy a new node.
2	Modifying OS configuration	The node will become unavailable.	Restore the original configuration or buy the node again.
3	Deleting the **opt** directory, **/var/paas** directory, or a data disk	The node will become unavailable.	Delete the node and buy a new node.
4	Formatting and partitioning a node disk	The node will become unavailable.	Delete the node and buy a new node.
5	Modifying a security group	The node will become unready or the cluster will exhibit unexpected behavior.	Correct the security group settings based on security group settings of normal clusters.

3.2 CCE Cluster Version Release Notes

To ensure that stable and reliable Kubernetes versions are available during your use of CCE, CCE provides the Kubernetes version support mechanism. A new supported version will be released every half a year with a support period of one year. You must upgrade your Kubernetes clusters before the support period ends.

For details about CCE cluster upgrade, see **Upgrade Overview**.

V1.17

Table 3-2 Feature description of clusters of v1.17

Kubernetes Version	Description
v1.17.9-r0	Main features: ● EulerOS 2.5 and CentOS 7.7 are supported. ● Features of Kubernetes v1.17.9 are incorporated.

V1.15

Table 3-3 Feature description of clusters of v1.15

Kubernetes version	Description
v1.15.11-r1	Main features: ● Support for EulerOS 2.5 ● Features of Kubernetes v1.15.11 are incorporated.
v1.15.6-r1	Main features: ● EulerOS 2.5 is supported. ● Kubernetes parameters can be dynamically configured. For details, see **Cluster Configuration Management** and **Managing a Node Pool**. ● CCE storage supports CSI-based cloud-native container storage systems. For details, see the **everest add-on**.

V1.13

Table 3-4 Feature description of clusters of v1.13

Kubernetes (CCE Enhanced Version)	Description
v1.13.10-r0	Main features: ● EulerOS 2.5 and CentOS 7.6 are supported. ● Features of Kubernetes v1.13.10 are incorporated.

V1.11 and Earlier Versions

Table 3-5 Feature description of clusters of v1.11 or earlier

Kubernetes (CCE Enhanced Version)	Description
v1.11.7-r2	Main features: ● Support for GPU V100 is provided. ● Features of Kubernetes v1.11.7 are incorporated.
v1.11.3-r1	Main features: ● Perl regular expressions can be used for matching ingress URLs. ● Features of Kubernetes v1.11.3 are incorporated. ● Master nodes of a cluster can be deployed across multiple AZs.
v1.9.7-r1	Main features: ● The mechanism for reporting PVC and PV events is enhanced. ● CCE works with a third-party authentication system. ● Physical machines that use EulerOS 2.3 can be managed. ● BMS clusters can work with Elastic Volume Service (EVS).

3.3 Cluster Node OS Patch Notes

Hybrid cluster

In a hybrid cluster, CCE nodes support EulerOS 2.2, EulerOS 2.5, CentOS 7.4, CentOS 7.6, and CentOS 7.7. The corresponding kernel versions are as follows:

Table 3-6 Cluster Node OS Patch Notes

OS	Patch
CentOS Linux release 7.4	3.10.0-693.11.1.el7.x86_64
EulerOS release 2.0 (SP2)	3.10.0-327.62.59.83.h128.x86_64
CentOS Linux release 7.6.1810 (Core)	3.10.0-957.5.1.el7.x86_64
EulerOS release 2.0 (SP5)	3.10.0-862.14.1.2.h249.eulerosv2r7.x86_64
CentOS Linux release 7.7	3.10.0-1062.12.1.el7.x86_64

The OS patch release and verification results will be updated periodically. You can update the operating system if needed.

3.4 Security Vulnerability Responses

3.4.1 Notice on Fixing Linux Kernel SACK Vulnerabilities

The Linux Kernel SACK vulnerabilities have been fixed. This section describes the solution to these vulnerabilities.

Vulnerability Details

On June 18, 2019, Red Hat released a security notice, stating that the TCP SACK module of the Linux kernel is exposed to three security vulnerabilities (CVE-2019-11477, CVE-2019-11478, and CVE-2019-11479). These vulnerabilities are related to the maximum segment size (MSS) and TCP Selective Acknowledgment (SACK) packets. Remote attackers can exploit these vulnerabilities to trigger a denial of service (DoS), resulting in server unavailability or breakdown.

Reference link:

https://www.suse.com/support/kb/doc/?id=7023928

https://access.redhat.com/security/vulnerabilities/tcpsack

https://www.debian.org/lts/security/2019/dla-1823

https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/SACKPanic?

https://lists.centos.org/pipermail/centos-announce/2019-June/023332.html

https://github.com/Netflix/security-bulletins/blob/master/advisories/third-party/2019-001.md

Table 3-7 Vulnerability information

Vulnerability Type	CVE-ID	Published	Fixed
Input validation flaw	**CVE-2019-11477**	2019-06-17	2019-07-06
Resource management flaw	**CVE-2019-11478**	2019-06-17	2019-07-06
Resource management flaw	**CVE-2019-11479**	2019-06-17	2019-07-06

Affected Versions

Linux> = 2.6.29 (CVE-2019-11477)

Solutions

NOTICE

- EulerOS 2.2 supports an upgrade to the kernel version 3.10.0-327.62.59.83.h162.x86_64.
- CentOS 7.4 supports an upgrade to the latest kernel 3.10.0-957.21.3.e17.x86_64.
- The node must be accessible to external networks. After the kernel is upgraded, restart the system.
- The following errors may be encountered during the upgrade. However, they do not affect system functions and can be ignored.

```
depmod: ERROR: fstatat(9, vport-gre.ko): No such file or directory
depmod: ERROR: fstatat(9, vport-vxlan.ko): No such file or directory
depmod: ERROR: fstatat(9, vport-geneve.ko): No such file or directory
depmod: ERROR: fstatat(9, openvswitch.ko): No such file or directory
depmod: ERROR: fstatat(5, vport-gre.ko): No such file or directory
depmod: ERROR: fstatat(5, vport-vxlan.ko): No such file or directory
depmod: ERROR: fstatat(5, vport-geneve.ko): No such file or directory
depmod: ERROR: fstatat(5, openvswitch.ko): No such file or directory
```

Step 1 Log in to the node as user **root** and run the following command to update the kernel:

yum update kernel -y

Step 2 When the **yum update** command is used to upgrade the operating system, container network components could become unavailable. Run the following command to restore the components:

- EulerOS 2.2

```
#!/bin/bash
function upgrade_kmod()
{
    openvswicth_mod_path=$(rpm -qal openvswitch-kmod)
    rpm_version=$(rpm -qal openvswitch-kmod|grep -w openvswitch|head -1|awk -F "/" '{print $4}')
    sys_version=`cat /boot/grub2/grub.cfg | grep EulerOS|awk 'NR==1{print $3}' | sed 's/[()]//g'`

    if [[ "${rpm_version}" != "${sys_version}" ]];then
        mkdir -p /lib/modules/"${sys_version}"/extra/openvswitch
        for path in ${openvswicth_mod_path[@]};do
            name=$(echo "$path" | awk -F "/" '{print $NF}')
            rm -f /lib/modules/"${sys_version}"/updates/"${name}"
            rm -f /lib/modules/"${sys_version}"/extra/openvswitch/"${name}"
            ln -s "${path}" /lib/modules/"${sys_version}"/extra/openvswitch/"${name}"
        done
    fi
    depmod ${sys_version}
}
upgrade_kmod
```

- CentOS 7.4

```
function upgrade_ovs()
{
    wget http://endpoint/cce-east/cce-openvswitch/openvswitch-1.0.RC10.SPC100.B050.tar.gz
    mv /var/paas/kubernetes/canal/openvswitch /var/paas/kubernetes/canal/openvswitch.bak
    tar zxvf openvswitch-1.0.RC10.SPC100.B050.tar.gz -C /var/paas/kubernetes/canal/
    bash /var/paas/kubernetes/canal/openvswitch/can_ovs.sh install
    su paas; monit restart ovsdb-server ovs-vswitchd
}
upgrade_ovs
```

Step 3 Restart the VM.

reboot

----End

4 Cluster Management

4.1 Cluster Overview

The clustering technology improves performance, reliability, and flexibility at an affordable cost. Task scheduling is essential to clusters.

A cluster is a collection of cloud resources required for container running. A cluster is associated with cloud resources such as cloud server nodes, load balancers, and virtual private clouds (VPCs). A cluster can be seen as one or more elastic cloud servers (ECSs, also called nodes) in a same subnet. It provides computing resource pool for the container running through computer groups formed by relevant technologies.

Relationship Between a Cluster, VPC, and Subnet

- A **VPC** is similar to a private local area network (LAN) managed by a home gateway whose IP address is 192.168.0.0/16. A VPC is a private network built on the cloud and provides basic network environment for running elastic cloud servers (ECSs), elastic load balancers (ELBs), and middleware. Networks of different scales can be set according to the actual service requirements. Generally, the networks can be 10.0.0.0/8–24, 172.16.0.0/12–24, or 192.168.0.0/16–24. The largest network is the class-A address network of 10.0.0.0/8.
- A subnet is a subset of a VPC. A VPC can be divided into one or more subnets. Security groups are configured to determine whether these subnets can communicate with each other. This ensures that subnets can be isolated from each other, so that you can deploy different services on different subnets.
- A cluster is one or more elastic cloud servers or bare metal servers (also called nodes) in multiple subnets. It provides computing resource pool for the container running through computer groups formed by relevant technologies.

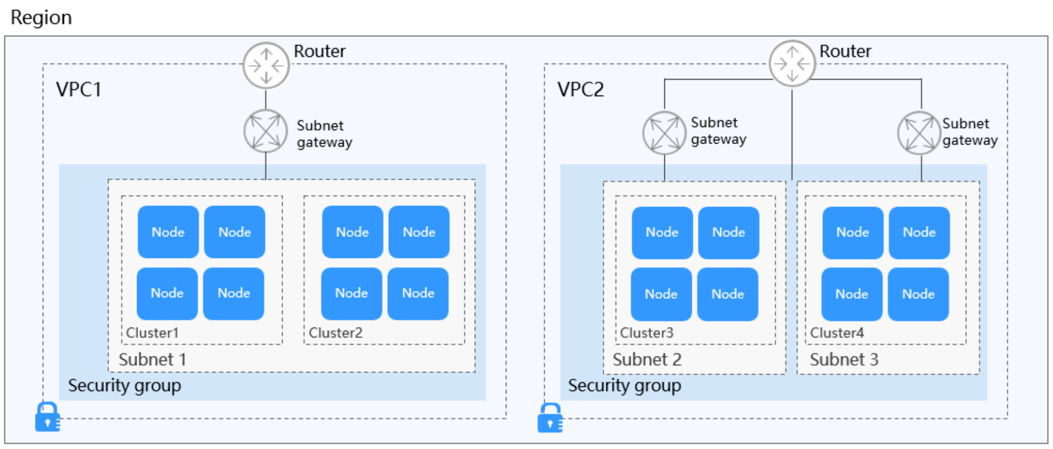
  NOTE

  Nodes in a cluster can belong to different subnets.

As shown in the following figure, multiple VPCs are configured in a region. A VPC consists of one or more subnets. The subnets communicate with each other through the subnet gateway. A cluster is created in a subnet. Therefore, there are three scenarios:

- Different clusters are created in different VPCs.
- Different clusters are created in the same subnet.
- Different clusters are created in different subnets.

Figure 4-1 Relationship between a cluster, VPC, and subnet



Precautions for Using a Cluster

- When performing operations such as creating, deleting, and scaling clusters, do not change user permission in the Identity and Access Management (IAM) console. Otherwise, these operations may fail.
- The containerized network canal of CCE nodes uses a CIDR block as the CIDR block of the container network. This CIDR block can be configured during cluster creation and defaults to 172.16.0.0/16. The Docker service creates a docker0 bridge by default. The default docker0 address is 172.17.0.1. When creating a cluster, ensure that the CIDR block of the VPC in the cluster is different from the CIDR blocks of the container network docker0 bridge. If VPC peering connections are used, ensure that the CIDR block of the peer VPC is different from those of the container network and the docker0 bridge.
- Nodes in clusters of Kubernetes v1.15 uses the DNS in the VPC subnet. The CoreDNS of Kubernetes is not added. Ensure that the DNS address has been and can be configured in the subnet.
- In clusters of Kubernetes v1.17, the node network is a single network plane. In a multi-network plane scenario, if you bind a new NIC to the ECS, you need to configure the NIC information on the node and restart the NIC after the binding.
- Do not modify the security groups, Elastic Volume Service (EVS) disks, and other resources created by CCE. Otherwise, clusters may not function properly. The resources created by CCE are labeled **cce**, for example, **cce-evs-jwh9pcl7-******.
- When adding a node, ensure that the DNS server in the subnet can resolve the domain name of the service. Otherwise, the node cannot be installed.

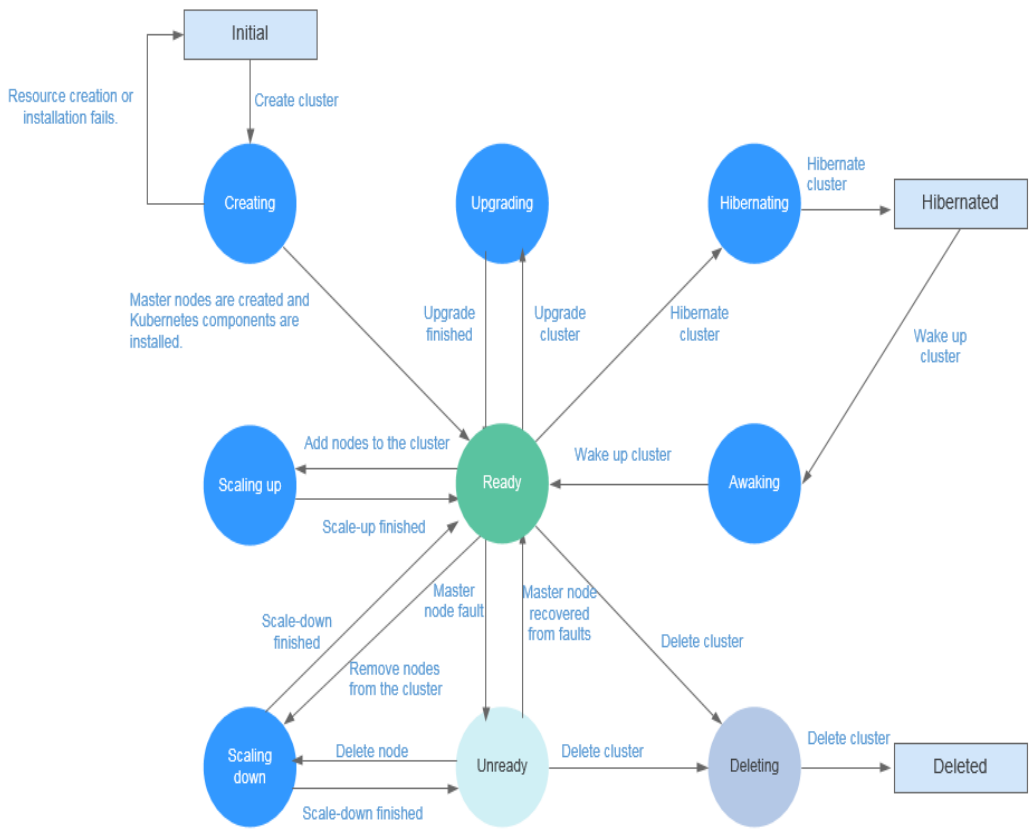
4.2 Cluster Lifecycle

This section describes the status of each cluster lifecycle, helping you understand the running status of the cluster in different phases.

Table 4-1 Cluster status description

Status	Description
Creating	A cluster is being created and is applying for cloud resources.
Normal	The cluster is running properly.
Scaling-out	A node is being added to the cluster.
Scaling-in	A node is being deleted from the cluster.
Hibernating	The cluster is hibernating.
Awaking	The cluster is being woken up.
Upgrading	The cluster is being upgraded.
Unavailable	The current cluster is unavailable.
Deleting	The cluster is being deleted.

Figure 4-2 Cluster status transition



4.3 Creating a Hybrid Cluster

On the CCE console, you can easily create Kubernetes clusters. Kubernetes can manage container clusters at scale. A cluster manages a group of node resources.

Support for hybrid clusters is available now. VM clusters are renamed as hybrid clusters, which can manage VM nodes, bare-metal nodes, and both.

Preparation

- Before creating your first cluster, you must create a VPC. If you already have an available VPC, skip this preparatory step.

  A VPC provides an isolated, configurable, and manageable virtual network for CCE clusters.
- A key pair has been created. The key pair will be used for identity authentication upon remote node login.
- Plan the container CIDR block and service CIDR block before creating a cluster. The CIDR block is a one-time configuration and cannot be changed after the cluster is created. If you want to use another container CIDR block, you have to create a new cluster and assign the new container CIDR block to the cluster.

Precautions for Creating a Cluster

Some basic resources are created during cluster creation, as shown in the following table.

Table 4-2 Precautions for creating a cluster

Resource	Description
Master nodes and related resources	Associated with CCE resource tenants, and invisible to you.
ECSs (optional)	An ECS corresponds to a cluster node that provides computing resources. An ECS is named in the format of *Cluster name-Random number*. The name format is user-defined. ECSs created in batches are named in the format of *Cluster name-Random number 1-Random number 2*.
Security groups	Two security groups are created for a cluster: one for managing master nodes, and the other for managing worker nodes. **NOTICE** Do not delete the security group settings and security group rules configured during cluster creation. Otherwise, the cluster will exhibit unexpected behavior. 1. Security group for master nodes Name format: *Cluster name*-**cce-control**-*Random number* Functions: ● Allows outbound traffic. ● Allows other nodes to access Kubernetes services of masters. 2. Security group for nodes Name format: *Cluster name*-**cce-node**-*Random number* Functions: ● Allows outbound traffic. ● Allows remote login to Linux or Windows OSs using ports 22 and 3389. ● Allows communication between Kubernetes components using ports 4789 and 10250. ● Allows Kubernetes to provide services for external systems using ports 30000 and 32767. ● Allows communication between nodes in the same security group.
Disks (optional)	Two disks are configured for each node. One is the system disk, and the other is the data disk used to run Docker.

Resource	Description
Elastic IP address (optional)	Any node that requires access to external networks must have an elastic IP address (EIP).

Procedure

Step 1 Log in to the CCE console. Choose **Dashboard** > **Create Cluster** to go to the **Create Hybrid Cluster** page. Alternatively, choose **Resource Management** > **Clusters** in the navigation pane and click **Create** under **Hybrid Cluster**.

Step 2 Set the parameters listed in **Table 4-3**. The parameters marked with an asterisk (*) are mandatory.

Table 4-3 Parameters for creating a cluster

Parameter	Description
*Region	To minimize network latency and resource access time, select the nearest region. Cloud resources are region-specific and cannot be used across regions through internal network connections.
*Cluster Name	Name of the new cluster, which cannot be changed after the cluster is created. A cluster name contains 4 to 128 characters starting with a letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
*Version	Kubernetes community baseline version. The latest version is recommended. For details about version upgrade, see **Upgrade Overview**.
*Management Scale	Maximum number of worker nodes that can be managed by a cluster. If you select **50 nodes**, the cluster can manage a maximum of 50 worker nodes. The cluster management scale cannot be modified after a cluster is created. Exercise caution when creating a cluster. Each cluster contains at least one **master node** and at least one **worker node**. A worker node is a cloud server. ● Master node: controls worker nodes in the cluster. The master node is automatically created along with the cluster, and manages and schedules the entire cluster. ● Worker node: runs the workload you deploy. You can create a worker node. The master node assigns a worker node to each deployable component of your workload. When a worker node is down, the master node migrates the workload to another worker node.

Parameter	Description
*High Availability	● **Yes**: An HA cluster will be created. Typically, an HA cluster has three master nodes. The cluster is still available even when a master node is down. ● **No**: The cluster has only one master node. If the master node is down, the cluster stops serving new workloads, but existing workloads are not affected. The HA setting is a one-time configuration and cannot be changed after the cluster is created. If you want to use another HA setting for the cluster, you have to create a new cluster that has exactly the same parameter settings (except for the **High Availability** parameter) as the current cluster. Set this parameter based on the site requirements. ● In production environments, it is advised to set **High Availability** to **Yes** to improve cluster's disaster recovery capabilities. ● In R&D and testing environments that do not require high reliability, it is not always necessary to set **High Availability** to **Yes**.
*VPC	VPC where the new cluster is located. A VPC provides a secure and logically isolated network environment. If no VPC is available, click **Create VPC** to create a VPC. After the VPC is created, click the refresh icon.
* Subnet	The subnet where cluster nodes will run. A subnet improves network security by providing exclusive network resources that are isolated from other networks. For details about the relationship between VPCs, subnets, and clusters, see **Cluster Overview**. **The selected subnet cannot be modified after the cluster is created.** Exercise caution when selecting a subnet.

Parameter	Description
*Network Model	● **Tunnel network**: The container network is an overlay tunnel network on top of a VPC network and uses the VXLAN technology. This network model is applicable when there is no high requirements on performance. VXLAN encapsulates Ethernet packets as UDP packets for tunnel transmission. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications. ● **VPC network**: The container network uses VPC routing to integrate with the underlying network. This network model is applicable to performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. VPC networks are free from tunnel encapsulation overhead and outperform container tunnel network models. In addition, as VPC routing includes routes to node IP addresses and container network segment, container instances in the cluster can be directly accessed from outside the cluster. **NOTE** When **VPC network** is selected, you can set the maximum number of pods that can be created on a node in the cluster. The value cannot be changed after being set. After the cluster is created, the value of **Max Pods** in the **Advanced Settings** area of the **Create Node** page must be lesser than or equal to the value specified here.
*Container Network Segment	An IP address range that can be allocated to container instances. ● If **Automatically select** is deselected, you must select a CIDR block. If the CIDR block you select conflicts with a subnet CIDR block, the system prompts you to select another CIDR block. The recommended CIDR blocks are 10.0.0.0/8-18, 172.16.0.0/16-18, and 192.168.0.0/16-18. **If different clusters share a container CIDR block, an IP address conflict will occur and access to applications will fail.** ● If **Automatically select** is selected, the system automatically assigns a CIDR block that does not conflict with any subnet CIDR block. The mask of the container CIDR block must be appropriate. It determines the number of available nodes in a cluster. A too small mask value will cause the cluster to soon fall short of nodes. After the mask is set, the estimated maximum number of containers supported by the current CIDR block will be displayed.

Parameter	Description
*Service Network Segment	CIDR block of Kubernetes Services. ● **Default**: The default CIDR block 10.247.0.0/16 will be used. ● **Custom**: Manually set a CIDR block and mask based on service requirements. The mask determines the maximum number of Service IP addresses available in the cluster.
Authorization Mode	By default, **RBAC** is selected. After RBAC is enabled, IAM users access resources in the cluster according to fine-grained permissions policies.
Authentication Mode	The authentication mechanism controls user permission on resources in a cluster. For example, you can grant user A the read and write permissions on applications in a specified namespace, while granting user B the read permission on resources in a cluster. For details about role-based permission control, see **3.7- Cluster Management Permission Control**. ● By default, X.509 authentication instead of **Enhanced authentication** is enabled. X.509 is a standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols. ● If permission control on a cluster is required, select **Enhanced authentication** and then **Authenticating Proxy**. Click **Upload** next to **CA Root Certificate** to upload a valid certificate. Select the check box to confirm that the uploaded certificate is valid. If the certificate is invalid, the cluster cannot be created. The uploaded certificate file must be smaller than 1 MB and in .crt or .cer format.
Cluster Description	Optional. Enter the description of the new container cluster.

Parameter	Description
Advanced Settings	Click ^ to show advanced settings. **Service Forwarding Mode** ● **iptables**: Traditional kube-proxy uses iptables rules to implement service load balancing. In this mode, too many iptables rules will be generated when many Services are deployed. In addition, non-incremental updates will cause a latency and even tangible performance issues in the case of service traffic spikes. ● **ipvs**: Optimized kube-proxy mode with higher throughput and faster speed. This mode supports incremental updates and can keep connections uninterrupted during service updates. It is suitable for large-sized clusters. **NOTE** ● ipvs provides better scalability and performance for large clusters. ● ipvs supports more complex load balancing algorithms than iptables. ● ipvs supports server health checking and connection retries. **Open EIP** An independent public IP address that is reachable from public networks. Select an EIP that has not been bound to any node. A cluster's EIP is preset in the cluster's certificate. Do no delete the EIP after the cluster has been created. Otherwise, two-way authentication will fail. ● **Do not configure**: The cluster's master node will not have an EIP. ● **Configure now**: If no EIP is available for selection, create a new EIP. **CPU Policy** This parameter is displayed only for clusters of v1.13.10-r0 and later. ● **On**: Exclusive CPU cores can be allocated to workload pods. Select **On** if your workload is sensitive to latency in CPU cache and scheduling. ● **Off**: Exclusive CPU cores will not be allocated to workload pods. Select **Off** if you want a large pool of shareable CPU cores. For details about CPU policies, see **Feature Highlight: CPU Manager**. **Master AZs** Master nodes can be deployed in multiple AZs. The multi-AZ deployment of master nodes achieves disaster tolerance on the cluster management plane at a certain sacrifice of cluster performance. **NOTE** This parameter is displayed only when there are three or more AZs.

Step 3 Click **Next: Create Node**. On the **Create Node** page, set the following parameters.

- **Node**
  - **Create now**: A specified number of nodes will be created along with the cluster. The **Nodes** parameter indicates the quantity of nodes.
  - **Create later**: Create an empty cluster without creating nodes. Click **Next**.
- **Current Region**: Geographic location of the nodes to be created.
- **AZ**: Set this parameter based on the site requirements. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

  To enhance workload availability, create nodes in different AZs.
- **Node Type**

  VM node: A VM node will be created in the cluster.
- **Node Name**: Set the name of the new node. A node name contains 1 to 56 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **Specifications**: Select node specifications that fit your business needs.
  - **General-purpose**: provides a balance of computing, memory, and network resources. It is a good choice for many applications. General-purpose nodes can be used for web servers, workload development, workload testing, and small-scale databases.
  - **Memory-optimized**: provides higher memory capacity than general-purpose nodes and is suitable for relational databases, NoSQL, and other workloads that are both memory-intensive and data-intensive.
  - **GPU-accelerated**: provides powerful floating-point computing and is suitable for real-time, highly concurrent massive computing. Graphical processing units (GPUs) of P series are suitable for deep learning, scientific computing, and CAE. GPUs of G series are suitable for 3D animation rendering and CAD.
  - **General computing-plus**: provides stable performance and exclusive resources to enterprise-class workloads with high and stable computing performance.
  - **Ultra-high I/O**: provides ultra-low SSD access latency and ultra-high IOPS performance. This type of specifications is suitable for high-performance relational databases, NoSQL databases (such as Cassandra and MongoDB), and Elasticsearch.

  To ensure node stability, CCE automatically reserves some resources to run necessary system components. For details, see **Formula for Calculating the Reserved Resources of a Node**.
- **OS**: Select the operating system (OS) of the nodes to be created. For details, see **Cluster Node OS Patch Notes**.
- **System Disk**: Set the system disk space of the worker node. The value ranges from 40 GB to 1024 GB. The default value is 40 GB.

  By default, the following types of EVS disks are supported:
  - **Common I/O**: EVS disks of this type deliver a maximum of 2,200 IOPS. This disk type is suitable for application scenarios that require large capacity, a medium read/write rate, and fewer transactions, such as enterprise office applications and small-scale testing.

  - **High I/O**: EVS disks of this type deliver a maximum of 5,000 IOPS and a minimum of 1 ms read/write latency. This disk type is designed to meet the needs of mainstream high-performance, high-reliability application scenarios, such as enterprise applications, small- and medium-scale development and testing, and web server logs.
  - **Ultra-high I/O**: EVS disks of this type deliver a maximum of 33,000 IOPS and a minimum of 1 ms read/write latency. This disk type is excellent for ultra-high I/O, ultra-high bandwidth, and read/write-intensive application scenarios, such as distributed file systems in HPC scenarios or NoSQL/RDS in I/O-intensive scenarios.
- **Data Disk**: Set the data disk space of the worker node. The value ranges from 100 GB to 32,678 GB. The types of EVS disks supported for data disks are the same as those for the system disk.

  ⚠ CAUTION

  If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk.

  - **LVM**: If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space allocation for different resources.
    - This option is selected by default, indicating that LVM management is enabled.
    - You can deselect the check box to disable LVM management.

    ⚠ CAUTION

    - When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in **How Do I Add a Second Data Disk to a Node in a CCE Cluster?** to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM.
    - When creating a node in a cluster of v1.13.10 or earlier, if a data disk is not managed by LVM, format the data disk. Otherwise, either this data disk or the first data disk will be managed by LVM, which is not as expected.
  - **Add Data Disk**: Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to attach more data disks.
  - **Data disk space allocation**: Click **Change Configuration** to specify the resource ratio for **Kubernetes Space** and **User Space**.
    - **Kubernetes Space**: You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include Docker images and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs.

- **User Space**: You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted. **Path inside a node** cannot be set to the root directory **/**. Otherwise, the mounting fails.

NOTE

Disk space of the data disks managed by LVM will be allocated according to the ratio you set.

If the cluster version is v1.13.10-r0 or later and the node type is ultra-high I/O, the following options are displayed for data disks:

– **EVS**: The disk capacity ranges from 100 to 32,678 GB. You can set a value as required.

  If you select **Data disk space allocation**, you can allocate data disk space for storing Docker and Kubelet resources.

– **Local disk**: This value is displayed only for **ultra-high I/O** nodes in clusters of v1.13.10-r0 or later. Local disks may break down and do not ensure data reliability. It is recommended that you store service data in EVS disks, which are more reliable than local disks. Local disk parameters are as follows:

  - **Disk Mode**: When the node type is ultra-high I/O, solid state disks (SSDs) are supported.
  - **Kubernetes Space**: the percentage of data disk space allocated for storing Docker and kubelet resources. Docker resources include Docker images and image metadata; kubelet resources include pod configuration files, secrets, and emptyDirs.
  - **User Space**: You can specify the ratio of the local disk space that is not allocated for storing Kubernetes resources.

NOTICE

By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability. For more information, click **here**.

- **VPC**: This parameter is available only for clusters of v1.13.10-r0 and later.
- **Subnet**: A subnet improves network security by providing exclusive network resources that are isolated from other networks. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets.

  This parameter is displayed only for clusters of v1.13.10-r0 and later.
- **EIP**: An independent public IP address. If the nodes to be created require Internet access, select **Automatically assign** or **Use existing**.

NOTE

By default, VPC's SNAT feature is disabled for CCE. If SNAT is enabled, you do not need to use EIPs to access public networks.

- **Do not use**: A node without an EIP cannot be accessed from public networks. It can be used only as a cloud server for deploying services or clusters on a private network.
- **Automatically assign**: An EIP with specified configurations is automatically assigned to each node. If the number of EIPs is less than the number of nodes, the EIPs are randomly bound to the nodes.

  Set the specifications, required quantity, billing mode, and bandwidth as required. When creating an , ensure that the EIP quota is sufficient.
- **Use existing**: Existing EIPs are assigned to the nodes to be created.

- **Login Mode**: A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair** and create one.
- **Advanced ECS Settings**: (Optional) Click ⌄ to show advanced ECS settings.
  - **ECS Group**: Select an existing ECS group, or click **Create ECS Group** to create a new one. After the ECS group is created, click the refresh icon.

    An ECS group allows you to create ECSs on different hosts, thereby improving service reliability.
  - **Resource Tags**: By adding tags to resources, you can classify resources.

    You can create predefined tags in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency.

    CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag. A maximum of 5 tags can be added.
  - **Agency**: An agency is created by the account administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
  - **Pre-installation Script**: Enter a maximum of 1,000 characters.

    The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may not be installed successfully. The script is usually used to format data disks.
  - **Post-installation Script**: Enter a maximum of 1,000 characters.

    The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
  - **Subnet IP Address**: Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.
- **Advanced Kubernetes Settings**: (Optional) Click ⌄ to show advanced Kubernetes settings.
  - **Max Pods**: Maximum number of pods that can be created on a node, including the system's default pods. Value range: 16 to 250.

    This maximum limit prevents the node from being overloaded by managing too many instances.

- **insecure-registries**: Click **Add insecure-registry** and enter the address of the image repository.

  Add the address of the custom image repository with no valid SSL certificate to the docker startup option to avoid unsuccessful image pulling from the personal image repository. The address is in the format of IP address:Port number (or domain name). Post-installation script and insecure-registries cannot be used together.
- **Maximum Data Space per Container**: maximum data space that can be used by a container. The value ranges from 10 GB to 80 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.

- **Nodes**: number of nodes to be created. The value cannot exceed the maximum number of nodes that can be managed by the cluster. Set this parameter based on service requirements and the message displayed on the page.

Step 4 Click **Next: Install Add-on** to install add-ons.

System resource add-ons must be installed. Advanced functional add-ons are optional.

You can also install optional add-ons after the cluster is created. To do so, choose **Add-ons** in the navigation pane of the CCE console and select the add-on you will install. For details, see **13 Add-on Management**.

Step 5 Click **Next: Confirm**. Read the product constraints and select **I am aware of the above limitations**. Confirm the configured parameters and specifications.

Step 6 Click **Submit**.

It takes about 6 to 10 minutes to create a cluster. You can click **Back to Cluster List** to perform other operations on the cluster or click **Go to Cluster Events** to view the cluster details.

----End

Related Operations

- After creating a cluster, you can use the Kubernetes CLI tool kubectl to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.
- Add more nodes to the cluster. For details, see **Creating a Node**.
- Log in to a node. For details, see **Logging In to a Node**.
- Create a namespace. You can create multiple namespaces in a cluster and organize resources in the cluster into different namespaces. These namespaces serve as logical groups and can be managed separately. For more information about how to create a namespace for a cluster, see **Namespace**.
- Click the cluster name to view cluster details.

Table 4-4 Cluster details

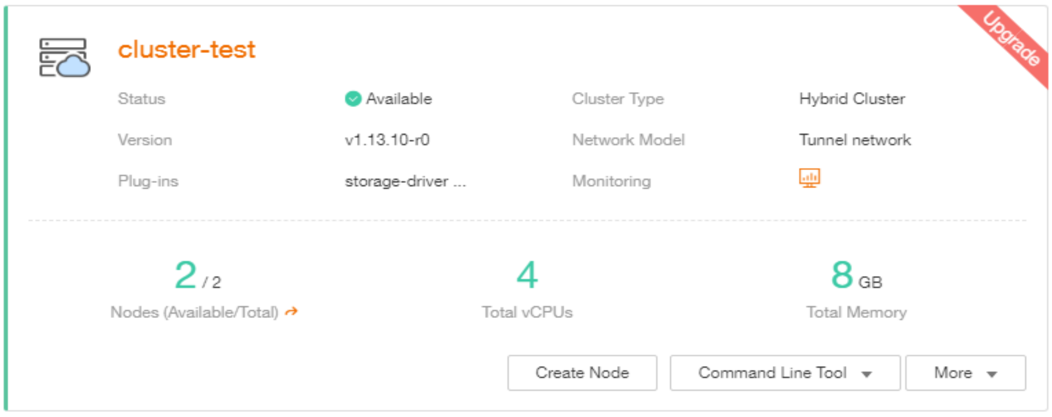
Tab	Description
Cluster Details	View the details and operating status of the cluster.
Monitoring	Check the CPU and memory usage of the cluster over the past 1 hour, 3 hours, or 12 hours.
Events	● View cluster events on the **Events** tab page. ● Set search criteria. For example, you can set the time segment or enter an event name to view corresponding events.

4.4 Upgrade Overview

You can upgrade your cluster as long as your cluster is ready for upgrade.

You can log in to the CCE console, choose **Resource Management** > **Clusters**, and check whether a upgrade flag is displayed in the upper right corner of the cluster to be upgraded. The flag indicates that the cluster can be upgraded.

Figure 4-3 Cluster with the upgrade flag



cluster-test

Status	Available	Cluster Type	Hybrid Cluster
Version	v1.13.10-r0	Network Model	Tunnel network
Plug-ins	storage-driver ...	Monitoring	

2 / 2 Nodes (Available/Total)

4 Total vCPUs

8 GB Total Memory

Table 4-5 describes how to upgrade a cluster of v1.17 or earlier.

Table 4-5 Instructions for upgrade between major versions

Source Version	Target Version	Description
v1.15.11-r1 v1.15.6-r1	v1.17.9-r0	Changelog from v1.15 to v1.17 **https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.17.md**

Source Version	Target Version	Description
v1.13.10-r0	v1.15.11-r1	Changelog from v1.13 to v1.15 **https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.15.md**
v1.11.7-r2	v1.13.10-r0	● Changelog from v1.9 to v1.13 Changelog from 1.11 to 1.13: **https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.13.md** Changelog from 1.10 to 1.11: **https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.11.md** Changelog from 1.9 to 1.10: **https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.10.md** ● kube-dns of the cluster is replaced by core-dns.
v1.11.3-r1		
v1.9.7-r1		

4.5 Upgrading a Cluster

Kubernetes versions are expressed as x.y.z, where x is the major version, y is the minor version (for example, v1.13), and z is the patch version. Minor releases occur approximately every 3 months to provide new features, design updates, and bug fixes, and each minor release branch is maintained for approximately one year. You can upgrade your cluster to the latest Kubernetes version or a bug-fixed version on the CCE console.

Precautions

- There is a certain possibility that cluster upgrade may fail. To avoid data loss, back up data before the upgrade.
- During the upgrade from Kubernetes 1.9 to 1.11, the kube-dns of the cluster will be uninstalled and replaced with CoreDNS, which may cause loss of the cascading DNS configuration in the kube-dns or temporary interruption of the DNS service. Back up the DNS address configured in the kube-dns so you can configure the domain name in the CoreDNS again when domain name resolution is abnormal.
- You can upgrade user nodes only by resetting the nodes. Resetting the nodes to upgrade is simple and with high success rate. You can also upgrade the operating system. During the upgrade, services will be interrupted, the OS of the node will be reinstalled, and the data on the system disk and data disk will be cleared.

- User nodes can be upgraded in batches to reduce the impact of service interruption.

Preparation

Before upgrading a cluster, make sure that the cluster is healthy.

Step 1 Connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Run the following command to verify that all modules are in the Healthy state:

kubectl get cs

Information similar to the following is displayed:

```
NAME                STATUS    MESSAGE             ERROR
 scheduler           Healthy   ok
 controller-manager   Healthy   ok
 etcd-0              Healthy   {"health": "true"}
 etcd-1              Healthy   {"health": "true"}
 etcd-2              Healthy   {"health": "true"}
```

Step 3 Run the following command to verify that all nodes are in the Ready state:

kubectl get nodes

NOTE

All nodes must be in the Ready state.

```
NAME             STATUS    ROLES     AGE      VERSION
eu-west-0a-xxx   Ready     master    38d      v1.9.7-r1
eu-west-0a-xxx   Ready     <none>    38d      v1.9.7-r1
eu-west-0a-xxx   Ready     <none>    38d      v1.9.7-r1
eu-west-0a-xxx   Ready     <none>    38d      v1.9.7-r1
eu-west-0a-xxx   Ready     master    38d      v1.9.7-r1
eu-west-0a-xxx   Ready     master    38d      v1.9.7-r1
```

----End

Cluster Pre-upgrade Checklist

Before upgrading a cluster, follow the pre-upgrade checklist to identify risks and problems in advance.

Table 4-6 Cluster pre-upgrade checklist

Object	Check Item
Cluster	Check whether IP addresses (including EIPs) of cluster nodes are included in other configurations/whitelists.
	Complete the pre-upgrade check.
Workload	Record the quantity and status of workloads for comparison after the upgrade.
	Evaluate the possible changes that the upgrade may bring to whitelists, routing, and security group policies of databases.
Storage	Record storage status. Storage must not be lost after the upgrade.

Object	Check Item
Network	Check and back up load balancing services and Ingresses.
	If Direct Connect is used and node/pod IP addresses may change after the upgrade, configure new routes on Direct Connect in advance.
Add-on	When Kubernetes v1.9 is upgraded to v1.11, the kube-dns of the cluster will be uninstalled and replaced with CoreDNS. Back up the DNS address configured in kube-dns so that you can configure the DNS address in CoreDNS when domain name resolution is abnormal.
O&M	Ensure that nodes and containers do not store configurations such as data-plane passwords, certificates, and environment variables. After the upgrade, nodes and containers may restart (typically, a container restarts when the pod is migrated from a faulty node to an available node) and the configurations will become lost, which results in service exceptions.
	Check and back up kernel parameters or system configurations.

Procedure

This section describes how to upgrade a VM cluster from v1.9.7-r1 to v1.11.7-r2. The procedure for other upgrade paths is similar.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. On the cluster list, view your cluster version. In this example, the cluster version is v1.9.7-r1.

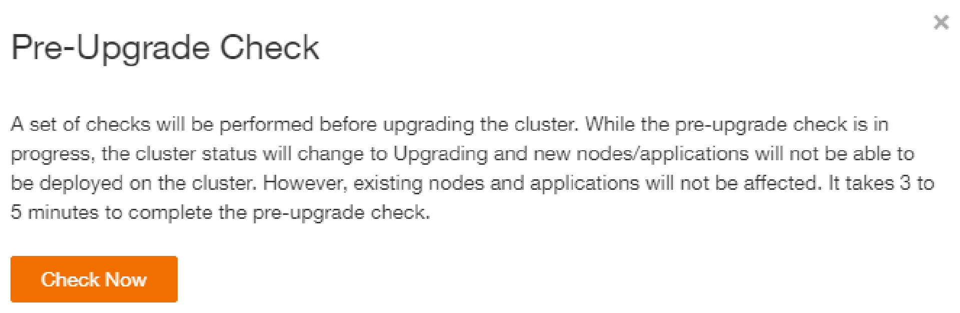
NOTE

If your cluster version is up-to-date, the **Upgrade** button is unavailable.

Step 2 Click **More** for the cluster you want to upgrade, and select **Upgrade** from the drop-down menu.

Step 3 In the displayed **Pre-upgrade Check** dialog box, click **Check Now**.

Figure 4-4 Pre-upgrade check

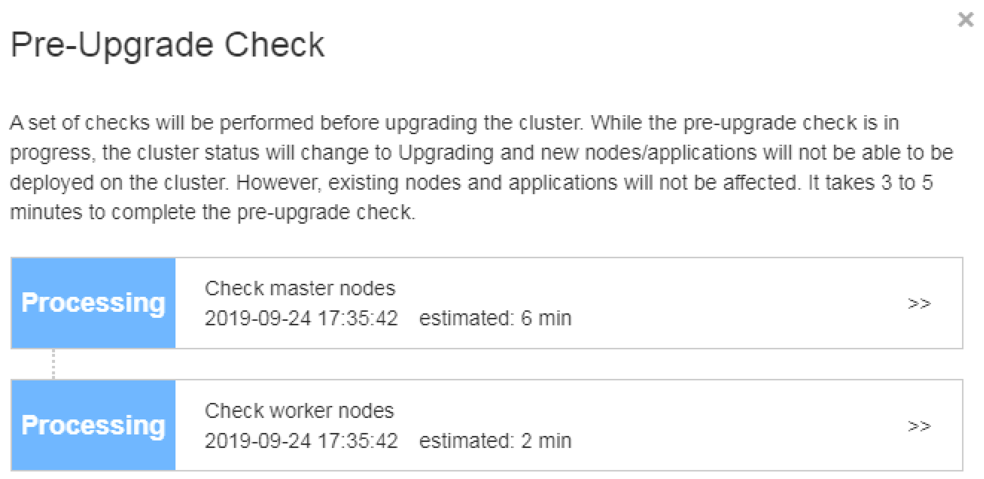


Pre-Upgrade Check

A set of checks will be performed before upgrading the cluster. While the pre-upgrade check is in progress, the cluster status will change to Upgrading and new nodes/applications will not be able to be deployed on the cluster. However, existing nodes and applications will not be affected. It takes 3 to 5 minutes to complete the pre-upgrade check.

Step 4 The pre-upgrade check starts. While the pre-upgrade check is in progress, the cluster status will change to **Upgrading** and new nodes/applications will not be able to be deployed on the cluster. However, existing nodes and applications will not be affected. It takes 3 to 5 minutes to complete the pre-upgrade check.

Figure 4-5 Pre-upgrade check in process



Pre-Upgrade Check

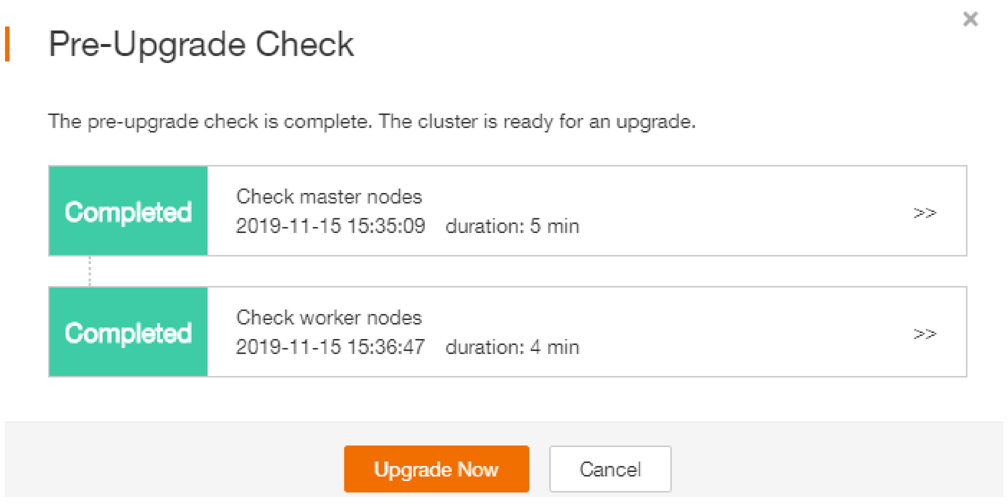
A set of checks will be performed before upgrading the cluster. While the pre-upgrade check is in progress, the cluster status will change to Upgrading and new nodes/applications will not be able to be deployed on the cluster. However, existing nodes and applications will not be affected. It takes 3 to 5 minutes to complete the pre-upgrade check.

Processing — Check master nodes — 2019-09-24 17:35:42 estimated: 6 min

Processing — Check worker nodes — 2019-09-24 17:35:42 estimated: 2 min

Step 5 When the status of the pre-upgrade check is Completed, click **Upgrade**.

Figure 4-6 Pre-upgrade check completed



Pre-Upgrade Check

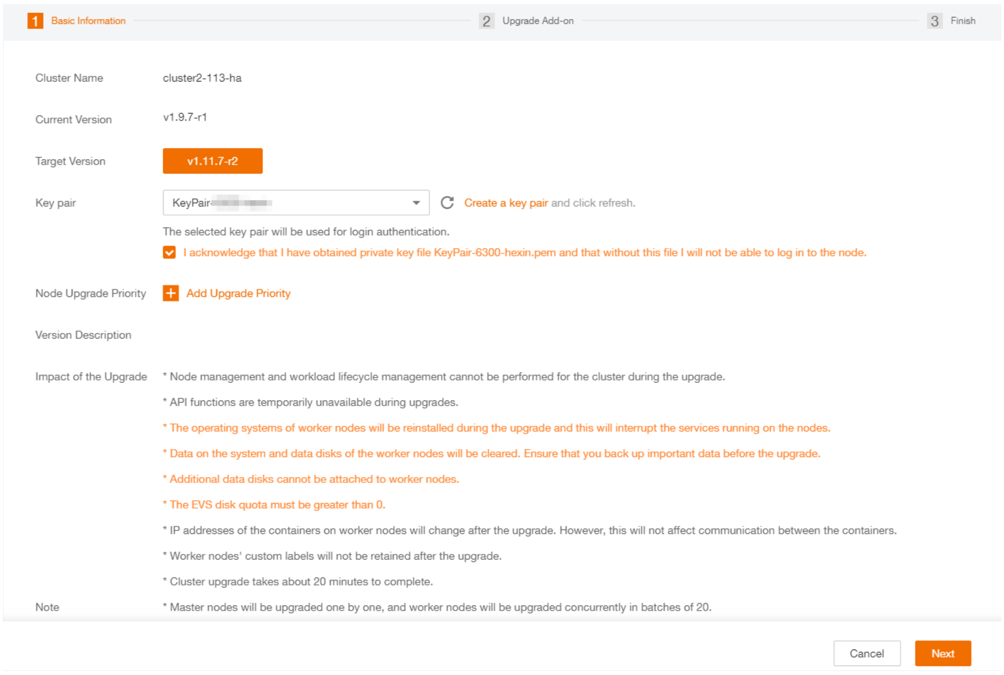
The pre-upgrade check is complete. The cluster is ready for an upgrade.

Completed — Check master nodes — 2019-11-15 15:35:09 duration: 5 min

Completed — Check worker nodes — 2019-11-15 15:36:47 duration: 4 min

Step 6 On the **Upgrade Cluster** page, confirm the basic information and click **Next**.

Figure 4-7 Basic information about cluster upgrade



Cluster Name	cluster2-113-ha
Current Version	v1.9.7-r1
Target Version	v1.11.7-r2
Key pair	KeyPair-[redacted] Create a key pair and click refresh. The selected key pair will be used for login authentication. ✓ I acknowledge that I have obtained private key file KeyPair-6300-hexin.pem and that without this file I will not be able to log in to the node.
Node Upgrade Priority	Add Upgrade Priority
Version Description	
Impact of the Upgrade	* Node management and workload lifecycle management cannot be performed for the cluster during the upgrade. * API functions are temporarily unavailable during upgrades. * The operating systems of worker nodes will be reinstalled during the upgrade and this will interrupt the services running on the nodes. * Data on the system and data disks of the worker nodes will be cleared. Ensure that you back up important data before the upgrade. * Additional data disks cannot be attached to worker nodes. * The EVS disk quota must be greater than 0. * IP addresses of the containers on worker nodes will change after the upgrade. However, this will not affect communication between the containers. * Worker nodes' custom labels will not be retained after the upgrade. * Cluster upgrade takes about 20 minutes to complete.
Note	* Master nodes will be upgraded one by one, and worker nodes will be upgraded concurrently in batches of 20.

Step 7 On the **Upgrade Add-on** page, click **Upgrade** in the lower right corner.

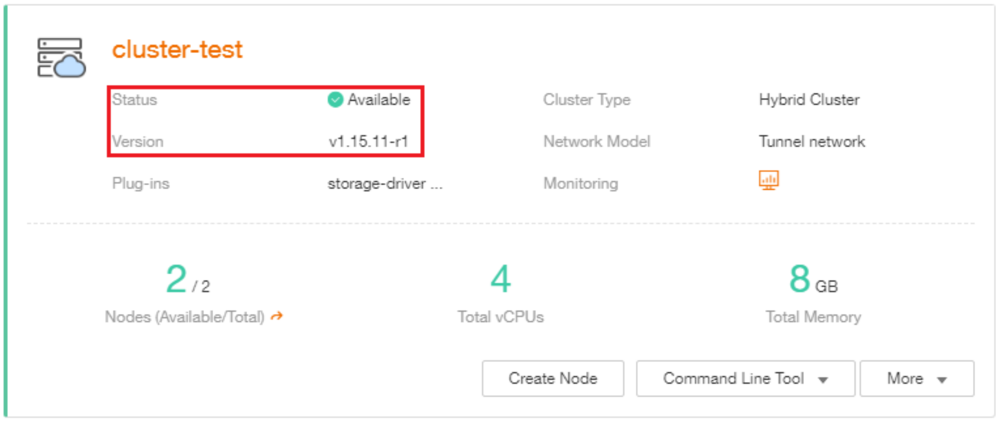
Step 8 In the displayed **Upgrade** dialog box, confirm the information and click **OK**. The upgrade cannot be rolled back.

Step 9 You can click **Back to Cluster List** or **Go to Cluster Details Page** to view the cluster upgrade status.

Step 10 Back to the cluster list, you can see that the cluster status is **Upgrading**. Wait until the upgrade is completed.

- After the upgrade is successful, you can view the cluster status and version on the cluster list or cluster details page.

Figure 4-8 Verifying the upgrade success



cluster-test

Status	Available	Cluster Type	Hybrid Cluster
Version	v1.15.11-r1	Network Model	Tunnel network
Plug-ins	storage-driver ...	Monitoring	

2 / 2	4	8 GB
Nodes (Available/Total)	Total vCPUs	Total Memory

- If the upgrade fails, perform **Step 2** to **Step 9** to upgrade the cluster again. If the fault persists, contact technical support for assistance.

----End

4.6 Cluster Auto Scaling

The Cluster Auto Scaling feature allows CCE to automatically scale out a cluster (add nodes to a cluster) according to custom policies when workloads cannot be scheduled into the cluster due to insufficient cluster resources.

NOTE

- Currently, master nodes cannot be automatically added to or removed from clusters.
- The Cluster Auto Scaling feature does not provide auto scale-down. You have to manually downsize a cluster based on resource usage.
- If both auto scale-up and auto scale-down are required, use the autoscaler add-on. For details, see **autoscaler**.
- Clusters of v1.17 and later do not support auto scaling using AOM. You can use node pools for auto scaling. For details, see **Node Pool Overview**.

Automatic Cluster Scaling-up

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane. In the card view of the cluster that needs auto scale-up, choose **More** > **Auto Scaling**.

Step 2 Click the **Scale-out Settings** tab and then **Edit**. Set the maximum number of nodes, cooldown period, and node configuration.

Table 4-7 scale-up settings

Parameter	Description
Cooldown Period	Interval between consecutive scale-up operations, in the unit of second. The cooldown period ensures that a scale-up operation is initiated only when previous scaling operation is finished and the system is running stably. The value ranges from 60 seconds to 3600 seconds. The default value is 900 seconds. If the value is less than 900 seconds, scale-out may not meet your expectation.
Maximum Nodes	Maximum number of nodes to which the cluster can scale out. 1 ≤ Maximum Nodes < cluster node quota **NOTE** The cluster node quota depends on the cluster size (maximum number of nodes that can be managed by a cluster) and the node quota of the account. The cluster node quota used here is the smaller of the two.
Node Configuration	If scale-up is required after the scale-up policy is executed, the system creates a node. 1. Click **Set** and set the node parameters. For details about how to set the node parameters, see **Step 2**. 2. Click **Now Config**.

Step 3 After confirming the scale-up configuration and node parameters, click **OK**.

Step 4 Set the scale-up policy for the cluster. Click the **Scaling-out Policies** tab and click **Add Scale-out Policy**.

- Policy Name: Enter a policy name, for example, **policy01**.
- Set **Policy Type**. Currently, the following types of auto scale-up policies are supported:
  - **Alarm Policy**: scale-up based on the CPU or memory settings

Table 4-8 Parameters for adding an alarm policy

Parameter	Description
*Metric	Select **Allocated CPU** or **Allocated Memory**.
*Trigger Condition	Set a condition for triggering a scale-up policy, that is, when the average CPU or memory allocation value is greater than or less than a specified percentage.
*Monitoring Window	Duration for which the metric is measured. Select a cluster from the drop-down list. If you select **15min**, the selected metric is measured every 15 minutes.
*Consecutive Occurrence Times	If you set this parameter to **3**, the action is triggered if the metrics meet the specified threshold for three consecutive times.
*Action	Action executed after a policy is triggered.

  - **Scheduled Policy**: scale-up at a specified time

Table 4-9 Parameters for adding a scheduled policy

Parameter	Description
*Policy Type	Set this parameter to **Scheduled Policy**.
*Trigger Time	Time at which the policy is triggered.
*Action	Action executed after a policy is triggered.

  - **Periodic Policy**: The scale-up policy can be performed by day, week, or month.

Table 4-10 Parameters for adding a periodic policy

Parameter	Description
*Policy Type	Set the parameter to **Periodic Policy**.
*Select Time	Specify the time for triggering the policy.

Parameter	Description
*Action	Action executed after a policy is triggered.

Step 5 Click **OK**.

After the auto scale-out is completed, choose **Resource Management** > **Nodes** in the navigation pane. On the node list, you can view the nodes added during cluster auto scaling.

----End

4.7 Deleting, Hibernating, and Waking Up a Cluster

After a cluster is created, you can delete, hibernate, or wake up the cluster.

Deleting a Cluster

NOTICE

Exercise caution when deleting a cluster because this operation will destroy the nodes in the cluster and running services.

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane.

Step 2 Choose **More > Delete**.

Step 3 The **Delete Cluster** dialog box then appears.

NOTE

- Deleting the cluster will delete all nodes in the cluster and the running workloads and services.
- The delete operation takes 1 to 3 minutes to complete.
- Enter **DELETE** into the text box below to confirm that the delete operation will continue.
- If a cluster whose status is Unavailable is deleted, some storage resources of the cluster may need to be manually deleted.

Step 4 Click **OK** to start deleting the cluster.

----End

Hibernating a Cluster

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

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

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane.

Step 2 Choose **More** > **Hibernate** for the target cluster.

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

NOTE

After the cluster is hibernated, resources such as the working node (ECS) to which the cluster belongs, bound EIP, and bandwidth continue to run. To shut down nodes, select **Stop all nodes in the cluster** in the dialog box or see **Stopping a Node**.

Step 4 When the cluster status changes from **Hibernating** to **Hibernation**, the cluster is hibernated.

----End

Waking Up a Cluster

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

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane.

Step 2 Choose **More** > **Wake**.

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

Wait until the cluster is woken up.

----End

4.8 Cluster Configuration Management

To better manage Kubernetes parameters in a cluster, the configuration management function is provided. This function allows you to perform in-depth configuration on core components.

Constraints

This function can be used only in clusters of v1.15 or later.

Procedure

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane.

Step 2 Choose **More > Configuration** next to the cluster to be configured.

Step 3 On the **Configuration** page on the right, change the values of the following Kubernetes parameters:

Table 4-11 Kubernetes parameters

Components	Parameters	Details	Default
kube-apiserver	default-not-ready-toleration-seconds	Default interval of the toleration for notReady: NoExecute.	300
	default-unreachable-toleration-seconds	Default interval of the toleration for unreachable: NoExecute.	300
	max-mutating-requests-inflight	Maximum number of mutating requests in flight at a given time.	1000
	max-requests-inflight	Maximum number of non-mutating requests in flight at a given time.	2000
	service-node-port-range	Range of node port numbers.	30000-32767
kube-controller-manager	concurrent-deployment-syncs	Number of Deployments that are allowed to synchronize concurrently.	5
	concurrent-endpoint-syncs	Number of endpoints that are allowed to synchronize concurrently.	5
	concurrent-gc-syncs	Number of garbage collector workers that are allowed to synchronize concurrently.	20
	concurrent-namespace-syncs	Number of namespaces that are allowed to synchronize concurrently.	10
	concurrent-replicaset-syncs	Number of ReplicaSets that are allowed to synchronize concurrently.	5
	concurrent-resource-quota-syncs	Number of resource quotas that are allowed to synchronize concurrently.	5
	concurrent-service-syncs	Number of Services that are allowed to synchronize concurrently.	10
	concurrent-serviceaccount-token-syncs	Number of service account tokens that are allowed to synchronize concurrently.	5

Components	Parameters	Details	Default
	concurrent-ttl-after-finished-syncs	Number of TTL-after-finished controller workers that are allowed to synchronize concurrently.	5
	concurrent_rc_syncs	Number of replication controllers that are allowed to synchronize concurrently.	5
	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	100
	kube-api-burst	Burst to use while talking with kube-apiserver.	100
kube-scheduler	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	100
	kube-api-burst	Burst to use while talking with kube-apiserver.	100

Step 4 Click **OK**.

----End

References

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

4.9 Obtaining a Cluster Certificate

Before accessing cluster resources through open-source Kubernetes APIs, obtain the cluster's certificate.

Constraints

- You can download the certificate of a newly created cluster. The certificate of an existing cluster cannot be downloaded.
- After a cluster is created, if you unbind EIP bound during cluster creation and bind a new EIP, the CA certificate does not update the IP address and still uses the previously bound IP address. As a result, the Kubernetes fails to be connected.

Procedure

Step 1 Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane.

Step 2 In the card view of the target cluster, choose **More > Download X.509 Certificate**.

Step 3 In the displayed **Download X.509 Certificate** dialog box, 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.

----End

4.10 Namespace

A namespace is a collection of resources and objects. Multiple namespaces can be created in a single cluster, but they are isolated from each other. This enables namespaces to share the services of the same cluster without affecting each other.

For example, you can deploy workloads in a development environment into one namespace, and deploy workloads in a test environment into another namespace.

Preparations

At least one cluster is created. For details, see **Creating a Hybrid Cluster**.

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 automatically or manually.

- Namespaces created automatically: When a cluster is up, the **default**, **kube-public**, and **kube-system** namespaces are created by default.
  - **default**: used if no namespace is specified.
  - **kube-public**: used for deploying public add-ons and container templates.
  - **kube-system**: used for deploying Kubernetes system components.
- Namespaces 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. In the navigation pane, choose **Resource Management** > **Namespaces**, and click **Create Namespace**.

Step 2 Set the parameters listed in **Table 4-12**. The parameters marked with an asterisk (*) are mandatory.

Table 4-12 Parameters for creating a namespace

Parameter	Description
*Namespace	Name of the namespace, which must be unique in a cluster.
*Cluster	Cluster to which the namespace belongs.
Node Affinity	If this parameter is set to on, workloads in the current namespace will be scheduled only to nodes with specified labels. To add labels to a node, choose **Resource Management** > **Nodes** > **Manage Labels**.
Description	Description of the namespace.
Set Resource Quotas	Set resource quotas to limit resource usage in the namespace, thereby organizing resources into different namespaces. **NOTICE** It is advised to assign a pod quota to each namespace. Without a pod quota, clusters or nodes may soon run out of system resources and exhibit unexpected behavior when too many pods are created. The pod quotas of all namespaces in a cluster must not exceed 110 (maximum pods per node) multiplied by the quantity of nodes in a cluster. For example, in a cluster of 50 nodes, a maximum of 5,500 (110 * 50) pods can be created. This means that pod quotas of all namespaces in the cluster should not exceed 5,500.

Step 3 Click **OK**.

----End

Using Namespaces

Step 1 When you create a workload, select a namespace for it.

Step 2 When you query workloads, select a namespace to view all workloads in the namespace.

----End

Namespace Application Scenarios

- **Dividing workloads into namespaces by environment type**

  Before being released, a workload generally goes through the phases of development, joint debugging, testing, and production. In this process, the workloads of different environments are logically defined but are basically the same. There are two methods:

– Creating clusters for different environments:

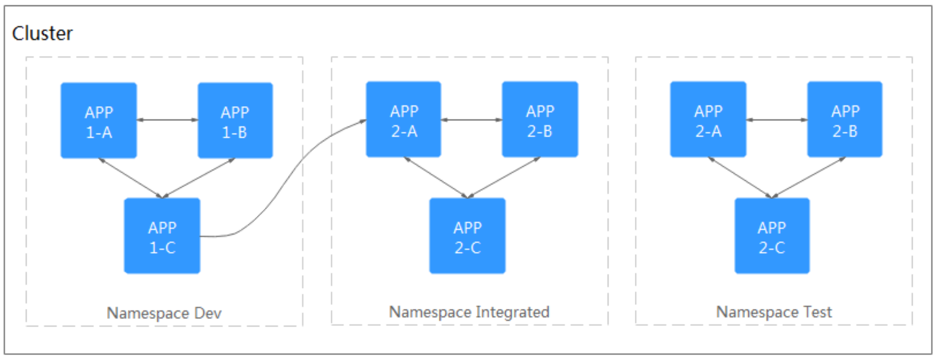
  Resources cannot be shared among different clusters. A load balancer is required in order to enable mutual access between services in different environments.

– Creating namespaces in the same cluster for different environments:

  Workloads in the same namespace access each other using service names, while workloads in different namespaces access each other using service names and namespace names.

  Figure 4-9 shows namespaces respectively created for the development, joint debugging, and testing environments.

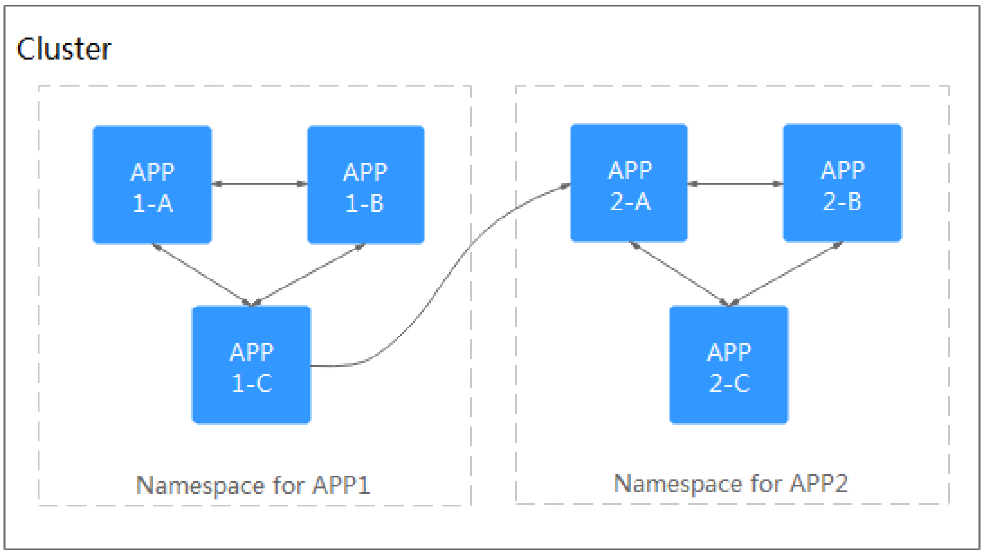
Figure 4-9 Dividing workloads into namespaces by environment types



- **Dividing workloads into namespaces by application**

  You are advised to use this method if a large number of workloads are deployed in the same environment. As shown in the following figure, different namespaces are created for App 1 and App 2. Workloads in a namespace are managed as a workload group. Workloads in the same namespace access each other using service names, while workloads in different namespaces access each other using service names and namespace names.

Figure 4-10 Dividing workloads into namespaces by workload type



Configuring a Namespace-level Network Policy

You can configure a namespace-level network policy by enabling network isolation.

By default, **Network Isolation** is disabled for namespaces. For example, if network isolation is off for namespace **default**, **all workloads in the current cluster** can access the **workloads in namespace default**.

To prevent other pods from accessing the pods in namespace **default**, perform the following steps:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Namespaces**.

Step 2 In the same row as the namespace (for example, **default**) for which you will create a network policy, enable network isolation.

After network isolation is enabled, workloads in namespace **default** can access each other but they cannot be accessed by workloads in other namespaces.

NOTE

After the network policy is set, network isolation may be damaged.

----End

Configuring Namespace-level Resource Quotas

Namespace-level resource quotas limit the total numbers of resources that can be used when multiple teams or users share cluster resources. 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.

NOTE

This function is supported only when the cluster version is 1.9 or later.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Namespaces**.

Step 2 In the **Operation** column of a namespace, click **Manage Quota**.

This operation cannot be performed on system namespaces **kube-system** and **kube-public**.

Step 3 Set the following resource quotas and click **OK**:

- **CPU (cores)**: Maximum number of CPU cores that can be allocated to workload pods in the namespace. Unit: cores.
- **Memory (MiB)**: Maximum amount of memory that can be allocated to workload pods in the namespace. Unit: MiB.
- **StatefulSet**: Maximum number of StatefulSets that can be created in the namespace.
- **Deployment**: maximum number of Deployments that can be created in the namespace.
- **Job**: maximum number of one-off jobs that can be created under the namespace.
- **Cron Job**: maximum number of cron jobs that can be created in the namespace.
- **Pod**: maximum number of pods that can be created in the namespace.
- **Service**: maximum number of Services that can be created in the namespace.

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**, no limit is posed on the resource.
- Accumulated quota usage includes the default resources created by CCE, such as the kubernetes service (view this service using the kubectl tool) created under the **default** namespace. Therefore, you are advised to set a resource quota greater than what you expect.

----End

Deleting Namespaces

If a namespace is deleted, all resources (such as workloads, one-off jobs, and ConfigMaps) in this namespace will be also deleted. Exercise caution when deleting a namespace.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Namespaces**.

Step 2 In the **Clusters** drop-down list, select the cluster where the namespace to be deleted is located.

Step 3 Select the namespace to be deleted and click **Delete**.

Follow the prompts to delete it. The built-in namespaces of the system cannot be deleted.

----End

4.11 Cluster Management Permission Control

To perform permission control on resources in a cluster (for example, user A can only read and write applications in a namespace, while user B can only read resources in a cluster), perform the operations described in this section.

Procedure

Step 1 To perform permission control on a cluster, select **Enhanced authentication capability** for **Authentication Mode** and then select **Authenticating Proxy**. Click **Upload** next to **CA root certificate** to upload a qualified and valid certificate. For details, see **Table 4-3**.

Step 2 Create a role using kubectl.

The following example shows how to create a **role** and allow the role to read all pods in the default namespace. For details about the parameters, see the **official Kubernetes documentation**.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

Step 3 Bind the role to a user by using kubectl.

In the following example, the **RoleBinding** assigns the role of **pod-reader** in the default namespace to user **jane**. This policy allows user **jane** to read all pods in the default namespace. For details about the parameters, see the **official Kubernetes documentation**.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane #User name
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader    #Name of the role that is created
  apiGroup: rbac.authorization.k8s.io
```

Step 4 After a role is created and bound to a user, call a Kubernetes API by initiating an API request message where headers carry user information and the certificate uploaded during cluster creation. For example, to call the pod query API, run the following command:

curl -H "X-Remote-User: ***jane*****" --cacert /root/tls-ca.crt --key /root/tls.key --cert /root/tls.crt https://***192.168.23.5:5443***/api/v1/namespaces/default/pods**

If **200** is returned, user **jane** is authorized to read pods in the cluster's default namespace. If **403** is returned, user **jane** is not authorized to read pods in the cluster's default namespace.

NOTE

To prevent the command execution failure, upload the certificate to the **/root** directory in advance.

The parameter descriptions are as follows:

- **X-Remote-User: *jane***: The request header is fixed at **X-Remote-User**, and **jane** is the username.
- ***tls-ca.crt***: CA root certificate uploaded during cluster creation.
- **tls.crt**: Client certificate that matches the CA root certificate uploaded during cluster creation.
- **tls.key**: Client key corresponding to the CA root certificate uploaded during cluster creation.
- **192.168.23.5:5443**: address for connecting to the cluster. To obtain the address, perform the following steps:

  Log in to the CCE console. In the navigation pane, choose **Resource Management > Clusters**. Click the name of the cluster to be connected and obtain the IP address and port number in **Internal API Server Address**.

In addition, the **X-Remote-Group** header field, that is, the user group name, is supported. During role binding, a role can be bound to a group and carry user group information when you access the cluster.

----End

5 Node Management

5.1 Creating a Node

A node is a virtual or physical machine that provides computing resources. Sufficient nodes must be available in your project to ensure that operations, such as creating workloads, can be performed.

Preparations

- At least one cluster is available. For details on how to create a cluster, see **Creating a Hybrid Cluster**.
- Create a key pair for identity authentication upon remote node login.

Precautions for Using a Node

Some of the node resources will be used to run necessary Kubernetes system components and resources to make the node as part of your cluster. Therefore, the total number of node resources and the amount of allocatable node resources for your cluster are different. The larger the node specifications, the more the containers deployed on the node. Therefore, more node resources need to be reserved to run Kubernetes components.

To ensure node stability, a certain amount of CCE node resources will be reserved for Kubernetes components (such as kubelet, kube-proxy, and docker) based on the node specifications.

NOTE

You are advised not to install private software or modify the operating system (OS) configuration on a cluster node. This may cause exceptions on Kubernetes components installed on the node, and make the node unavailable.

Creating Nodes

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Clusters**. Click **Create Node**.

Step 2 Select a region and an AZ.

- **Current Region**: Select the geographic location of the node to be created.
- **AZ**: Select the AZ where the node is to be deployed. An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

  To enhance workload availability, create nodes in different AZs.

Step 3 Configure the node specifications and quantity.

- **Node type**: Currently, only VM nodes are supported.
- **Node Name**: name of the new node. A node name contains 1 to 56 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **Specifications**: Select node specifications that best fit your business needs.
  - **General-purpose**: provides a balance of computing, memory, and network resources. It is a good choice for many applications. General-purpose nodes can be used for web servers, workload development, workload testing, and small-scale databases.
  - **Memory-optimized**: provides higher memory capacity than general-purpose nodes and is suitable for relational databases, NoSQL, and other workloads that are both memory-intensive and data-intensive.
  - **GPU-accelerated**: provides powerful floating-point computing and is suitable for real-time, highly concurrent massive computing. Graphical processing units (GPUs) of P series are suitable for deep learning, scientific computing, and CAE. GPUs of G series are suitable for 3D animation rendering and CAD.
  - **General computing-plus**: provides stable performance and exclusive resources to enterprise-class workloads with high and stable computing performance.
  - **Ultra-high I/O**: provides ultra-low SSD access latency and ultra-high IOPS performance. This type of specifications is suitable for high-performance relational databases, NoSQL databases (such as Cassandra and MongoDB), and Elasticsearch.

  To ensure node stability, the system automatically reserves some resources for running necessary system components. For details, see the **formula for calculating reserved node resources**.
- **OS**: Select the operating system (OS) of the nodes to be created. For details, see **Cluster Node OS Patch Notes**.
- **VPC:** This parameter is available only for clusters of v1.13.10-r0 or later.
- **Subnet**: A subnet improves network security by providing exclusive network resources that are isolated from other networks. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets. This parameter is displayed only for clusters of v1.13.10-r0 and later.

Step 4 **System Disk**: Set the system disk space of the worker node. The value ranges from 40 GB to 1024 GB. The default value is 40 GB.

By default, the following types of EVS disks are supported:

- **Common I/O**: EVS disks of this type deliver a maximum of 2,200 IOPS. This disk type is suitable for application scenarios that require large capacity, a medium read/write rate, and fewer transactions, such as enterprise office applications and small-scale testing.

- **High I/O**: EVS disks of this type deliver a maximum of 5,000 IOPS and a minimum of 1 ms read/write latency. This disk type is designed to meet the needs of mainstream high-performance, high-reliability application scenarios, such as enterprise applications, small- and medium-scale development and testing, and web server logs.
- **Ultra-high I/O**: EVS disks of this type deliver a maximum of 33,000 IOPS and a minimum of 1 ms read/write latency. This disk type is excellent for ultra-high I/O, ultra-high bandwidth, and read/write-intensive application scenarios, such as distributed file systems in HPC scenarios or NoSQL/RDS in I/O-intensive scenarios.

Step 5 **Data Disk**: Set the data disk space of the worker node. The value ranges from 100 GB to 32,678 GB. The types of EVS disks supported for data disks are the same as those for the system disk.

CAUTION

If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk.

- **LVM**: If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space allocation for different resources.
  - This option is selected by default, indicating that LVM management is enabled.
  - You can deselect the check box to disable LVM management.

  CAUTION

  - When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in **Adding a Second Data Disk to a Node in a CCE Cluster** to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM.
  - When creating a node in a cluster of v1.13.10 or earlier, if a data disk is not managed by LVM, format the data disk. Otherwise, either this data disk or the first data disk will be managed by LVM, which is not as expected.
- **Add Data Disk**: Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to attach more data disks.
- **Data disk space allocation**: Click **Change Configuration** to specify the resource ratio for **Kubernetes Space** and **User Space**.
  - **Kubernetes Space**: You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include Docker

images and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs.

- **User Space**: You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted. **Path inside a node** cannot be set to the root directory **/**. Otherwise, the mounting fails.

NOTE

Disk space of the data disks managed by LVM will be allocated according to the ratio you set.

If the cluster version is v1.13.10-r0 or later and the node type is ultra-high I/O, the following options are displayed for data disks:

- **EVS**: The disk capacity ranges from 100 to 32,678 GB. You can set a value as required.

  If you select **Data disk space allocation**, you can allocate data disk space for storing Docker and Kubelet resources.
- **Local disk**: This value is displayed only for **ultra-high I/O** nodes in clusters of v1.13.10-r0 or later. Local disks may break down and do not ensure data reliability. It is recommended that you store service data in EVS disks, which are more reliable than local disks. Local disk parameters are as follows:
  - **Disk Mode**: When the node type is **ultra-high I/O**, solid state disks (SSDs) are supported.
  - **Kubernetes Space**: the percentage of data disk space allocated for storing Docker and Kubelet resources. Docker resources include Docker images and image metadata; Kubelet resources include pod configuration files, secrets, and EmptyDirs.
  - **User Space**: You can specify the ratio of the local disk space that is not allocated for storing Kubernetes resources.

NOTICE

By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability. For more information, click **here**.

Step 6 **EIP**: An independent public IP address. If the nodes to be created require Internet access, select **Automatically assign** or **Use existing**.

NOTE

By default, VPC's SNAT feature is disabled for CCE. If SNAT is enabled, you do not need to use EIPs to access public networks.

- **Do not use**: A node without an EIP cannot be accessed from public networks. It can be used only as a cloud server for deploying services or clusters on a private network.
- **Automatically assign**: An EIP with specified configurations is automatically assigned to each node. If the number of EIPs is less than the number of nodes, the EIPs are randomly bound to the nodes.

Set the specifications, required quantity, billing mode, and bandwidth as required. When creating an ECS, ensure that the EIP quota is sufficient.

- **Use existing**: Existing EIPs are assigned to the nodes to be created.

Step 7 **Login Mode**: If the login mode is **Key pair**, select a key pair for logging to the node.

A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair**.

Step 8 **Advanced ECS Settings**: (Optional) Click ⌄ to show advanced ECS settings.

- **ECS Group**: Select an existing ECS group, or click **Create ECS Group** to create a new one. After the ECS group is created, click the refresh icon.

  An ECS group allows you to create ECSs on different hosts, thereby improving service reliability.
- **Resource Tags**: Resource tags can be added to classify resources.

  You can create **predefined tags** in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency.

  CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag. A maximum of 5 tags can be added.
- **Agency**: An agency is created by the account administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
- **Pre-installation Script**: Enter a maximum of 1,000 characters.

  The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may not be installed successfully. The script is usually used to format data disks.
- **Post-installation Script**: Enter a maximum of 1,000 characters.

  The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
- **Subnet IP Address**: Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.

Step 9 **Advanced Kubernetes Settings**: (Optional) Click ⌄ to show advanced Kubernetes settings.

- **Max Pods**: Maximum number of pods that can be created on a node, including the system's default pods. Value range: 16 to 250.

  This limit prevents the node from being overloaded by managing too many pods.
- **insecure-registries**: Click **Add insecure-registry** and enter the address of the image repository.

  Add the address of the custom image repository with no valid SSL certificate to the docker startup option to avoid unsuccessful image pulling from the personal image repository. The address is in the format of IP address:Port number (or domain name). Post-installation script and insecure-registries cannot be used together.

- **Maximum Data Space per Container**: maximum data space that can be used by a container. The value ranges from 10 GB to 80 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.

Step 10 Click **Next: Confirm**. After confirming that the configuration information is correct, click **Submit**. It takes 6 to 10 minutes to create a node.

Step 11 Click **Back to Node List**. The node has been created successfully if it is in the Available state.

----End

5.2 Logging In to a Node

Constraints

- If you use SSH to log in to a node (an ECS), ensure that the ECS already has an EIP (a public IP address).
- Only login in to a running ECS is allowed.
- Only the user **cloud** can log in to a Linux ECS.

Login Modes

You can log in to an ECS in either of the following modes:

- Management console (VNC)

  If an ECS has no EIP, log in to the ECS console and click **Remote Login** in the same row as the ECS.
- SSH

  This mode applies only to ECSs running Linux. Usually, you can use a remote login tool, such as PuTTY, Xshell, and SecureCRT, to log in to your ECS. If none of the remote login tools can be used, log in to the ECS console and click **Remote Login** in the same row as the ECS to view the connection status and running status of the ECS.

  NOTE

  - When you use the Windows OS to log in to a Linux node, set **Auto-login username** to **cloud**.
  - The CCE console does not support node OS upgrade. Do not upgrade the node OS using YUM. Otherwise, the container network will be unavailable.

Table 5-1 Linux ECS login modes

EIP Binding	On-Premises OS	Connection Method
Yes	Windows	Use remote login tools such as PuTTY and Xshell. For details, see "Logging In to the Linux ECS from Local Windows" in Login Using an SSH Key.
Yes	Linux	Run commands. For details, see "Logging In to the Linux ECS from Local Linux" in Login Using an SSH Key.
Yes/No	Windows/ Linux	Use VNC.

5.3 Deleting a Node

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

NOTICE

After a CCE cluster is deleted, the ECS nodes in the cluster are also deleted.

Background

- Deleting a node will lead to pod migration, which may affect services. Perform this operation during off-peak hours.
- Unexpected risks may occur during the operation. Back up related data in advance.
- During the operation, the backend will set the node to the unschedulable state.
- Only worker nodes can be deleted, and master nodes cannot.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**. In the same row as the node you will delete, choose **More** > **Delete**.

Step 2 In the **Delete Node** dialog box, enter **DELETE** and click **Yes**.

NOTE

After the node is deleted, workload pods on the node are automatically migrated to other available nodes.

----End

5.4 Stopping a Node

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

Constraints

- Deleting a node will lead to pod migration, which may affect services. Therefore, delete nodes during off-peak hours.
- Unexpected risks may occur during node deletion. Back up related data in advance.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- Only worker nodes can be stopped.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

Step 2 In the node list, click the name of the node to be stopped. The node details page is displayed.

Step 3 Click the node name to open the ECS details page on the Cloud Server Console.

Step 4 In the upper right corner of the ECS details page, click **Stop**. In the **Stop ECS** dialog box, click **Yes**.

----End

5.5 Synchronizing Node Data

Context

Each node in a cluster is a cloud server. After a cluster node is created, you can change the cloud server name or specifications as required.

After you change a cloud server **name** or specifications on the CCE console, use the **node data synchronization function** to synchronize the new name or specifications from the ECS console to the CCE console.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

Step 2 In the same row as the node whose data will be synchronized, choose **More** > **Sync Node data**.

NOTE

Alternatively, click the node name, and click **Sync Node Data** in the upper right corner of the node details page.

After the synchronization is complete, the **Sync success** message is displayed in the upper right corner.

----End

5.6 Resetting a Node

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

Constraints

This function is supported for clusters of v1.13 and later.

Precautions

- **Resetting a node will interrupt workload services running on the node. Therefore, perform this operation during off-peak hours.**
- **Data in the system and data disks will be cleared. Back up important data before resetting the node.**
- **You are not allowed to manually attach additional data disks to nodes.**
- The IP addresses of the workload pods on the node will change, but the container network access is not affected.
- There is remaining EVS disk quota.
- While the node is being deleted, the backend will set the node to the unschedulable state.
- Only worker nodes can be reset. If the node is still unavailable after the resetting, delete the node and create a new one.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**. In the same row as the node you will reset, choose **More > Reset**.

Step 2 In the **Reset** dialog box displayed, enter **RESET** and set **Key pair**.

Step 3 Click **Yes** and wait until the node is reset.

After the node is reset, pods on it are automatically migrated to other available nodes.

----End

5.7 Managing Node Labels

You can add different labels to nodes and define different attributes for labels. By using these node labels, you can quickly understand the characteristics of each node.

Node Label Usage Scenario

Node labels are mainly used in the following scenarios:

- Node management: Node labels are used to classify nodes.
- Affinity and anti-affinity between a workload and node:
  - Some workloads require a large CPU, some require a large memory, some require a large I/O, and other workloads may be affected. In this case, you are advised to add different labels to nodes. When deploying a workload, you can select node affinity deployment of the corresponding label to ensure the normal operation of the system. Otherwise, node anti-affinity deployment can be used.
  - A system can be divided into multiple modules. Each module consists of multiple microservices. To ensure the efficiency of subsequent O&M, you can add a module label to each node so that each module can be deployed on the corresponding node, does not interfere with other modules, and can be easily developed and maintained on its node.

Inherent Node Labels

After a node is created, some fixed labels exist and cannot be deleted. For details about these labels, see **Table 5-2**.

Table 5-2 Inherent labels of a node

Key	Value
failure-domain.beta.kubernetes.io/is-baremetal	Indicates whether the node is a bare metal node. **false** indicates that the node is not a bare-metal node.
failure-domain.beta.kubernetes.io/region	Indicates the area where the node is located.
failure-domain.beta.kubernetes.io/zone	Indicates the AZ where the node is located.
node.kubernetes.io/subnetid	Indicates a subnet ID.
os.architecture	Indicates the processor architecture of a node. For example, **amd64** indicates a AMD64-bit processor.

Key	Value
os.name	Indicates the operating system name of a node. For example, **EulerOS_2.0_SP2** indicates the version of Euler 2.2.
os.version	Indicates the kernel version of a node.

Adding a Node Label

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

Step 2 In the same row as the node for which you will add labels, choose **Operation** > **Manage Labels**.

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

As shown in the figure, the key is **deploy_qa** and the value is **true**, indicating that the node is used to deploy the QA (test) environment.

Step 4 After the label is added, click **Manage Labels**, then you will see the label that you have added.

----End

Deleting a Node Label

Only labels added by users can be deleted. Labels that are fixed on the node cannot be deleted.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

Step 2 In the same row as the node for which you will delete labels, choose **Operation** > **Manage Labels**.

Step 3 Click **Delete**, and then click **OK** to delete the label.

Label updated successfully is displayed.

----End

5.8 Monitoring a Node

CCE is seamlessly integrated with the monitoring service Cloud Eye. You can learn the resource usage of the worker nodes in each cluster and monitoring metrics of ECSs in the cluster in real time and receive alarms in a timely manner to ensure smooth service running.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management > Nodes**.

Step 2 Select a cluster in the upper right corner of the **Nodes** page.

Step 3 In the same row of the node to be monitored, click **Monitoring** to switch to the Cloud Eye console and view the monitoring information.

Step 4 View the basic monitoring information about the node, including the CPU usage, inbound bandwidth, outbound bandwidth, system disk BPS, and system disk IOPS.

----End

5.9 Formula for Calculating the Reserved Resources of a Node

Some of the resources on the node need to run some necessary Kubernetes system components and resources to make the node as part of your cluster. Therefore, the total number of node resources and the number of assignable node resources in Kubernetes are different. The larger the node specifications, the more the containers deployed on the node. Therefore, Kubernetes needs to reserve more resources.

To ensure node stability, CCE cluster nodes reserve some resources for Kubernetes components (such as kubelet, kube-proxy, and docker) based on node specifications.

CCE calculates the resources that can be allocated to user nodes as follows:

Allocatable = Capacity - Reserved - Eviction Threshold

That is, **Configurable amount on the node** = **Total amount – Reserved amount – Eviction threshold**.

- The rules for reserving the node memory are as follows:
  a. total_mem ≤ 4 GB, reserved_value = total_mem x 25%
  b. 4 GB < total_mem ≤ 8 GB, reserved_value = 4 GB x 25% + (total_mem – 4 GB) x 20%
  c. 8 GB < total_mem ≤ 16 GB, reserved_value = 4 GB x 25% + 4 GB x 20% + (total_mem – 8 GB) x 10%
  d. 16 GB < total_mem ≤ 128 GB, reserved_value = 4 GB x 25% + 4 GB x 20% + 8 GB x 10% + (total_mem – 16 GB) x 6%
  e. total_mem > 128 GB, reserved_value = 4 GB x 25% + 4 GB x 20% + 8 GB x 10% + 112 GB x 6% + (total_mem – 128 GB) x 2%

  In the preceding information, **total_mem** indicates the total memory and **reserved_value** indicates the reserved memory.
- The rules for reserving the node CPU are as follows:
  a. total_cpu ≤ 1 core, reserved_value = total_cpu x 6%
  b. 1 core < total_cpu ≤ 2 core, reserved_value = 1 core x 6% + (total_cpu – 1 core) x 1%
  c. 2 core < total_cpu ≤ 4 core, reserved_value = 1 core x 6% + 1 core x 1% + (total_cpu – 2 core) x 0.5%
  d. total_cpu > 4 core, reserved_value = 1 core x 6% + 1 core x 1% + 2 core x 0.5% + (total_cpu – 4 core) x 0.25%

In the preceding information, **total_cpu** indicates the total CPU, and **reserved_value** indicates the reserved CPU.

- CCE reserves an extra 100Mi for kubelet eviction.

6 Node Pool Management

6.1 Node Pool Overview

This section describes how node pools work in CCE and how to create and manage node pools.

User Monitoring

A node pool contains one node or a group of nodes with identical configuration in a cluster.

In CCE, the nodes configured during cluster creation are grouped into the default node pool. This node pool is named **DefaultPool** and cannot be edited, deleted, migrated, expanded, or auto scaled.

On the CCE console, you can create custom node pools in a cluster to organize cluster nodes into different pools so that you can edit or delete a node pool individually without affecting the entire cluster. All nodes in a custom node pool have identical parameters and node type. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool. For details on how to control scheduling of workload pods onto nodes, see **Scheduling Policy Overview**.

Deploying a Workload in a Specified Node Pool

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

For example, you can choose **Edit YAML** on the workload page of the CCE console to set nodeSelector to forcibly deploy the workload in a specific node pool so that the workload runs only on nodes in the node pool.

For example, you can use container's resource request as a nodeSelector so that workloads will run only on the nodes that meet the resource request.

If the workload definition file defines a container that requires four CPUs, the scheduler will not choose the nodes with two CPUs to run workloads.

Nodes in a Multi-AZ Cluster

If you create a multi-AZ cluster, all node pools in the cluster will be automatically replicated in these AZs. The system automatically creates node pools in these AZs. Similarly, if you delete a node pool in an AZ, the node pool is also deleted from other AZs.

6.2 Creating a Node Pool

This topic describes how to create a node pool for a running CCE cluster and how to perform operations on the node pool. For details about how a node pool works, see **Node Pool Overview**.

Procedure

To create a node pool in a cluster, perform the following steps:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**.

Step 2 In the upper right corner of the page, click **Create Node Pool**.

Step 3 On the **Create Node Pool** page, configure node pool parameters listed in **Table 6-1**. Parameters with an asterisk (*) are mandatory.

Table 6-1 Parameters for creating a node pool

Parameter	Description
*Current Region	Physical region where the nodes are located. To minimize network latency and resource access time, select the region nearest to your node pool. Cloud resources are region-specific and cannot be used across regions over internal networks.
*Name	Name of the new node pool. By default, the name is in the format of *Cluster name*-nodepool-*Random number*. If you do not want to use the default name format, you can customize the name.
*Node Type	Currently, only VM nodes are supported.
*Nodes	Number of nodes created in the node pool. The value cannot exceed the maximum number of nodes that can be managed by the cluster. Set this parameter based on service requirements and the message displayed on the page.

Parameter	Description
Autoscaler	By default, this parameter is disabled. After you enable autoscaler by clicking ⚪, nodes in the node pool will be automatically created or deleted based on service requirements. ● **Maximum Nodes** and **Minimum Nodes**: You can set the maximum and minimum number of nodes to ensure that the number of nodes to be scaled is within a proper range. ● Priority: Set this parameter based on service requirements. A larger value indicates a higher priority. For example, if this parameter is set to **1** and **4** respectively for node pools A and B, B has a higher priority than A. If the priorities of multiple node pools are set to the same value, for example, **2**, the node pools are not prioritized and the system performs scaling based on the minimum resource waste principle. ● **Scaling-In Cooling Interval**: Set this parameter in the unit of minute. This parameter indicates the interval between the previous scale-out action and the next scale-in action. If **Autoscaler** is enabled, install the **autoscaler add-on** to use the auto scaling feature.
*AZ	An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. Set an AZ based on your requirements. To enhance workload reliability, you are advised to select **Random AZ**, allowing nodes to be randomly and evenly distributed among different AZs.

Parameter	Description
*Specifications	Select node specifications that best fit your business needs. ● **General-purpose**: provides a balance of computing, memory, and network resources. It is a good choice for many applications. General-purpose nodes can be used for web servers, workload development, workload testing, and small-scale databases. ● **Memory-optimized**: provides higher memory capacity than general-purpose nodes and is suitable for relational databases, NoSQL, and other workloads that are both memory-intensive and data-intensive. ● **GPU-accelerated**: provides powerful floating-point computing and is suitable for real-time, highly concurrent massive computing. Graphical processing units (GPUs) of P series are suitable for deep learning, scientific computing, and CAE. GPUs of G series are suitable for 3D animation rendering and CAD. ● **General computing-plus**: provides stable performance and exclusive resources to enterprise-class workloads with high and stable computing performance. ● **Ultra-high I/O**: provides ultra-low SSD access latency and ultra-high IOPS performance. This type of specifications is suitable for high-performance relational databases, NoSQL databases (such as Cassandra and MongoDB), and Elasticsearch. **NOTICE** To ensure node stability, the system automatically reserves some resources for running necessary system components. For details, see **Formula for Calculating the Reserved Resources of a Node**.
*OS	Select an operating system for the node pool. For details, see **Cluster Node OS Patch Notes**.
VPC	The node pool inherits VPC settings from the cluster to which it belongs. **This parameter is displayed only for clusters of v1.13.10-r0 and later.**
*Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets. **This parameter is displayed only for clusters of v1.13.10-r0 and later.**

Parameter	Description
*System Disk	Set the system disk space of the worker node. The value ranges from 40 GB to 1024 GB. The default value is 40 GB. By default, the following types of EVS disks are supported: ● **Common I/O**: EVS disks of this type deliver a maximum of 2,200 IOPS. This disk type is suitable for application scenarios that require large capacity, a medium read/write rate, and fewer transactions, such as enterprise office applications and small-scale testing. ● **High I/O**: EVS disks of this type deliver a maximum of 5,000 IOPS and a minimum of 1 ms read/write latency. This disk type is designed to meet the needs of mainstream high-performance, high-reliability application scenarios, such as enterprise applications, small- and medium-scale development and testing, and web server logs. ● **Ultra-high I/O**: EVS disks of this type deliver a maximum of 33,000 IOPS and a minimum of 1 ms read/write latency. This disk type is excellent for ultra-high I/O, ultra-high bandwidth, and read/write-intensive application scenarios, such as distributed file systems in HPC scenarios or NoSQL/RDS in I/O-intensive scenarios.

Parameter	Description
*Data Disk	Set the data disk space of the worker node. The value ranges from 100 GB to 32,678 GB. The types of EVS disks supported for data disks are the same as those for the system disk. **CAUTION** If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable. You are advised not to delete the data disk. **LVM**: If this option is selected, CCE data disks are managed by the Logical Volume Manager (LVM). On this condition, you can adjust the disk space allocation for different resources. ● This option is selected by default, indicating that LVM management is enabled. ● You can deselect the check box to disable LVM management. **CAUTION** – When creating a node in a cluster of v1.13.10 or later, if LVM is not selected for a data disk, follow instructions in **How Do I Add a Second Data Disk to a Node in a CCE Cluster?** to fill in the pre-installation script and format the data disk. Otherwise, the data disk will still be managed by LVM. – When creating a node in a cluster of v1.13.10 or earlier, if a data disk is not managed by LVM, format the data disk. Otherwise, either this data disk or the first data disk will be managed by LVM, which is not as expected. **Add Data Disk**: Currently, a maximum of two data disks can be attached to a node. After the node is created, you can go to the ECS console to attach more data disks. **Data disk space allocation**: Click **Change Configuration** to specify the resource ratio for **Kubernetes Space** and **User Space**. ● **Kubernetes Space**: You can specify the ratio of the data disk space for storing Docker and kubelet resources. Docker resources include Docker images and image metadata. kubelet resources include pod configuration files, secrets, and emptyDirs. ● **User Space**: You can set the ratio of the disk space that is not allocated to Kubernetes resources and the path to which the user space is mounted. **Path inside a node** cannot be set to the root directory **/**. Otherwise, the mounting fails. **NOTE** Disk space of the data disks managed by LVM will be allocated according to the ratio you set. **If the cluster version is v1.13.10-r0 or later and the node type is ultra-high I/O, the following options are displayed for data disks:** ● **EVS**: The disk capacity ranges from 100 to 32,678 GB. You can set a value as required. If you select **Data disk space allocation**, you can allocate data disk space for storing Docker and Kubelet resources.

Parameter	Description
	● **Local disk**: This value is displayed only for **ultra-high I/O** nodes in clusters of v1.13.10-r0 or later. Local disks may break down and do not ensure data reliability. It is recommended that you store service data in EVS disks, which are more reliable than local disks. Local disk parameters are as follows: – **Disk Mode**: When the node type is ultra-high I/O, solid state disks (SSDs) are supported. – **Kubernetes Space**: the percentage of data disk space allocated for storing Docker and kubelet resources. Docker resources include Docker images and image metadata; kubelet resources include pod configuration files, secrets, and emptyDirs. – **User Space**: You can specify the ratio of the local disk space that is not allocated for storing Kubernetes resources. **NOTICE** By default, disks run in the direct-lvm mode. If data disks are removed, the loop-lvm mode will be used and this will impair system stability. For more information, click **here**.
*Key Pair	A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair** and create one.

Step 4 **Advanced ECS Settings** (optional): Click ⌄ to show advanced ECS settings.

- **ECS Group**: Select an existing ECS group, or click **Create ECS Group** to create a new one. After the ECS group is created, click the refresh icon.

  NOTE

  An ECS group allows you to create ECSs on different hosts, thereby improving service reliability.

- **Resource Tags**: Resource tags can be added to classify resources.

  NOTE

  – You can create **predefined tags** in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency.
  – CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*node id*" tag.
  – A maximum of 5 tags can be added.

- **Agency**: An agency is created by the account administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
- **Pre-installation Script**: Enter the installation script.

NOTE

- The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may not be installed successfully. The script is usually used to format data disks.
- Enter a maximum of 1,000 characters.
- The script is written into nodes through **command line insertion**. For details about the command line injection example, see **How Do I Format a Data Disk Using Command Line Injection?**.

- **Post-installation Script**: Enter the installation script.

  NOTE

  - The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
  - Enter a maximum of 1,000 characters.
  - The script is written into nodes through **command line insertion**. For details about example scripts and how to inject data, see **Pre-installation Script**.

Step 5 **Advanced Kubernetes Settings**: (Optional) Click ⌄ to show advanced Kubernetes settings.

- **Max Pods**: The maximum number of pods that can be created on a node, including the system's default pods. Value range: 16 to 250.

  NOTE

  This limit prevents the node from being overloaded by managing too many pods.

- **insecure-registries**: Click **Add insecure-registry** and enter the address of the image repository.

  Add the address of the custom image repository with no valid SSL certificate to the docker startup option to avoid unsuccessful image pulling from the personal image repository. The address is in the format of IP address:Port number (or domain name). Post-installation script and insecure-registries cannot be used together.

- **Taints**: This field is left blank by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node pool. Each taint contains the following parameters:
  - **Key**: A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key.
  - **Value**: A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (_), and periods (.).
  - **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**.

NOTICE

- If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes.
- Taints cannot be modified after configuration. Incorrect taints may cause a scale-up failure or prevent pods from being scheduled onto the added nodes.

- **K8S Labels**: Labels are key/value pairs that are attached to objects, such as pods. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. For more information, see **Labels and Selectors**.
- **Maximum Data Space per Container**: The value ranges from 10 GB to 80 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is displayed only for clusters of v1.13.10-r0 and later.

Step 6 Click **Next: Confirm**. Confirm the node pool configuration and click **Submit**.

It takes about 6 to 10 minutes to create a node pool. You can click **Back to Node Pool List** to perform other operations on the node pool or click **Go to Node Pool Events** to view the node pool details.

----End

Viewing Node Pools in a Cluster

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**.

Step 2 In the upper right corner of the node pool list, select a cluster. All node pools in the cluster will be displayed. You can view the node type, node specifications, autoscaler status, and OS of each node pool.

NOTE

- A default node pool **DefaultPool** is automatically created in each cluster. The default node pool cannot be edited, deleted, or migrated. All nodes created during and after cluster creation are displayed in the default node pool.
- To display a list of nodes in **DefaultPool**, click the **Nodes** subcard in the **DefaultPool** card.

Step 3 To filter node pools by autoscaler status, select the autoscaler status in the upper right corner of the node pool list.

Step 4 In the node pool list, click a node pool name. On the node pool details page, view the basic information, advanced ECS settings, advanced Kubernetes settings, and node list of the node pool.

----End

6.3 Managing a Node Pool

Configuration Management

To better manage Kubernetes parameters in a cluster, the configuration management function is provided. This function allows you to perform in-depth configuration on core components. For more information about configuration management, see **kubelet** and **dockerd**.

NOTE

- This function is available only for clusters of **v1.15 or later**.
- The default node pool DefaultPool does not support this type of configuration.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

Step 2 In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

Step 3 Click **Configuration** next to the node pool name.

Step 4 On the **Configuration** page on the right, change the values of the following Kubernetes parameters:

Table 6-2 Kubernetes parameters

Components	Parameters	Details	Default
docker	native-umask	`--exec-opt native.umask	normal
	docker-base-size	`--storage-opts dm.basesize	10G
kubelet	cpu-manager-policy	`--cpu-manager-policy	none
	kube-api-qps	Query per second (QPS) to use while talking with kube-apiserver.	100
	kube-api-burst	Burst to use while talking with kube-apiserver.	100
	max-pods	Maximum number of pods managed by kubelet.	110
	with-local-dns	Whether to use the local IP address as the ClusterDNS of the node.	false
	allowed-unsafe-sysctls	Insecure system configuration allowed.	[]
kube-proxy	conntrack-min	sysctl -w net.nf_conntrack_max	131072

Components	Parameters	Details	Default
	conntrack-tcp-timeout-close-wait	sysctl -w net.netfilter.nf_conntrack_tcp_timeout_clouse_wait	1h0m0s

Step 5 Click **OK**.

----End

Editing a Node Pool

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

Step 2 In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

Step 3 Click **Edit** next to the name of the node pool you will edit. In the **Edit Node Pool** dialog box, edit the following parameters:

Table 6-3 Node pool parameters

Parameters	Parameter description
Name of the node pool	Name of the node pool.
Number of orderers	Modify the number of nodes based on service requirements.
Autoscaler	By default, this parameter is disabled. After you enable autoscaler by clicking , nodes in the node pool are automatically created or deleted based on service requirements. ● **Maximum Nodes** and **Minimum Nodes**: You can set the maximum and minimum number of nodes to ensure that the number of nodes to be scaled is within a proper range. ● **Priority**: You can set the priority of auto scaling between node pools based on service requirements. If the value is set to **0**, no priority is set. Scaling is performed on node pools with the same priorities based on the minimum resource waste principle. ● **Scaling-In Cooling Interval**: Set this parameter in the unit of minute. This parameter indicates the interval between the previous scale-out action and the next scale-in action. If **Autoscaler** is enabled, install the **autoscaler add-on** to use the auto scaling feature.

Parameters	Parameter description
Taints	This parameter is left blank by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node pool. Each taint contains the following parameters: ● **Key**: A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. ● **Value**: A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (_), and periods (.). ● **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. **NOTICE** ● If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes. ● Taints cannot be modified after configuration. Incorrect taints may cause a scale-up failure or prevent pods from being scheduled onto the added nodes.
K8S Labels	K8S labels are key/value pairs that are attached to objects, such as pods. Labels are used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. For more information, see **Labels and Selectors**.

Step 4 Click **Save**.

In the node pool list, the node pool status is **Scaling**. After the status changes to **Complete**, the node pool parameters are edited successfully.

----End

Copying a node pool

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

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

Step 2 In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

Step 3 Choose **More > Copy** next to a node pool name to copy the node pool.

Step 4 The configuration of the selected node pool is replicated to the **Create Node Pool** page. You can edit the configuration as required and click **Next: Confirm**.

Step 5 On the **Confirm** page, confirm the node pool configuration and click **Create Now**. Then, a new node pool is created based on the edited configuration.

----End

Migrating a Node

Nodes in a node pool can be migrated. Currently, nodes in a node pool created by you can be migrated only to the default node pool (DefaultPool) in the same cluster, but not any other created node pools. Neither can nodes in DefaultPool be migrated to any other node pools.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

Step 2 In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

Step 3 Click **More** > **Migrate** next to the name of the node pool.

Step 4 In the dialog box displayed, select the destination node pool and the node to be migrated.

NOTE

After node migration, original resources tags, Kubernetes labels, and taints will be retained, and new Kubernetes labels and taints from the destination node pool will be added.

Step 5 Click **OK**.

----End

Deleting a Node Pool

Deleting a node pool will delete nodes in the pool. Pods on these nodes will be automatically migrated to available nodes in other node pools. If pods in the node pool have a specific node selector and none of the other nodes in the cluster satisfies the node selector, the pods will become unschedulable.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**.

Step 2 In the upper right corner of the displayed page, select a cluster to filter node pools by cluster.

Step 3 Click **Delete** next to the node pool name.

Step 4 Read the precautions in the **Delete Node Pool** dialog box.

Step 5 Enter **DELETE** in the text box and click **Yes** to confirm that you want to continue the deletion.

----End

7 Workload Management

7.1 Workload Overview

A workload is an abstract model of a group of pods in Kubernetes. Workloads classified in Kubernetes include Deployments, StatefulSets, DaemonSets, jobs, and cron jobs.

CCE provides Kubernetes-native container deployment and management and supports lifecycle management of container workloads, including creation, configuration, monitoring, auto scaling, upgrade, uninstall, service discovery, and load balancing.

Basic Concepts

- **Deployment**: Pods are completely independent of each other and functionally identical. They feature auto scaling and rolling upgrade. Typical examples include Nginx and WordPress. For details on how to create a deployment, see **Creating a Deployment**.
- **StatefulSet**: Pods are not completely independent of each other. They have stable persistent storage, and feature orderly deployment and deletion. Typical examples include MySQL-HA and etcd. For details on how to create a StatefulSet, see **Creating a StatefulSet**.
- **DaemonSet**: A DaemonSet ensures that all or some nodes run a pod. It is applicable to pods running on every node. Typical examples include Ceph, Fluentd, and Prometheus Node Exporter. For details about how to create a DaemonSet, see **Creating a DaemonSet**.
- **Job**: A job is a resource object that Kubernetes uses to control tasks in batches.

  A job is different from a long-term servo workload (such as Deployment and StatefulSet). The former is started and terminated at specific times, while the latter runs unceasingly unless being terminated. The pods managed by a job automatically exit after successfully completing the job based on user configurations. The success flag varies according to the spec.completions policy.

  – One-off jobs: A single pod runs once until successful termination.

- Jobs with a fixed success count: N pods run until successful termination.
- A queue job is considered successful based on the global success confirmed by the application.

For details about how to create a job, see **Creating a Job**.

- **Cron job**: A cron job runs periodically at the specified time. It is similar to Linux crontab. A cron job has the following characteristics:
  - Run a scheduled job once at the specified time.
  - Run a scheduled job periodically at the specified time.

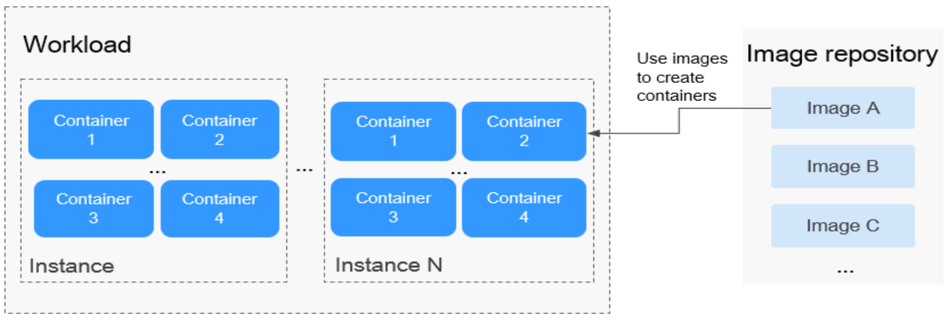
  The typical usage of a cron job is as follows:
  - Schedules jobs at the specified time.
  - Creates jobs to run periodically, for example, database backup and email sending.

  For details about how to create a cron job, see **Creating a Cron Job**.

Relationship Between Workloads and Containers

As shown in **Figure 7-1**, a workload controls one or more instances (pods). A pod consists of one or more containers. Each container is created from a container image. Pods of Deployments are exactly the same.

Figure 7-1 Relationship between workloads and containers



Workload Lifecycle

Table 7-1 Status description

Status	Description
Running	All pods are running.
Unready	All containers are in the pending state.
Updating	After the upgrade operation is triggered, the workload is being upgraded.
Stopped	After the stop operation is triggered, the workload is stopped and the number of pods changes to 0.

Status	Description
Available	For a multi-pod Deployment, some pods are abnormal but at least one pod is available.
Deleting	After the delete operation is triggered, the workload is being deleted.

7.2 Creating a Deployment

Deployments are workloads (for example, Nginx) that do not store any data or status.

Preparations

- Before creating a containerized workload, you must have an available cluster. For details on how to create a cluster, see **Creating a Hybrid Cluster**.

  NOTE

  When creating multiple containerized workloads, ensure that each workload has a unique port. Otherwise, workload deployment will fail.

- If the workload will be reachable from public networks, the workload must have a load balancer or at least one node in the cluster has an elastic IP address (EIP).

Using the CCE Console

Step 1 CCE provides multiple methods for creating a workload. You can use any of the following methods:

- Use a **third-party image** to create a workload, without the need to upload an image.
- Perform required operations on the **My Images** page on the CCE console to create a workload, you need to upload the image to the Software Repository for Container (SWR). .
- Use a **shared image** to create a workload. Specifically, other tenants share an image with you by using the **SWR service**.
- Use a YAML file to create a workload. You can click **Create YAML** on the right of the **Create Deployment** page. For details about YAML, see **Creating a Deployment Using kubectl**. After the YAML file is written, click **Create** to create a workload.

  NOTE

  The YAML file is synchronized with the console. You can also interact with the YAML to create a workload. For example, after you enter a workload name on the console, the YAML file is automatically associated with the name. For example, after an image is added on the console, the image is automatically associated with YAML.

Step 2 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**. On the page displayed, click **Create Deployment**. Set basic workload parameters as described in **Table 7-2**. The parameters marked with an asterisk (*) are mandatory.

Table 7-2 Basic workload parameters

Parameter	Description
* Workload Name	Name of the containerized workload to be created. The name must be unique.
* Cluster Name	Cluster in which the workload resides.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the default namespace is used.
* Instances	Number of pods in the workload. A workload can have one or more pods. You can set the number of pods. The default value is **2** and can be set to **1**. Each workload pod consists of the same containers. You can configure multiple pods for a workload to ensure high reliability. For such a workload, if one pod is faulty, the workload can still run properly. If only one pod is used, a node or pod exception may cause service exceptions.
Time Zone Synchronization	After **ON** is selected, the local storage guaranteed container time zone will be the same as the node time zone. **NOTICE** After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volume** area. Do not modify or delete the disks.
Description	Description of the workload.

Step 3 Click **Next: Add Container**.

1. Click **Add Container** and select the image to be deployed.
   - **My Images**: Create a workload using an image in the image repository you created.
   - **Third-party Images**: Create a workload using an image from any third-party image repository. When you create a workload using a third-party image, ensure that the node where the workload is running can access public networks. For details on how to create a workload using a third-party image, see **Using a Third-Party Image**.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address, and then click **OK**.
     - If your image repository requires authentications (account and password), create a key and then use a third-party image. For details, see **Using a Third-Party Image**.
   - **Shared Images**: Create a workload using an image shared by another tenant through the SWR service.

2. Configure basic image information.

Table 7-3 Parameter description of images

Parameter	Description
Image Name	Name of the image. You can click **Change Image** to update it.
*Image Version	Version of the image to be deployed.
*Container Name	Name of the container. You can modify it.
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is **On**, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
Container Resources	CPU quotas: – **CPU request**: the minimum number of CPU cores required by a container. The default value is 0.25 core. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. Memory quotas: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. GPU quotas: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this parameter to 0, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses this GPU.

3. **Lifecycle**: Manage commands for starting and running containers.

   - **Startup Command**: executed when the workload is started. For details, see **Setting Container Startup Commands**.
   - **Post-Start Processing**: executed after the workload is successfully run. For more information, see **Setting Container Lifecycle Parameters**.
   - **Pre-Stop Processing**: executed to delete logs or temporary files before the workload ends. For more information, see **Setting Container Lifecycle Parameters**.
4. **Health Check**: Check whether containers and services are running properly. CCE provides two types of probes: liveness probes and readiness probes. For more information, see **Setting Health Check for a Container**.
   - Liveness probe: restarts the unhealthy container.
   - Readiness probe: change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
5. **Environment Variables**: Environment variables can be added to a container and are generally used to set parameters.

   On the **Environment Variables** tab page, click **Add Environment Variables**. Add an environment variable in one of the following ways:
   - **Added manually**: Set **Variable Name** and **Variable/Variable Reference**.
   - **Added from Secret**: Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see **Creating a secret**.
   - **Added from ConfigMap**: Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see **Creating a ConfigMap**.
6. **Data Storage**: Mount data storage to containers for persistent storage and high disk I/O. For details, see **Storage Management**.
7. **Security Context**: Configure container permissions to protect CCE and other containers from being affected.

   Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies**: Configure the log collection policies and log directory to collect container logs for unified management and analysis. For details, see **Collecting Standard Output Logs of Containers** and **Collecting Container Logs from Specified Paths**.
9. (Optional) A pod contains one or more closely related containers. If your workload is a multi-container workload, click **Add Container** to add more containers.

Step 4 Click **Next: Set Application Access**. Then, click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see **Network Overview**.

Step 5 Click **Next: Configure Advanced Settings** and configure advanced settings.

- **Upgrade Policy**
  - **Rolling upgrade**: Gradually replaces old pods with new pods. During the upgrade, service traffic is evenly distributed to both old and new pods to ensure service continuity.

    Maximum Number of Unavailable Pods: 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 – Maximum number of unavailable pods
  - **In-place upgrade**: An old pod is deleted and then a new one is created. Services are interrupted during the upgrade.
- **Graceful Deletion**: On the **Graceful Time Window (s)** page, set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s. The graceful scale-in policy provides a time window for workload deletion and is reserved for executing commands in the PreStop phase in the lifecycle. If workload processes are not terminated after the time window elapses, the workload will be forcibly deleted.
  - **Graceful Time Window (s)**: Set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
  - **Scale Order**: Choose **Prioritize new pods** or **Prioritize old pods** based on service requirements. **Prioritize new pods** indicates that new pods will be first deleted when a scale-in is triggered.
- **Migration Policy**: Set the time window. When the node where the Deployment pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes. By default, the migration time window is 300s.
- **Scheduling Policies**: You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see **Scheduling Policy Overview**.
- **Advanced Pod Settings** > **Pod Label**: Built-in app labels are specified during workload creation and cannot be modified. You can click **Add Label** to add labels.
- **Client DNS Configuration**: A CCE cluster has a built-in DNS add-on (coredns) to provide domain name resolution for workloads in the cluster. For details, see **Using Kubernetes In-Cluster DNS**.
  - **DNS Policy**
    - **ClusterFirst**: The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
    - **None**: The pod uses the following configurations of the client.
    - **Default**: The pod inherits the DNS configuration from the node on which the pod runs.
  - **Nameserver**: You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.
  - **DNS Search Domain**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the

domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.

– **Timeout (s)**: amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.

– **ndots**: threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

Step 6 After the configuration is complete, click **Create**. Click **Back to Deployment List** to view the deployment status.

If the workload is in the **Running** state, it has been successfully created.

Workload status is not updated in real time. Click ↻ in the upper right corner or press **F5** to refresh the page.

Step 7 To access the workload in a browser, go to the workload list on the **Workload** page. Copy the corresponding **External Access Address** and paste it into the address box in the browser.

NOTE

- External access addresses are available only if the Deployment access type is set to **NodePort** and an EIP is assigned to any node in the cluster, or if the Deployment access type is set to **LoadBalancer (ELB)**.
- When the deployment list contains more than 500 records, the Kubernetes pagination mechanism will be used. Kubernetes pagination mechanism: You can only go to the first page or the next page, but cannot go to the previous page. In addition, if resources are divided into discrete pages, the total number of resources displayed is the maximum number of resources that can be queried at a time, not the actual total number of resources.

----End

Creating a Deployment Using kubectl

The following procedure uses an Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisites

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name, and you can change it as required.

vi nginx-deployment.yaml

The following is an example yaml file. For more information about Deployments, see **Kubernetes documentation**.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

For details about deployment.yaml parameters, see **Table 7-4**.

Table 7-4 deployment.yaml parameters

Parameter	Description	Mandatory/ Optional
apiVersion	API version.	Mandatory
kind	Type of a created object.	Mandatory
metadata	Metadata of a resource object.	Mandatory
name	Name of the Deployment.	Mandatory
Spec	Detailed description of the Deployment.	Mandatory
replicas	Number of pods.	Mandatory
selector	Container pods that can be managed by the Deployment.	Mandatory
strategy	Upgrade mode. Possible values: ● RollingUpdate ● ReplaceUpdate By default, rolling update is used.	Optional
template	Detailed description of a created container pod.	Mandatory
metadata	Metadata.	Mandatory
labels	**metadata.labels**: Container labels.	Optional

Parameter	Description	Mandatory/ Optional
spec: containers	• **image** (mandatory): Name of a container image. • **imagePullPolicy** (optional): Policy for obtaining an image. The options include **Always** (attempting to download images each time), **Never** (only using local images), and **IfNotPresent** (using local images if they are available; downloading images if local images are unavailable). The default value is **Always**. • **name** (mandatory): Container name.	Mandatory
imagePullSecrets	Name of the secret used during image pulling. If a private image is used, this parameter is mandatory. • To pull an image from the Software Repository for Container (SWR), set this parameter to **default-secret**. • To pull an image from a third-party image repository, set this parameter to the name of the created secret.	Optional

Step 3 Create a Deployment.

kubectl create -f nginx-deployment.yaml

If the following information is displayed, the workload is being created.

```
deployment "nginx" created
```

Step 4 Query the Deployment status.

kubectl get po

If the following information is displayed, the Deployment is running.

```
NAME                   READY     STATUS    RESTARTS   AGE
icagent-m9dkt            0/0       Running   0          3d
nginx-1212400781-qv313   1/1       Running   0          3d
```

Step 5 If the Deployment will be accessed through a ClusterIP or NodePort Service, add the corresponding Service. For details, see **Network Management**.

----End

7.3 Creating a StatefulSet

StatefulSets are a type of workloads whose data or status is stored while they are running. For example, MySQL is a StatefulSet because it needs to store new data.

A container can be migrated between different hosts, but data is not stored on the hosts. To store StatefulSet data persistently, attach HA storage volumes provided by CCE to the container.

Preparations

Before creating a containerized workload, you must have an available cluster. For details on how to create a cluster, see **Creating a Hybrid Cluster**.

NOTE

When creating a workload with multiple containers, ensure that each container has a unique port. Otherwise, workload deployment will fail.

Using the CCE Console

Step 1 CCE provides multiple methods for creating a workload. You can use any of the following methods:

- Use a **third-party image** to create a workload, without the need to upload an image.
- Perform required operations on the **My Images** page on the CCE console to create a workload, you need to upload the image to the Software Repository for Container (SWR). .
- Use a **shared image** to create a workload. Specifically, other tenants share an image with you by using the **SWR service**.
- Use a YAML file to create a workload. You can click **Create YAML** on the right of the **Create StatefulSet** page. For details about YAML, see **Creating a StatefulSet Using kubectl**. After the YAML file is written, click **Create** to create a workload.

  NOTE

  Settings in the YAML file are synchronized to the console. You can use both the console and YAML to create a workload. For example, after you enter a workload name on the console, this name will automatically appear in the YAML file. Similarly, if you add an image on the console, the image will be automatically added to the YAML file.

Step 2 Log in to the CCE console. In the navigation pane, choose **Workloads** > **StatefulSets**. On the displayed page, click **Create StatefulSet**. Set basic workload parameters as described in **Table 7-5**. The parameters marked with an asterisk (*) are mandatory.

Table 7-5 Basic workload parameters

Parameter	Description
* Workload Name	Name of a workload, which must be unique.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the **default** namespace is used.
* Instances	Number of pods in the workload. A workload can have one or more pods. You can set the number of pods. The default value is **2** and can be set to **1**. Each workload pod consists of the same containers. You can configure multiple pods for a workload to ensure high reliability. For such a workload, if one pod is faulty, the workload can still run properly. If only one pod is used, a node or pod exception may cause service exceptions.
Time Zone Synchronization	If this parameter is enabled, the container and the node use the same time zone. **NOTICE** After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volume** area. Do not modify or delete the disks.
Description	Description of the workload.

Step 3 Click **Next: Add Container**.

1. Click **Add Container** and select the image to be deployed.
   - **My Images**: Create a workload using an image in the image repository you created.
   - **Third-Party Images**: Create a workload using an image pulled from a third-party image repository. Ensure that the node where the workload is running is accessible from public networks. For details on how to create a workload using a third-party image, see **Using a Third-Party Image**.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address, and then click **OK**.
     - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see **Using a Third-Party Image**.
   - **Shared Images**: Create a workload using an image shared by another tenant through the SWR service.
2. Configure basic image information.

Table 7-6 Parameter description of images

Parameter	Description
Image Name	Name of the image. You can click **Change Image** to select another image.
*Image Version	Tag of the image to be deployed. If you chose to use an official image in the Docker Hub repository, you must specify this parameter.
*Container Name	Container name, which is modifiable.
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
Container Specifications	**CPU Quota**: – **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. **Memory Quota**: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. **GPU**: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this parameter to **0**, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.

3. **Lifecycle**: Manage commands for starting and running containers.

   - **Start Command**: executed when the workload is started. For details, see **Setting Container Startup Commands**.
   - **Post-Start**: executed after a container runs successfully. For details, see **Setting Container Lifecycle Parameters**.
   - **Pre-Stop**: executed to delete logs or temporary files before a container ends. For details, see **Setting Container Lifecycle Parameters**.

4. **Health Check**: Check whether containers and services are running properly. CCE provides two types of probes: liveness probes and readiness probes. For more information, see **Setting Health Check for a Container**.
   - **Liveness Probe**: used to restart the unhealthy container.
   - **Readiness Probe**: used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
5. **Environment Variables**: Environment variables can be added to a container and are generally used to set parameters.

   In the **Environment Variables** area, click **Add Environment Variable**. Currently, environment variables can be added using any of the following methods:
   - **Added manually**: Set **Variable Name** and **Variable Value**.
   - **Added from Secret**: Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see **Creating a secret**.
   - **Added from ConfigMap**: Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see **Creating a ConfigMap**.
6. **Data Storage**: Mount data storage to containers for persistent storage and high disk I/O. For details, see **Storage Management**.
7. **Security Context**: Configure container permissions to protect CCE and other containers from being affected.

   Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies**: Configure the log collection policies and log directory to collect container logs for unified management and analysis. For details, see **Collecting Standard Output Logs of Containers** and **Collecting Container Logs from Specified Paths**.

Step 4 Click **Next: Set Application Access** and set headless service parameters, as shown in **Table 7-7**.

Table 7-7 Headless service parameters

Parameter	Description
Service Name	Name of the service corresponding to the workload for mutual access between instances. This service is used for internal discovery of instances, and does not require an independent IP address or load balancing.

Parameter	Description
Port Name	Name of the container port. You are advised to enter a name that indicates the function of the port.
Container Port	Listening port inside the container.

Step 5 Click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see **Network Overview**.

Step 6 Click **Next: Configure Advanced Settings** and configure advanced settings.

- **Upgrade Policy**: Only **Rolling Upgrade** is supported.
- **Pod Management Policy**: There are two types of policies: **ordered** and **parallel**.
  - **Ordered**: The StatefulSet will deploy, delete, or scale pods in order and one by one (The StatefulSet waits until each pod is ready before continuing).
  - **Parallel**: The StatefulSet will create pods in parallel to match the desired scale without waiting, and will delete all pods at once.
- **Graceful Deletion**:
  - **Graceful Time Window (s)**: Set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
  - **Scale Order**: Choose **Prioritize new pods** or **Prioritize old pods** based on service requirements. **Prioritize new pods** indicates that new pods will be first deleted when a scale-in is triggered.
- **Scheduling Policies**: You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see **Scheduling Policy Overview**.
- **Advanced Pod Settings** > **Pod Label**: Built-in app labels are specified during workload creation and cannot be modified. You can click **Add Label** to add labels.
- **Client DNS Configuration**: A CCE cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster. For details, see **Using Kubernetes In-Cluster DNS**.
  - **DNS Policy**
    - **ClusterFirst**: The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
    - **None**: The pod uses the following configurations of the client.
    - **Default**: The pod inherits the DNS configuration from the node on which the pod runs.

- **Nameserver**: You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.
- **DNS Search Domain**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.
- **Timeout (s)**: amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.
- **ndots**: threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

Step 7 Click **Create** and then **Back to StatefulSet List**.

NOTE

- If the StatefulSet is in the **Running** state, it has been successfully created.
- Workload status is not updated in real time. Click in the upper right corner or press **F5** to refresh the page.
- When the workload list contains more than 500 records, the Kubernetes pagination mechanism will be used. With Kubernetes pagination, you can only go to the first page or the next page, but cannot go to the previous page. In addition, if resources are divided into discrete pages, the total number of resources displayed is the maximum number of resources that can be queried at a time, not the actual total number of resources.

----End

Creating a StatefulSet Using kubectl

The following procedure uses an etcd workload as an example to describe how to create a workload using kubectl.

Prerequisites

You have configured the kubectl and connected an to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create and edit the **etcd-statefulset.yaml** file.

etcd-statefulset.yaml is an example file name, and you can change it as required.

vi etcd-statefulset.yaml

The following provides an example of the file contents. For more information on StatefulSet, see the **Kubernetes documentation**.

```
apiVersion: apps/v1beta1
kind: StatefulSet
```

```
metadata:
  name: etcd
spec:
  replicas: 2
  selector:
    matchLabels:
      app: etcd
  serviceName: etcd-svc
  template:
    metadata:
      labels:
        app: etcd
    spec:
      containers:
      - env:
        - name: PAAS_APP_NAME
          value: tesyhhj
        - name: PAAS_NAMESPACE
          value: default
        - name: PAAS_PROJECT_ID
          value: 9632fae707ce4416a0ab1e3e121fe555
        image: etcd
        imagePullPolicy: IfNotPresent
        name: container-0
  updateStrategy:
    type: RollingUpdate
```

vi etcd-headless.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: etcd
  name: etcd-svc
spec:
  clusterIP: None
  ports:
  - name: etcd-svc
    port: 3120
    protocol: TCP
    targetPort: 3120
  selector:
    app: etcd
  sessionAffinity: None
  type: ClusterIP
```

Step 3 Create a workload and the corresponding headless Service.

kubectl create -f etcd-statefulset.yaml

If the following information is displayed, the StatefulSet has been successfully created.

```
statefulset "etcd" created
```

kubectl create -f etcd-headless.yaml

If the following information is displayed, the headless Service has been successfully created.

```
service "etcd-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 **Network Management**.

----End

7.4 Creating a DaemonSet

CCE provides deployment and management capabilities for multiple types of containers and supports features of container workloads, including creation, configuration, monitoring, scaling, upgrade, uninstall, service discovery, and load balancing.

DaemonSet ensures that only one pod runs on all or some nodes. When a node is added to a cluster, a new pod is also added for the node. When a node is removed from a cluster, the pod is also reclaimed. If a DaemonSet is deleted, all pods created by it will be deleted.

Typical types of DaemonSets:

- Run the cluster storage daemon, such as glusterd or Ceph, on each node.
- Run the log collection daemon, such as Fluentd or Logstash, on each node.
- Run the monitoring daemon, such as Prometheus Node Exporter, collectd, Datadog agent, New Relic agent, or Ganglia (gmond), on each node.

In a simple case, one DaemonSet, covering all nodes, will be used for each type of daemon. A more complex setup may use multiple DaemonSets for a single type of daemon, but with different flags and/or different memory and CPU requests for different hardware types.

Preparations

You must have one cluster available before creating a DaemonSet. For details on how to create a cluster, see **Creating a Hybrid Cluster**.

Procedure

Step 1 CCE provides multiple methods for creating a workload. You can use any of the following methods:

- Use a **third-party image** to create a workload, without the need to upload an image.
- Perform required operations on the **My Images** page on the CCE console to create a workload, you need to upload the image to the Software Repository for Container (SWR). .
- Use a **shared image** to create a workload. Specifically, other tenants share an image with you by using the **SWR service**.
- Use a YAML file to create a workload. You can click **Create YAML** on the right of the **Create Deployment** page. After the YAML file is written, click **Create** to create a workload.

  NOTE

  Settings in the YAML file are synchronized with settings on the console. You can also interact with the YAML to create a workload. For example, if you enter a workload name on the console, the name will automatically appear in the YAML file. Similarly, if you add an image on the console, the image will be automatically added to the YAML file.

Step 2 Log in to the CCE console. In the navigation pane, choose **Workloads > DaemonSets**. On the displayed page, click **Create DaemonSet**.

Step 3 Set basic workload parameters as described in **Table 7-8**. The parameters marked with an asterisk (*) are mandatory.

Table 7-8 Basic workload parameters

Parameter	Parameter description
* Workload Name	Name of a workload, which must be unique.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	In a single cluster, data in different namespaces is isolated from each other. This enables applications to share the services of the same cluster without interfering each other. If no namespace is set, the **default** namespace is used.
Time Zone Synchronization	If this parameter is enabled, the container and the node use the same time zone. **NOTICE** After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volume** area. Do not modify or delete the disks.
Description	Description of the workload.

Step 4 Click **Next: Add Container**.

1. Click **Add Container** and select the image to be deployed.
   – **My Images**: Create a workload using an image in the image repository you created.
   – **Third-party Images**: Create a workload using an image from any third-party image repository. Ensure that the node where the workload is running is accessible from public networks. For details on how to create a workload using a third-party image, see **Using a Third-Party Image**.
     ▪ If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address, and then click **OK**.
     ▪ If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see **Using a Third-Party Image**.
   – **Shared Images**: Create a workload using an image shared by another tenant through the SWR service.
2. Configure basic image information.

Table 7-9 Parameter description of images

Parameter	Description
Image Name	Name of the image. You can click **Change Image** to select another image.
*Image Version	Tag of the image to be deployed. If you chose to use an official image in the Docker Hub repository, you must specify this parameter.
*Container Name	Container name, which is modifiable.
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.
Container Specifications	**CPU Quota**: – **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. **Memory Quota**: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. **GPU**: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this parameter to **0**, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.

3. **Lifecycle**: Manage commands for starting and running containers.

   - **Start Command**: executed when the workload is started. For details, see **Setting Container Startup Commands**.
   - **Post-Start**: executed after a container runs successfully. For details, see **Setting Container Lifecycle Parameters**.
   - **Pre-Stop**: executed to delete logs or temporary files before a container ends. For details, see **Setting Container Lifecycle Parameters**.
4. **Health Check**: Check whether containers and services are running properly. CCE provides two types of probes: liveness probes and readiness probes. For more information, see **Setting Health Check for a Container**.
   - **Liveness Probe**: used to restart the unhealthy container.
   - **Readiness Probe**: used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
5. **Environment Variables**: Environment variables can be added to a container and are generally used to set parameters.

   In the **Environment Variables** area, click **Add Environment Variable**. Currently, environment variables can be added using any of the following methods:
   - **Added manually**: Set **Variable Name** and **Variable Value**.
   - **Added from Secret**: Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see **Creating a secret**.
   - **Added from ConfigMap**: Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see **Creating a ConfigMap**.
6. **Data Storage**: Mount data storage to containers for persistent storage and high disk I/O. For details, see **Storage Management**.
7. **Security Context**: Configure container permissions to protect CCE and other containers from being affected.

   Enter the user ID to set container permissions and prevent systems and other containers from being affected.
8. **Log Policies**: Configure the log collection policies and log directory to collect container logs for unified management and analysis. For details, see **Collecting Standard Output Logs of Containers** and **Collecting Container Logs from Specified Paths**.
9. (Optional) A pod contains one or more closely related containers. If your workload is a multi-container workload, click **Add Container** to add more containers.

Step 5 Click **Next: Set Application Access**. Then, click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see **Network Overview**.

Step 6 Click **Next: Configure Advanced Settings** to configure advanced policies.

- **Upgrade Policy**:

  Upgrade Mode: Only **Rolling Upgrade** is supported. During a rolling upgrade, old pods are gradually replaced with new ones, and service traffic is evenly distributed to both pods to ensure service continuity.

  Maximum Number of Unavailable Pods: 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 – Maximum number of unavailable pods

- **Graceful Deletion**:
  - **Graceful Time Window (s)**: Set a time window (0–9999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
  - **Scale Order**: Choose **Prioritize new pods** or **Prioritize old pods** based on service requirements. **Prioritize new pods** indicates that new pods will be first deleted when a scale-in is triggered.
- **Scheduling Policies**: You can combine static global scheduling policies or dynamic runtime scheduling policies as required. For details, see **Scheduling Policy Overview**.
- **Advanced Pod Settings** > **Pod Label**: Built-in app labels are specified during workload creation and cannot be modified. You can click **Add Label** to add labels.
- **Client DNS Configuration**: A CCE cluster has a built-in DNS add-on (CoreDNS) to provide domain name resolution for workloads in the cluster. For details, see **Using Kubernetes In-Cluster DNS**.
  - **DNS Policy**
    - **ClusterFirst**: The default DNS configuration overrides the **Nameserver** and **DNS Search Domain** configurations of the client.
    - **None**: The pod uses the following configurations of the client.
    - **Default**: The pod inherits the DNS configuration from the node on which the pod runs.
  - **Nameserver**: You can configure a domain name server for a user-defined domain name. The value is one or a group of DNS IP addresses, for example, 1.2.3.4.
  - **DNS Search Domain**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried.
  - **Timeout (s)**: amount of time the resolver will wait for a response from a remote name server before retrying the query on a different name server. Set it based on the site requirements.
  - **ndots**: threshold for the number of dots that must appear in a domain name before an initial absolute query will be made. If a domain name has **ndots** or more than **ndots** dots, the name is a fully qualified domain name (FQDN) and will be tried first as an absolute name. If a domain name has less than **ndots** dots, the operating system will look up the name in a list of search domain names.

Step 7 After the preceding configurations are complete, click **Create**. On the page displayed, click **Return to Workload List** to view the workload status.

If the workload is in the **Running** state, it has been successfully created.

Workload status is not updated in real time. Click ↻ in the upper right corner or press **F5** to refresh the page.

----End

7.5 Creating a Job

A one-off job is executed only once immediately after being deployed. Before creating a workload, you can execute a one-off job to upload an image to the image repository.

Preparations

Nodes have been added. For more information, see **Creating a Node**. If clusters and nodes are available, skip this operation.

Procedure

Step 1 (Optional) If you use a private container image to create your one-off job, upload the container image to the image repository.

Step 2 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Jobs**. Click **Create Job**.

Step 3 Configure the basic job information listed in **Table 7-10**. The parameters marked with an asterisk (*) are mandatory.

Table 7-10 Basic job information

Parameter	Description
* Job Name	Name of a new job. The name must be unique.
* Container Cluster	Cluster to which a new job belongs.
* Namespace	Namespace to which the new job belongs. By default, this parameter is set to **default**.
*Instance Quantity	Number of pods in this job. A job can have one or more pods. You can specify the number of pods. The default value is **1**. Each job pod consists of the same containers. Configuring multiple job pods can ensure high availability. The job can continue to run even if one of the pods is faulty.
Description	Description of a job.

Step 4 Click **Next: Add Container** to add a container.

1. Click **Select Container Image** to select the image to be deployed.
   - **My Images**: displays all image repositories you create.
   - **Third-party Images**: CCE allows you to create a job from any third-party image repository. When you create a workload using a third-party image, ensure that the node where the workload is running can access public networks. For details about how to use a third-party image, see **Using a Third-Party Image**.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address, and then click **OK**.
     - If your image repository must be authenticated (account and password), you need to create a key and then use a third-party image. For details, see **Using a Third-Party Image**.
   - **Shared Images**: The images shared by other tenants using the SWR service are displayed here. You can create workloads based on the shared images.
2. Set image parameters.

Table 7-11 Image parameters

Parameter	Description
Image	Name of the image. You can click **Change Image** to update it.
*Image Version	Version of the image to be deployed.
*Container Name	Name of the container. You can modify it.

Parameter	Description
Container Specifications	**CPU Quota**: – **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. **Memory Quota**: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. **GPU**: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this parameter to **0**, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.

3. (Optional) Configure advanced settings.

Table 7-12 Advanced settings

Parameter	Description
Lifecycle	Lifecycle scripts define the actions taken for container-related jobs when a lifecycle event occurs. – **Start**: If you enter a container startup command, the command is immediately executed after the container starts. For details, see **Advanced Container Settings**. – **Post-Start**: The command is triggered after a job starts. For details, see **Setting Container Lifecycle Parameters**. – **Pre-Stop**: The command is triggered before a job is stopped. For details, see **Setting Container Lifecycle Parameters**.

Parameter	Description
Environment Variables	Environment variables can be added to a container. In general, environment variables are used to set parameters. On the **Environment Variables** tab page, click **Add Environment Variables**. Currently, environment variables can be added using any of the following methods: – Manually added: Set **Variable Name** and **Variable/ Variable Reference**. – Importing a secret: Set **Variable Name** and select the desired secret name and data. The prerequisite of this method is that a secret has been created. For details, see **Creating a secret**. – Importing a ConfigMap: Set **Variable Name** and select the desired ConfigMap name and data. The prerequisite of this method is that a ConfigMap has been created. For details, see **Creating a ConfigMap**.
Data Storage	The local disk or cloud storage can be mounted to a container to implement persistent data file storage. For details, see **Storage Management**.
Log Policy	Set a log policy and log path for collecting workload logs and preventing logs from being over-sized. For details, see **Collecting Standard Output Logs of Containers**.

4. (Optional) One job instance contains one or more related containers. If your job contains multiple containers, add containers.

Step 5 After the configuration is complete, click **Submit**.

If the status of the job is **Execution completed**, the job has been created successfully.

----End

Creating a Job Using kubectl

A job has the following configuration parameters:

- **spec.template**: has the same schema as a pod.
- **RestartPolicy**: can only be set to **Never** or **OnFailure**.
- For a single-pod job, the job ends after the pod runs successfully by default.
- **.spec.completions**: indicates the number of pods that need to run successfully to end a job. The default value is **1**.
- **.spec.parallelism**: indicates the number of pods that run concurrently. The default value is **1**.
- **spec.backoffLimit**: indicates the maximum number of retries performed if a pod fails. When the limit is reached, the pod will not try again.
- **.spec.activeDeadlineSeconds**: indicates the running time of pods. Once the time is reached, all pods of the job are terminated. The priority

of .spec.activeDeadlineSeconds is higher than that of .spec.backoffLimit. That is, if a job reaches the .spec.activeDeadlineSeconds, the spec.backoffLimit is ignored.

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

Table 7-13 Job types

Job Type	Description	Example
One-off jobs	A single pod runs once until successful termination.	Database migration
Jobs with a fixed completion count	One pod runs until reaching the specified **completions** count.	Work queue processing pod
Parallel jobs with a fixed completion count	Multiple pods run until reaching the specified **completions** count.	Multiple pods processing from a centralized work queue
Parallel jobs	One or more pods run until successful termination.	Multiple pods processing from a centralized work queue

The following is an example job, which calculates π till the 2000th digit and prints the output.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-timeout
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 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
```

Related Operations

After a one-off job is created, you can perform operations listed in **Table 7-14**.

Table 7-14 Other operations

Operation	Description
Viewing a YAML	Click **View YAML** next to the job name to view the YAML file corresponding to the current job.
Deleting a one-off job	1. Select the job to be deleted and click **Delete** in the **Operation** column. 2. Click **OK**. Deleted jobs cannot be restored. Therefore, exercise caution when deleting a job.

7.6 Creating a Cron Job

A cron job is a short-lived job that runs at a specified time. You can perform time synchronization for all active nodes at a fixed time point.

Preparations

Nodes have been added. For more information, see **Creating a Node**. If clusters and nodes are available, skip this operation.

Procedure

Step 1 (Optional) If you use a private container image to create your containerized cron job, upload the container image to the image repository.

Step 2 Log in to the CCE console. In the navigation pane, choose **Workloads > Cron Jobs**. Then, click **Create Cron Job**.

Step 3 Configure the basic cron job information listed in **Table 7-15**. The parameters marked with an asterisk (*) are mandatory.

Table 7-15 Basic job information

Parameter	Description
* Job Name	Name of a new job. The name must be unique.
* Container Cluster	Cluster to which a new job belongs.
* Namespace	Namespace to which a cron job belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of a cron job.

Step 4 Click **Next: Configure Timing Rule**.

Step 5 Set the scheduling rule.

Table 7-16 Scheduling rule parameters

Parameter	Description
* Concurrent Policy	The following three modes are supported: ● Forbid: A new job cannot be created before the previous job is complete. ● Allow: New cron jobs can be created continuously. ● Replace: A new job replaces the previous job when it is time to create the new job but the previous job is not complete.
* Timing Rule	Specifies the time when a new cron job is executed.
Job Record	You can set the number of tasks that are successfully executed or fail to be executed. The value **0** indicates that the number of reserved tasks is not limited.

Step 6 Click **Next: Add Container** to add a container.

1. Click **Select Container Image** to select the image to be deployed.
   - **My Images**: displays all image repositories you created.
   - **Third-party Images**: CCE allows you to create a job from any third-party image repository. When you create a job using a third-party image, ensure that the node where the job is running can access public networks. For details on how to create a workload using a third-party image, see **Using a Third-Party Image**.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, enter an image address, and then click **OK**.
     - If your image repository must be authenticated (account and password), you need to create a secret and then use a third-party image. For details, see **Using a Third-Party Image**.
   - **Shared Images**: The images shared by other tenants using the SWR service are displayed here. You can create workloads based on the shared images.
2. Set image parameters.

Table 7-17 Image parameters

Parameter	Description
Image	Name of the image. You can click **Change Image** to select another image.
*Image Version	Version of the image to be deployed.
*Container Name	Container name, which is modifiable.

Parameter	Description
Container Specifications	**CPU Quota**: – **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. **Memory Quota**: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. **GPU**: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this parameter to **0**, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.

3. (Optional) Perform advanced settings.

Table 7-18 Advanced settings

Parameter	Description
Lifecycle	Actions defined in the lifecycle script definition are taken for the lifecycle events of container tasks. – **Start**: If you enter a container startup command, the command is immediately executed after the container starts. For details, see **Advanced Container Settings**. – **Post-Start**: The command is triggered after a job starts. For details, see **Setting Container Lifecycle Parameters**. – **Pre-Stop**: The command is triggered before a job is stopped. For details, see **Setting Container Lifecycle Parameters**.

Parameter	Description
Environment Variables	Environment variables can be added to a container. In general, environment variables are used to set parameters. On the **Environment Variables** tab page, click **Add Environment Variables**. Currently, environment variables can be added using any of the following methods: – Manually added: Set **Variable Name** and **Variable/ Variable Reference**. – Importing a secret: Set **Variable Name** and select the desired secret name and data. The prerequisite of this method is that a secret has been created. For details, see **Creating a secret**. – Importing a ConfigMap: Set **Variable Name** and select the desired ConfigMap name and data. The prerequisite of this method is that a ConfigMap has been created. For details, see **Creating a ConfigMap**.

4. (Optional) One job instance contains one or more related containers. If your job contains multiple containers, click **Add Container** to add containers.

Step 7 Click **Create**.

If the status is **Started**, the cron job has been created successfully.

----End

Creating a Cron Job Using kubectl

A Cron job has the following configuration parameters:

- **.spec.schedule**: takes a **Cron** format string, for example, **0 * * * *** or **@hourly**, as schedule time of jobs to be created and executed.
- **.spec.jobTemplate**: specifies jobs to be run, and has exactly the same schema as when you are **Creating a Job Using kubectl**.
- **.spec.startingDeadlineSeconds**: specifies the deadline for starting a job.
- **.spec.concurrencyPolicy**: specifies how to treat concurrent executions of a job created by the Cron job. The following options are supported:
  – **Allow** (default value): allows concurrently running jobs.
  – **Forbid**: forbids concurrent runs, skipping next run if previous has not finished yet.
  – **Replace**: cancels the currently running job and replaces it with a new one.

The following is an example Cron job, which is saved in the **cronjob.yaml** file.

```
apiVersion: batch/v2alpha1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
```

```
jobTemplate:
  spec:
    template:
      spec:
        containers:
        - name: hello
          image: busybox
          args:
          - /bin/sh
          - -c
          - date; echo Hello from the Kubernetes cluster
        restartPolicy: OnFailure
```

Step 1 Create a cron job.

kubectl create -f cronjob.yaml

Information similar to the following is displayed:

```
cronjob "hello" created
```

Step 2 Query the running status of the cron job:

kubectl get cronjob

```
NAME      SCHEDULE      SUSPEND   ACTIVE    LAST-SCHEDULE
hello     */1 * * * *   False     0         <none>
```

kubectl get jobs

```
NAME               DESIRED   SUCCESSFUL   AGE
hello-1202039034   1         1            49s
$ pods=$(kubectl get pods --selector=job-name=hello-1202039034 --
output=jsonpath={.items..metadata.name} -a)
```

kubectl logs $pods

```
Mon Aug 29 21:34:09 UTC 2016
Hello from the Kubernetes cluster
```

kubectl delete cronjob hello

```
cronjob "hello" deleted
```

NOTICE

Deleting a Cron job will not automatically delete its jobs. You can delete the jobs by running the **kubectl delete job** command.

----End

Related Operations

After a cron job is created, you can perform operations listed in **Table 7-19**.

Table 7-19 Other operations

Operation	Description
Viewing a YAML file	Click **More** > **View YAML** next to the cron job name to view the YAML file of the current job.

Operation	Description
Stopping a cron job	1. Select the job to be stopped and click **Stop** in the **Operation** column. 2. Click **OK**.
Deleting a cron job	1. Select the job to be deleted and click **More** > **Delete** in the **Operation** column. 2. Click **OK**. Deleted jobs cannot be restored. Therefore, exercise caution when deleting a job.

7.7 Managing a Workload

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

NOTE

This section uses a Deployment as an example to describe the workload management functions supported by CCE. If any of the following functions is not available on the CCE console, it is not supported yet.

Viewing Deployment Logs

You can view **logs** of Deployments.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads > Deployments**.

Step 2 In the same row as the Deployment you will view, click **Logs**.

In the displayed **Logs** window, view the logs generated in the last 5 minutes, 30 minutes, or 1 hour.

----End

Upgrading a Deployment

CCE enables you to quickly and hitlessly upgrade Deployments by replacing images or image versions.

Before replacing an image or image version, upload the new image to the SWR service.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**, and click **Upgrade** for the Deployment to be upgraded.

Step 2 Upgrade the Deployment.

- To replace the Deployment image, click **Replace Image** and select a new image.
- To replace the Deployment image version, select a new version from the **Image Version** drop-down list.

- To change the container name, click ✎ next to **Container Name** and enter a new name.
- Configure advanced settings.

Table 7-20 Advanced settings

Parameter	Description
Lifecycle	Commands that are executed in each lifecycle phase of a container. – **Startup Command**: executed when the container is started. For more information, see **Setting Container Startup Commands**. – **Post-start Processing**: executed after the container is successfully run. For more information, see **Setting Container Lifecycle Parameters**. – **Pre-stop Processing**: executed to delete logs or temporary files before the container stops. For more information, see **Setting Container Lifecycle Parameters**.
Health Check	The system checks whether containers and services are running properly. Two types of probes are set: workload liveness probe and workload service probe. For more information, see **Setting Health Check for a Container**. – **Workload Liveness Probe**: Restarts the container when detecting that the container instance is unhealthy. – **Workload Service Probe**: Sets the workload to the unready state when detecting that the workload's container instance is unhealthy. In this way, the service traffic will not be directed to the container instance.
Environment Variables	Environment variables are usually to store parameter values. To add environment variables to a container, click **Add Environment Variables** on the **Environment Variables** tab page. Currently, there are three types of environment variables: – **Added manually**: Set **Variable Name** and **Variable/ Variable Reference**. – **Added from Secret**: Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see **Creating a secret**. – **Added from ConfigMap**: Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see **Creating a ConfigMap**. **NOTE** To edit an environment variable that has been set, click **Edit**. To delete an environment variable that has been set, click **Delete**.

Parameter	Description
Data Storage	Local disks can be upgraded. For details, see **Using Local Disks as Storage Volumes**.
Security Context	Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
Log Policy	Set the container log collection policies and the log directory to collect container logs for unified management and analysis. For details, see **Collecting Standard Output Logs of Containers** and **Collecting Container Logs from Specified Paths**.

Step 3 Click **Submit**.

----End

Editing a YAML File

You can download and edit YAML files of Deployments through the online YAML editing window.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment you will edit, choose **Operation** > **More** > **Edit YAML**. In the **Edit YAML** window, edit the YAML file of the current Deployment.

Step 3 Click **Edit** and then **OK** to save the changes.

Step 4 (Optional) In the **Edit YAML** window, click **Download** to download the YAML file.

----End

Scaling a Deployment

A Deployment can be automatically resized according to custom scaling policies, freeing you from the efforts to manually adapt the amount of resources to fluctuating traffic load. This saves you big on both resources and labors.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment for which you will add a scaling policy, choose **Operation** > **More** > **Scaling**.

Step 3 On the **Scale** tab page, add or edit scaling policies. Scaling policies are classified as auto and manual scaling policies. For details, see **Scaling a Workload**.

----End

Monitoring a Deployment

You can view CPU and memory usage of Deployments on the CCE console.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 Click the name of the Deployment to be monitored. On the Deployment details page that is displayed, click the **Monitoring** tab to view CPU usage and memory usage of the Deployment.

Step 3 Click the **Instances** tab. Click ⌄ next to an instance to be monitored and click **Monitoring**.

Step 4 Check CPU usage and memory usage of the instance.

- CPU usage

  The horizontal axis indicates time while the vertical axis indicates the CPU usage. The green line indicates the CPU usage while the red line indicates the CPU usage limit.

  NOTE

  It takes some time to calculate CPU usage. Therefore, when CPU and memory usage are displayed for the first time, CPU usage is displayed about one minute later than memory usage.

  CPU and memory usage are displayed only for instances in the running state.

- Memory usage

  The horizontal axis indicates time while the vertical axis indicates the memory usage. The green line indicates the memory usage while the red line indicates the memory usage limit.

  NOTE

  Memory usage is displayed only for a running instance.

----End

Rolling Back a Deployment

The 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. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment you will roll back, choose **Operation** > **More** > **Roll Back**.

Step 3 In the **Roll Back to This Version** drop-down list, select the version to which you will roll back the Deployment. Then, click **OK**.

----End

Pausing a Deployment

You can pause Deployments. After a Deployment is paused, the upgrade command can be successfully issued but will not be applied to the pods.

If you are performing a rolling upgrade, the rolling upgrade stops after the pause command is delivered. In this case, the new and old pods co-exist.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment you will pause, choose **Operation** > **More** > **Pause**.

Step 3 In the displayed **Pause Workload** dialog box, click **OK**.

Step 4 Click **OK**.

NOTICE

Deployments in the paused state cannot be rolled back.

----End

Resuming a Deployment

You can resume paused Deployments. After a Deployment is resumed, 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.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment you will resume, choose **Operation** > **More** > **Resume**.

Step 3 In the displayed **Resume Workload** dialog box, click **OK**.

----End

Attaching Labels to Deployments

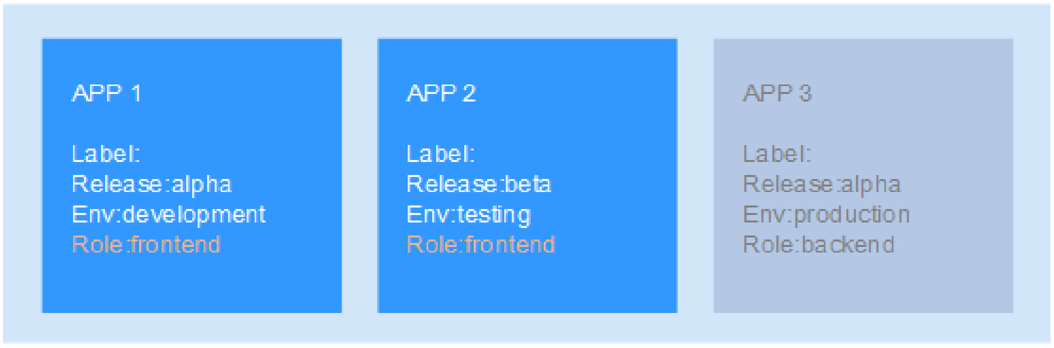
Labels are key-value pairs and can be attached to Deployments. Deployments can be easily selected based on labels. Usually, Deployment labels are used during affinity and anti-affinity scheduling. You can add labels to multiple Deployments or a specified Deployment.

In the following figure, three labels (release, env, and role) are defined for Deployments APP1, APP2, and APP3. The values of these labels vary with Deployments.

- Label of APP 1: [release:alpha;env:development;role:frontend]
- Label of APP 2: [release:beta;env:testing;role:frontend]
- Label of APP 3: [release:alpha;env:production;role:backend]

If you set **key** to **role** and **value** to **frontend** when using Deployment scheduling or another function, the function will apply to APP1 and APP2.

Figure 7-2 Label example



Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 Click the name of the Deployment whose labels will be managed.

Step 3 On the Deployment details page, click **Manage Labels** and **Add Label**. Enter the label key and value, and click **OK**.

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 Deployment

Delete a Deployment that will be no longer in use. Deleted Deployments cannot be recovered. Exercise caution when you perform this operation.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**.

Step 2 In the same row as the Deployment you will delete, choose **Operation** > **More** > **Delete**. Follow on-screen instructions to delete the Deployment.

Step 3 Click **OK**.

----End

7.8 Scaling a Workload

After scaling policies are defined, instances can be automatically added or deleted based on resource changes, fixed time, and fixed periods. This reduces manual resource adjustment to cope with service changes and peak pressure, helping you save resources and labor costs.

- **Auto Scaling**: includes alarm, scheduled, and periodic policies. This mode automatically scales in or out instances on a workload based on resource usage, scheduled time, or specified periods.
- **Manual Scaling**: Manually scale in or out instances on a workload immediately after the workload is created.

- **Maximum Number of Unavailable Pods**: 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 – Maximum number of unavailable pods

  NOTE

  If **Maximum Number of Unavailable Pods** is set to **0** when there is only one pod in the cluster, services will be interrupted during scaling.

Auto Scaling - HPA

This AS capability is implemented by creating HPA. You can view all policies or perform more operations in **Auto Scaling**.

Auto Scaling - AOM

You can define auto scaling policies as required, releasing you from the workload of repeatedly adjusting resources in response to service changes and heavy burden during peak hours and saving resource and labor costs.

The scaling capability is provided by the Application Operations Management (AOM) service. It is no longer supported in clusters of v1.17 and later.

Currently, CCE supports the following types of auto scaling policies:

Metric-based Policy: After a workload is created, pods will be automatically scaled when the workload's CPU or memory usage exceeds or falls below a preset limit.

Scheduled Policy: scaling at a specified time. Scheduled auto scaling is applicable flash sales, premier shopping events, and other regular events that bring a high burst of traffic load.

Periodic Policy: scaling at a specified time on a daily, weekly, or monthly basis. Periodic scheduling is applicable to scenarios where traffic changes periodically.

- **Metric-based Policy**: Supports automatic scaling of workload based on the setting of the CPU/memory.
  a. Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments** or **StatefulSets**. Click **Operation** > **More** > **Scaling** next to the workload name.
  b. In the **Auto Scaling** area, click **Add Scaling Policy**.

Table 7-21 Parameters for adding a metric-based policy

Parameter	Description
Policy Name	Enter the name of the scaling policy.
Policy Type	Set it to **Metric-based Policy**.

Parameter	Description
Metric	Set the metrics that describe the resource performance data or status. ● CPU Usage: indicates the CPU usage of the measured object, that is, the percentage of the CPU cores actually used by the measured object to the total CPU cores that the measured object has requested. ● Physical Memory Usage: indicates the percentage of the physical memory size used by the measured object to the physical memory size that the measured object has applied for.
Trigger Condition	Set it to **CPU Usage** or **Physical Memory Usage**. If you set this parameter to **Physical Memory Usage** and set the average value to be greater than 70%, the scaling policy is triggered when memory usage exceeds 70%.
Monitoring window	Metric statistics period. Select a value from the drop-down list box. If the parameter is set to 20s, metric statistics is collected every 20 seconds.
Consecutive Times	If the parameter is set to **3**, the action is triggered if threshold is reached for three consecutive measurement periods.
Action	Action executed after a policy is triggered. Two actions are available: add or reduce pods.

c. Click **OK**.
d. In the **Auto Scaling** area, check that the policy has been started.
   When the trigger condition is met, the auto scaling policy starts automatically.

- **Scheduled Policy:** scaling at a specified time.
  a. In the **Auto Scaling** area, click **Add Scaling Policy**. Select **Scheduled Policy**.

Table 7-22 Parameters for adding a scheduled policy

Parameter	Description
Policy Name	Enter the name of the scaling policy.
Policy Type	Set this parameter to **Scheduled Policy**.
Trigger Time	Time at which the policy is enforced.

Parameter	Description
Action	Action executed after a policy is triggered. Three actions are available: add pods, reduce pods, and set the number of pods.

b. Click **OK**.

c. In the **Auto Scaling** area, check that the policy has been started.

   When the trigger time is reached, you can see on the **Pods** tab page that the auto scaling policy has taken effect.

- **Periodic Policy:** scaling at a specified time on a daily, weekly, or monthly basis.

  a. In the **Auto Scaling** area, click **Add Scaling Policy**. Select **Periodic Policy**.

Table 7-23 Parameters for adding a periodic policy

Parameter	Description
Policy Name	Enter the name of the scaling policy.
Policy Type	Set this parameter to **Periodic Policy**.
Select Time	Specify the time for triggering the policy.
Action	Action executed after a policy is triggered. Three actions are available: add pods, reduce pods, and set the number of pods.

  b. Click **OK**.

  c. In the **Auto Scaling** area, check that the policy has been started.

     When the trigger condition is met, the auto scaling policy starts automatically.

Manual Scaling

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments** or **StatefulSets**. Click **Operation** > **More** > **Scaling** next to the workload name.

Step 2 In the **Manual Scaling** area, click ✎ and change the number of pods to, for example, **2**, then click **Save**. The scaling takes effect immediately.

Step 3 On the **Instances** tab page, check that a new instance is being created. When the instance status becomes **Running**, instance scaling is complete.

----End

Modifying Maximum Number of Unavailable Pods

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments** or **StatefulSets**. Click **Operation** > **More** > **Scaling** next to the workload name.

Step 2 Click ✎ to modify **Maximum Number of Unavailable Pods**.

- If **percentage** is not selected, **Maximum Number of Unavailable Pods** indicates the maximum quantity of unavailable pods and cannot exceed the total number of pods that the application has.
- If **percentage** is selected, **Maximum Number of Unavailable Pods** indicates the maximum percentage of unavailable pods. The value range is [0, 100].

Step 3 After the setting is completed, click **Save**.

----End

7.9 Using a Third-Party Image

CCE allows you to create workloads using images pulled from third-party image repositories.

Generally, you are required to pass authentication using your account and password before accessing a third-party image repository. The secret authentication is used for pulling images from a CCE container. Therefore, you must create a secret for accessing the image repository before pulling images.

Preparations

When using a third-party image, ensure that the node where the workload is running is accessible from public networks. For details about how to access public networks, see **LoadBalancer**.

By Using the Console

Step 1 Create a secret for accessing a third-party image repository.

In the navigation pane, choose **Configuration Center** > **Secrets**, and click **Create Secret**. **Secret Type** must be set to **dockerconfigjson**. For more information, see **Creating a secret**.

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

Step 2 Create a workload. For details, see **Creating a Deployment** or **Creating a StatefulSet**. If the workload will be created from a third-party image, set the image parameters as follows:

1. Set **Authenticate Secret** to **Yes**.
2. Select the secret created in step **Step 1**.
3. Enter the image address.

Step 3 Click **OK**.

----End

By Using CLI

Step 1 Configure kubectl. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Log in to the ECS on which kubectl has been configured.

Step 3 Create a secret of the dockercfg type using kubectl.

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

In the preceding commands, **myregistrykey** indicates the secret name, and other parameters are described 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 4 Use a third-party image to create a workload.

A dockecfg 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 key.
```

----End

8 Advanced Container Settings

8.1 Setting Container Specifications

CCE allows you to set resource restrictions for added containers during workload creation. You can apply for and limit the CPU and memory used by each instance of the workload.

NOTE

- After the CPU quota and memory quota are selected, select **Apply** to start the configuration. The system schedules the instance to the node that meets the requirements to deploy the workload based on the applied value. If you do not select **Apply**, the system schedules the instance to a random node to deploy the workload. If you select **Restrict**, the configuration is started and the resources used by the workload are restricted based on the configured value. If you do not select **Restrict**, the resources used by the instance are not restricted. If the memory resources used by the instance exceed the memory allocated to the node, the workload or node may be unavailable.
- When creating a workload, you are advised to set the upper and lower limits of CPU and memory resources. If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

- CPU quotas:

Table 8-1 Description of CPU quotas

Parameter	Description
CPU request	Minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available CPU on the node is greater than or equal to the number of containerized CPU applications.
CPU limit	Maximum number of CPU cores available for a container.

Recommended configuration method: The actual available CPU of a node ≥ The sum of CPU limits of all containers in the current pod ≥ The sum of CPU requests of all containers in the current pod. In the navigation pane, choose **Resource Management** > **Nodes**. Under the **Available CPU (Core)** column, view the actual available CPU of the node.

- Memory quotas:

Table 8-2 Description of memory quotas

Parameter	Description
Memory request	Minimum amount of memory required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available memory on the node is greater than or equal to the requested container memory.
Memory Limit	Maximum amount of memory available for a container. When the memory usage exceeds the configured memory limit, the pod may be restarted, which affects the normal use of the workload.

Recommended configuration method: The actual available memory of a node ≥ The sum of memory limits of all containers on the current node ≥ The sum of memory requests of all containers on the current node. In the navigation pane, choose **Resource Management** > **Nodes**. Under the **Available Memory (GB)** column, view the actual available memory of the node.

Example

Assume that a cluster contains a node with 4 cores and 8 GB. A workload containing two pods has been deployed on the cluster. The resources of the two pods (pods 1 and 2) are as follows: {CPU request, CPU limit, memory request, memory limit} = {1 core, 2 cores, 2 GB, 2 GB}.

The CPU and memory usage of the node are as follows:

- Allocatable CPU = 4 cores - (1 core applied by pod 1 + 1 core applied by pod 2) = 2 cores
- Allocatable memory = 8 GB - (2 GB applied by pod 1 + 2 GB applied by pod 2) = 4 GB

Therefore, the remaining 2 cores and 4 GB can be used by the next new pod.

8.2 Setting Container Lifecycle Parameters

CCE provides callback functions for the lifecycle management of containerized workloads. For example, if you want a container to perform a certain operation before stopping, you can register a hook function.

CCE provides the following lifecycle callback functions:

- **Start Command**: executed to start a container. For details, see **Setting Container Startup Commands**.
- **Post-Start Processing**: executed immediately after a workload is started. For details, see **Post-start Processing**.
- **Pre-Stop**: executed before the container is stopped. The pre-stop processing function helps you ensure that the services running in the instances can be completed in advance in the case of application upgrade or instance deletion. For details, see **Pre-stop Processing**.

Commands and Parameters Used to Run a Container

The Docker image has metadata that stores image information. If lifecycle commands and parameters are not set, CCE runs the default commands and parameters, that is, Docker native commands Entrypoint and CMD, provided during image creation. For details, see the description of **Entrypoint** and **CMD**.

If the commands and parameters used to run a container are set during workload creation, the default commands Entrypoint and CMD are overwritten during image building. The rules are as follows:

Table 8-3 Commands and parameters used to run a container

Image Entrypoint	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

Startup commands.

By default, the default command during image start. To run a specific command or rewrite the default image value, you must perform specific settings: For details, see **Setting Container Startup Commands**.

Post-start Processing

Step 1 Log in to the CCE console. Expand **Lifecycle** when creating a workload.

Step 2 Set the parameters for processing after startup, as listed in **Table 8-4**.

Table 8-4 Post-start processing parameters

Parameter	Description

CLI	Command to be executed in the container. The command format is **Command Args[1] Args[2]…**. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.
	Example command:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Enter **/install install_agent** in the script.
	This command indicates that **install_agent** will be executed after the container is created successfully.

HTTP request method	HTTP invocation request. The related parameters are described as follows: ● Path: (optional) request URL. ● Port: (mandatory) request port. ● Host address: (optional) IP address of the request. The default value is the IP address of the node where the container resides.

----End

Pre-stop Processing

Step 1 Log in to the CCE console. Click the **Pre-Stop Processing** tab during the lifecycle configuration when a workload is created.

Step 2 Set pre-stop parameters, as shown in **Table 8-4**.

Table 8-5 Pre-stop processing parameters

Parameter	Description

CLI	Set commands to be executed in the container for pre-stop processing. The command format is **Command Args[1] Args[2]…**. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.
	Example command:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Enter **/install install_agent** in the script.
	This command indicates that **install_agent** will be executed after the container is created successfully.

HTTP request method	HTTP invocation request. The related parameters are described as follows: ● Path: (optional) request URL. ● Port: (mandatory) request port. ● **Host Address**: (optional) IP address of the request. The default value is the IP address of the node where the container resides.

----End

Example YAML for Setting the Container Lifecycle

This section uses an Nginx application as an example to describe how to set the container lifecycle.

Prerequisite

You have configured the kubectl and connected an to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name, and you can change it as required.

vi nginx-deployment.yaml

In the following configuration file, the **postStart** command is defined to run the **install.sh** command in the **/bin/bash** directory. **preStop** is defined to run the **uninstall.sh** command.

```
apiVersion: extensions/v1beta1
kind: Deployment
```

```
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  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
```

----End

8.3 Setting Container Startup Commands

When creating a workload or job, you can use an image to specify the processes running in the container.

By default, the image runs the default command. To run a specific command or rewrite the default image value, you must perform the following settings:

- **Working directory**: working directory of the command.

  If the working directory is not specified in the image or on the console, the default value is **/**.
- **Command**: command that controls the running of an image.
- **Args**: parameters transferred to the running command.

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.

Commands and Parameters Used to Run a Container

A Docker image has metadata related to the stored image information. If no lifecycle command or parameter is set, the container runs the default command and parameter provided during image creation. The Docker defines the two fields as Entrypoint and CMD. For details, see **Entrypoint Description** and **CMD Description** in Docker documentation.

If the commands and parameters used to run a container are set during workload creation, the default commands Entrypoint and CMD are overwritten during image building. The rules are as follows:

Table 8-6 Commands and parameters used to run a container

Image Entrypoint	Image CMD	Command to Run a Container	Args 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]

Setting the Startup Command

Step 1 Log in to the CCE console. Expand **Lifecycle** when creating a workload or task.

Step 2 Enter the running command and parameters, as shown in **Table 8-7**.

NOTE

- The current startup command is provided as a string array and corresponds to the Entrypoint startup command of Docker. The format is as follows: ["executable", "param1", "param2",..]. For details about how to start the Kubernetes container, click **here**.
- The lifecycle of the container is the same as that of the startup command. That is, the lifecycle of the container ends after the command is executed.

Table 8-7 Container startup command

Configuration Item	Procedure
Command	Enter an executable command, for example, **/run/server**. If there are multiple commands, separate them with spaces. If the command contains a space, you need to add a quotation mark (""). **NOTE** In the case of multiple commands, you are advised to run the **/bin/sh** or other **shell** commands. Other commands are used as parameters.
Args	Enter the argument that controls the container running command, for example, **--port=8080**. If there are multiple arguments, separate them in different lines.

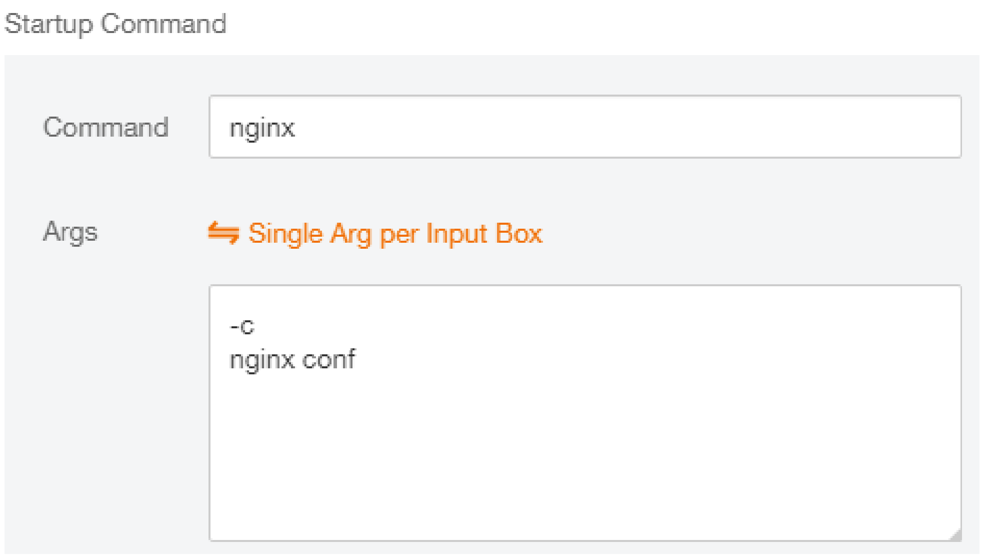
The following uses Nginx as an example to describe three typical application scenarios of the container startup command:

Example code:

```
nginx -c nginx.conf
```

- Scenario 1: Both the **command** and **arguments** are set.

Figure 8-1 Setting the startup command and parameters



Command: nginx

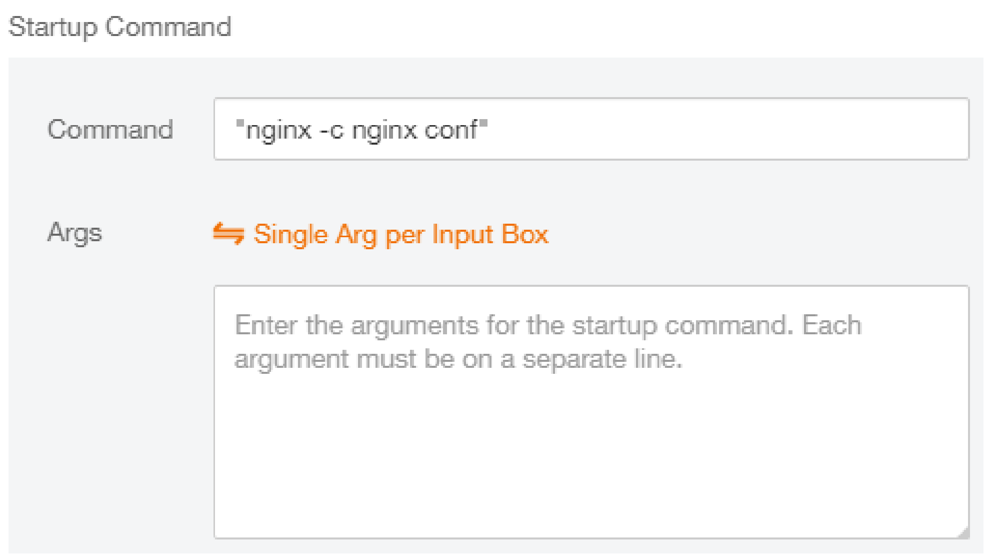
Args:
```
-c
nginx.conf
```

Example YAML file:

```
command:
  - nginx
args:
  - '-c'
  - nginx.conf
```

- Scenario 2: Only the **command** is set.

Figure 8-2 Setting the startup command


Startup Command

Command: "nginx -c nginx conf"

Args: Single Arg per Input Box

Enter the arguments for the startup command. Each argument must be on a separate line.

NOTE

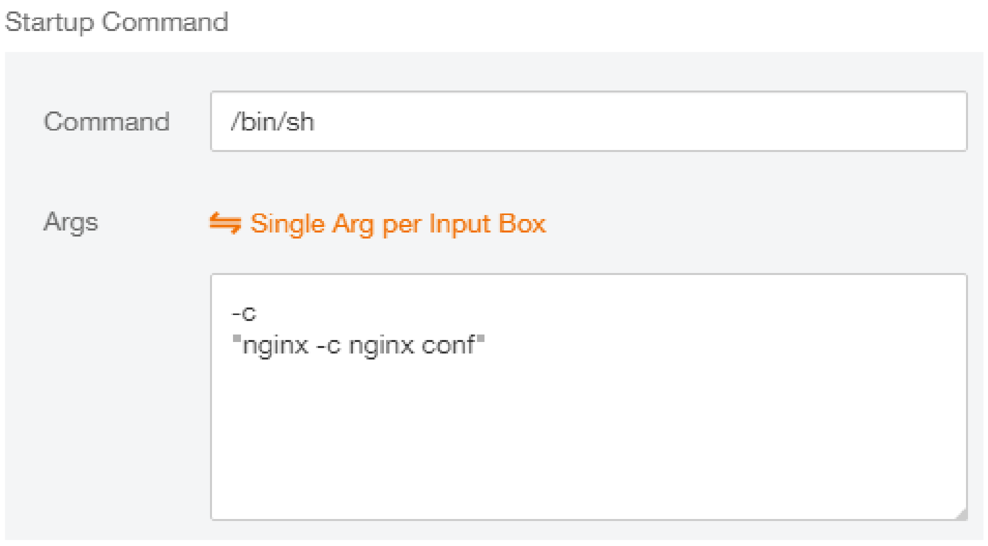
A command must be enclosed in double quotes. If no double quotes are added, the command is split into multiple commands based on space character.

Example YAML file:

```
command:
  - nginx -c nginx.conf
args:
```

- Scenario 2: Only **arguments** are set.

Figure 8-3 Setting startup arguments


Startup Command

Command: /bin/sh

Args: Single Arg per Input Box

```
-c
"nginx -c nginx conf"
```

NOTE

If the container startup command is not added to the system path, run the **/bin/sh** command to execute the container startup command. The container startup command must be enclosed in double quotes.

Example YAML file:

```
command:
  - /bin/sh
args:
  - '-c'
  - '"nginx -c nginx.conf"'
```

Step 3 Check or modify the YAML file.

- When creating a workload, in the **Configure Advanced Settings** step, click YAML on the right.
- After the workload is created, go to the workload list. In the same row as the workload choose **More** > **Edit YAML**.
- After the workload is created, go to the workload details page. On the displayed page, click **Edit YAML** in the upper right corner.

----End

Example YAML for Setting the Container Lifecycle

This section uses Nginx an example to describe how to set container startup commands using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, set the parameters in the container startup command as follows. For details, see the **official Kubernetes documents**.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  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
```

8.4 Setting Health Check for a Container

Health check regularly checks the health status of containers during container running. If the health check function is not configured, a pod cannot detect service exceptions or automatically restart the service to restore it. This will result in a situation where the pod status is normal but the service in the pod is abnormal.

CCE provides the following health check probes:

- **Liveness Probe**: checks whether a container exists. 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. If the liveness check of a container fails, the cluster restarts the container.
- **Readiness Probe**: checks whether a container is ready to process user requests. Upon detecting that the container is 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. For example, loading disk data or relying on startup of an external module. In this case, the application process is running, but the application cannot provide services. This check method is useful in this scenario. 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.

Health Check Methods

- **HTTP request mode**

  This health check mode is applicable 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, if you have an HTTP service container, after you specify port 80 for container listening and the HTTP request path **/health-check**, the cluster periodically initiates the **GET http://containerIP:80/health-check** request to the container.

- **TCP link setup mode**

  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 configure TCP port probe for the container and specify port 80 for the probe, the cluster periodically initiates a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

- **CLI mode**

  The CLI mode is an efficient health check mode. In this mode, you must specify an executable command in a container. The cluster will periodically execute 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 following modes:

  – TCP Link Setup Mode: Write a program script to connect to a container port. If the connection is successful, the script returns 0. Otherwise, the script returns –1.

  – HttpGet Request Mode: Write a program script to run the wget command for a 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.

    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 interpreter. 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.

Common Parameter Description

Table 8-8 Common parameter description

Parameter	Description
Delay Time (s)	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 Time	Timeout duration. 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.

8.5 Setting an Environment Variable

An environment variable is a variable set in the runtime environment of a container. Environment variables can be modified after the workload is deployed, providing flexibility for workloads.

The function of setting environment variables on CCE is the same as that of specifying **ENV** in a Dockerfile.

CCE provides three ways to add environment variables: Manually add environment variables, import environment variables from a secret, and import environment variables from a configMap.

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.

Manually Adding Environment Variables

Step 1 When creating a workload, after adding a container image, expand **Environment Variables**, and click **Add Environment Variables**.

Step 2 Configure the following parameters as required:

- **Type**: Select **Added manually**.
- **Variable Name**: Enter a variable name, for example, demo.
- **Variable Value/Reference**: Enter a variable value, for example, value.

----End

Importing Environment Variables from a Secret

Step 1 You need to create a key first. For details, see **Creating a secret**.

Step 2 When creating a workload, add a container image. Then, expand **Environment Variables** and click **Add Environment Variables**.

Step 3 Configure the following parameters as required:

- **Type**: Select **Added from secret**.
- **Variable Name**: Enter a variable name.
- **Variable/Variable Reference**: Select the corresponding secret name and key.

----End

Importing Environment Variables from a configMap

Step 1 You need to create a configMap first. For details, see **Creating a ConfigMap**.

Step 2 When creating a workload, add a container image. Then, expand **Environment Variables** and click **Add Environment Variables**.

Step 3 Configure the following parameters as required:

- **Type**: Select **Added from ConfigMap**.
- **Variable Name**: Enter a variable name.
- **Variable/Variable Reference**: Select the corresponding configMap name and key.

----End

8.6 Collecting Standard Output Logs of Containers

CCE allows you to configure policies for collecting, managing, and analyzing workload logs periodically to prevent logs from being over-sized.

This topic describes how to collect standard output logs of containers. If you want to configure a log policy to collect container logs from specified paths, see **Collecting Container Logs from Specified Paths**.

Procedure

Step 1 When creating a workload, add a container and expand **Log Policy**.

Step 2 Retain the default settings to complete the workload creation. (Nginx is used as an example.)

Step 3 View logs.

After the workload is created, access Nginx. Go to the workload details page and click the **Logs** button in the upper right corner to view the log details. Wait for about 5 minutes to view logs.

----End

8.7 Collecting Container Logs from Specified Paths

CCE allows you to configure policies for collecting, managing, and analyzing workload logs periodically to prevent logs from being over-sized.

This topic describes how to collect container logs from specified paths. If you want to collect standard output logs of containers, see **Collecting Standard Output Logs of Containers**.

Procedure

Step 1 When creating a containerized workload, add a container and expand **Log Policy**.

Step 2 Click **Add Log Policy**, set custom log parameters, and configure the log policy. The following uses Nginx as an example. Configure the log policy based on service requirements.

Step 3 Set **Storage Type** to **Host Path** or **Container Path**.

- **Host Path**: You can mount a host path to a specified container path. Set parameters according to the following table.

Table 8-9 Parameters for adding log policies (host path)

Parameter	Description
Storage Type	Set this parameter to **Host Path**.
Add Container Path	
*Host Path	Enter the path of the host, for example, **/var/paas/sys/log/nginx**.
Container Path	Container path to which a data volume is mounted. Example: **/tmp** **NOTICE** – Do not mount log files to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. – If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged.

Parameter	Description
Extended Host Path	A level-3 directory is added to the original volume directory/subdirectory. You can easily obtain the files output by a single Pod. – None: No extended path is configured. – PodUID: ID of a pod. – PodName: Name of a pod. – PodUID/ContainerName: ID of a pod or name of a container. – PodName/ContainerName: Pod name/container name.
Collection Path	Path for collecting logs precisely. Details are as follows: – If no collection path is specified, log files in **.log**, **.trace**, and **.out** formats will be collected from the current path by default. – **/Path/**/** indicates that all log files in **.log**, **.trace**, and **.out** formats will be recursively collected from the specified path and all subdirectories at 5 levels deep. – * in log file names indicates a fuzzy match. Example: If the collection path is **/tmp/**/test*.log**, all **.log** files prefixed with **test** will be collected from **/tmp** and its 5 levels of subdirectories. **CAUTION** To use the collection path function, ensure that the ICAgent version is 5.12.22 or later.
Log Dumping	Log files in the format of **.log, .trace, and .out.** are supported. If the size of a log file is greater than 50 MB, a maximum of 20 files are dumped. – **Enabled**: Log files are dumped. – **Disabled**: Log files are not dumped.

- **Container Path**: Logs will be stored in a container path. No host path needs to be mounted into the container. Set parameters according to the following table.

  NOTE

  Ensure that the ICAgent version is 5.10.79 or later.

Table 8-10 Parameters for adding log policies (container path)

Parameter	Description
Storage Type	Set this parameter to **Container Path**.
Add Container Path	

Parameter	Description
Container Path	Container path to which a data volume is mounted. Example: **/tmp** **NOTICE** – Do not mount log files to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. – If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host may be damaged.
Collection Path	Path for collecting logs precisely. Details are as follows: – If no collection path is specified, log files in **.log**, **.trace**, and **.out** formats will be collected from the current path by default. – **/Path/**/** indicates that all log files in **.log**, **.trace**, and **.out** formats will be recursively collected from the specified path and all subdirectories at 5 levels deep. – * in log file names indicates a fuzzy match. Example: If the collection path is **/tmp/**/test*.log**, all **.log** files prefixed with **test** will be collected from **/tmp** and its 5 levels of subdirectories. **CAUTION** To use the collection path function, ensure that the ICAgent version is 5.12.22 or later.
Log Dumping	Log files in the format of **.log, .trace, and .out.** are supported. If the size of a log file is greater than 50 MB, a maximum of 20 files are dumped. – **Enabled**: Log files are dumped. – **Disabled**: Log files are not dumped.

Step 4 Click **OK**. The workload is created.

Step 5 View logs.

After the workload is created, access Nginx. Go to the workload details page and click the **Log** button in the upper right corner to view the log details.

----End

9 Affinity and Anti-Affinity Scheduling

9.1 Scheduling Policy Overview

The CCE supports both simple and custom scheduling policies. A simple scheduling policy includes basic scheduling settings and is easy to configure, while a custom scheduling policy allows you to configure advanced scheduling (node affinity and workload affinity/anti-affinity).

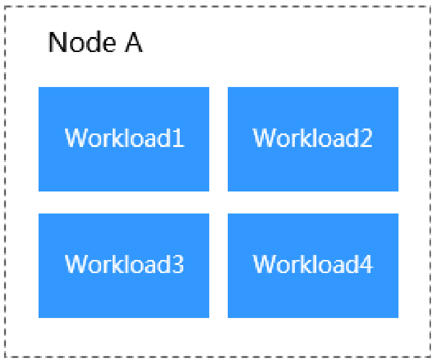
Simple Scheduling Policies

A simple scheduling policy allows you to configure affinity between workloads and AZs, between workloads and nodes, and between workloads.

- **Workload-AZ Affinity and Anti-Affinity:**
  - **Affinity between workloads and AZs**: **Workload-AZ Affinity**
  - **Anti-affinity between workloads and AZs**: **Workload-AZ Anti-Affinity**
- **Workload-Node Affinity and Anti-Affinity**
  - **Affinity between workloads and nodes**: **Workload-Node Affinity**
  - **Anti-affinity between workloads and nodes**: **Workload-Node Anti-Affinity**
- **Workload-Workload Affinity and Anti-Affinity:** Determines whether workloads are deployed in the same topology domain.
  - **Affinity between workloads**: For details, see **Workload-Workload Affinity**. to reduce consumption of network resources.

    As shown in **Figure 9-1**, Workload1, Workload2, Workload3, and Workload4 are deployed on the same node in affinity mode.

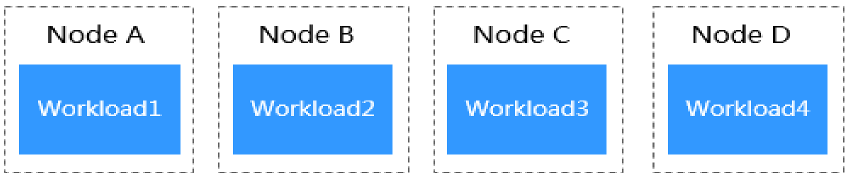
Figure 9-1 Affinity between workloads



– **Anti-affinity between workloads**: For details, see **Workload-Workload Anti-Affinity**. Constraining multiple instances of the same workload from being deployed on the same node reduces the impact of system breakdowns. Anti-affinity deployment is also recommended for workloads that may interfere with each other.

  Figure 9-2 shows an example of anti-affinity deployment, in which four workloads are deployed on four different nodes.

Figure 9-2 Inter-workload anti-affinity



NOTICE

When setting workload-workload and workload-node affinity/anti-affinity, ensure that the affinity relationships do not contradict each other; otherwise, workload deployment will fail. For example, workload deployment will fail for app 3 when the following conditions are met:

- Anti-affinity is configured for Workload1 and Workload2. Workload1 is deployed on Node A and Workload2 is deployed on Node B.
- Affinity is configured between Workload2 and Workload3, but anti-affinity is configured between Workload3 and Node C.

Custom Scheduling Policies

You can configure node affinity, workload affinity, and workload anti-affinity in a custom scheduling policy. For details, see **Affinity and anti-affinity**.

- **Node Affinity**

- **Workload Affinity**
- **Workload Anti-Affinity**

NOTE

Custom scheduling policies depend on node labels and pod labels. You can use default labels or customize labels as required.

9.2 Simple Scheduling Policies

9.2.1 Workload-AZ Affinity

Using the CCE Console

Step 1 When **Creating a Deployment** or **Creating a StatefulSet**, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, click next to **Workload-AZ Affinity and Anti-affinity** > **Affinity with AZs**.

Step 2 Select the AZ in which you want to deploy the workload.

The created workload will be deployed in the selected AZ.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure workload-AZ affinity. The following is an example YAML file for workload-AZ affinity.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: az-in-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: az-in-deployment
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: az-in-deployment
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: failure-domain.beta.kubernetes.io/zone #Node's label key
                operator: In
                values:
                - az1                           #Node's key value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Affinity Object**.

Step 3 Set **Object Type** to **Availability Zone**, and select the AZ in which the workload is eligible to be deployed. The workload will be deployed in the selected AZ.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.2.2 Workload-AZ Anti-Affinity

Using the CCE Console

Step 1 When **Creating a Deployment** or **Creating a StatefulSet**, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, click ⌄ next to **Workload-AZ Affinity and Anti-affinity** > **Anti-affinity with AZs**.

Step 2 Select an AZ in which the workload is ineligible to be deployed.

The created workload is not deployed on the selected AZ.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure workload-AZ anti-affinity. The following is an example YAML file for workload-AZ anti-affinity.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: failure-domain.beta.kubernetes.io/zone      #Node's label key
                operator: NotIn
                values:
                - az1                              #Node's key value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Anti-affinity Object**.

Step 3 Set **Object Type** to **Availability Zone** and select the AZ in which the workload is ineligible to be deployed. The workload will be constrained from being deployed in the selected AZ.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.2.3 Workload-Node Affinity

Using the CCE Console

Step 1 When **Creating a Deployment** or **Creating a StatefulSet**, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Workload-Node Affinity and Anti-affinity** > **Affinity with Nodes** > **Add**.

Step 2 Select the node on which you want to deploy the workload, and click **OK**.

If you select multiple nodes, the system automatically chooses one of them during workload deployment.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure workload-node affinity. The following is an example YAML file for workload-node affinity.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: nodeName          #Node's label key
                operator: In
                values:
                - test-node-1          #Node's label value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Affinity Object**.

Step 3 Set **Object Type** to **Node** and select the node where the workload is to be deployed. The workload will be deployed on the selected node.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.2.4 Workload-Node Anti-Affinity

Using the CCE Console

Step 1 When **Creating a Deployment** or **Creating a StatefulSet**, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Workload-Node Affinity and Anti-affinity** > **Anti-affinity with Nodes** > **Add**.

Step 2 Select the node on which the workload is ineligible to be deployed, and click **OK**.

If you select multiple nodes, the workload will not be deployed on these nodes.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure workload-node anti-affinity. The following is an example YAML file for workload-node anti-affinity.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
```

```
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: nodeName          #Node's label key
            operator: NotIn        #Indicates that the workload will not be deployed on the node.
            values:
            - test-node-1          #Node's label value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Anti-affinity Object**.

Step 3 Set **Object Type** to **Node** and select the node on which the workload is ineligible to be deployed. The workload will be constrained from being deployed on the selected node.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.2.5 Workload-Workload Affinity

Using the CCE Console

Step 1 When **Creating a Deployment** or **Creating a StatefulSet**, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Inter-Pod Affinity and Anti-affinity** > **Affinity with Pods** > **Add**.

Step 2 Select the workloads that will be co-located with the current workload on the same node, and click **OK**.

The workload being created will be deployed on the same node as the selected workloads.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure affinity between workloads. The following is an example YAML file for affinity between workloads.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: app          #Workload's label key
                operator: In
                values:
                - test     #Workload's label value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Affinity Object**.

Step 3 Set **Object Type** to **Workload** and select the workloads to be deployed on the same node as the created workload. The created workload and the selected workloads will be deployed on the same node.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.2.6 Workload-Workload Anti-Affinity

Using the CCE Console

Step 1 When Creating a Deployment or Creating a StatefulSet, in the **Scheduling Policies** area on the **Configure Advanced Settings** page, choose **Inter-Pod Affinity and Anti-affinity** > **Anti-affinity with Pods** > **Add**.

Step 2 Select the workloads to which you want to deploy the target workload on a different node, and click **OK**.

The workload to be created and the selected workloads will be deployed on different nodes.

----End

Using kubectl

This section uses a Nginx workload as an example to describe how to create a workload using kubectl.

Prerequisite

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

When **using kubectl to create a Deployment** or **using kubectl to create a StatefulSet**, configure anti-affinity between workloads. The following is an example YAML file for anti-affinity between workloads.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: app          #Workload's label key
                operator: NotIn
                values:
                - test     #Workload's label value
```

Adding a Service After Creating a Workload

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click the name of the workload for which you will add a scheduling policy. On the workload details page, choose **Scheduling Policies** > **Add Simple Scheduling Policy** > **Add Anti-affinity Object**.

Step 3 Set **Object Type** to **Workload** and select the workloads to be deployed on a different node from the created workload. The created workload and the selected workloads will be deployed on different nodes.

NOTE

This method can be used to add, edit, or delete scheduling policies.

----End

9.3 Custom Scheduling Policies

9.3.1 Node Affinity

Using the CCE Console

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.

Step 3 In the **Node Affinity** area, configure scheduling rules using node labels.

NOTE

Node affinity scheduling rules can be **Required** or **Preferred**, and the label operators include In, NotIn, Exists, DoesNotExist, Gt, and Lt.

- **Required**: This is a hard rule that must be met for scheduling. It corresponds to **requiredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Multiple required rules can be set, and scheduling will be performed if only one of them is met.
- **Preferred**: This is a soft rule specifying preferences that the scheduler will try to enforce but will not guarantee. It corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Multiple required rules can be set, and scheduling will be performed even if only one or none of them are met. You can set the weight for a preferred rule. A higher weight indicates a higher priority.
- **Selector**: This corresponds to **matchExpressions** in Kubernetes. Multiple selectors can be set for a scheduling rule, and all of them need to be met for the rule to take effect.
- **Label**: This indicates the node label. You can use default or custom labels.
- **Operator**: Six operators are provided for you to configure label matching relationships: In, NotIn, Exists, DoesNotExist, Gt, and Lt). Operators In and NotIn allow one or more label values. Values are separated with colons (;). Operators Exists and DoesNotExist are used to determine whether a label exists, and do not require a label value. If you set the operator to Gt or Lt for a label, the label value must be greater than or less than a certain integer.

----End

Using kubectl

This section uses Nginx as an example to describe how to configure node affinity.

Prerequisite

A workload that uses the nginx container image has been deployed on a node.

Procedure

Set **Label** to **kubernetes.io/hostname**, add affinity nodes, and set the operator to **In**. Then, click **OK**.

YAML file of the workload with node affinity:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      imagePullSecrets:
        - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: kubernetes.io/hostname
                    operator: In
                    values:
                      - 192.168.6.174
```

9.3.2 Pod Affinity

Using the CCE Console

Pod affinity determines the pods as which the target workload will be deployed in the same topology domain.

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.

Step 3 In the **Pod Affinity** area, set the namespace, topology key, and the label requirements to be met.

NOTE

There are two types of pod affinity rules: Required (hard rule) and Preferred (soft rule), and the label operators include In, NotIn, Exists, and DoesNotExist.

- **Required**: This is a hard rule that must be met for a pod to be scheduled onto a node. The rule is expressed in the format of **requiredDuringSchedulingIgnoredDuringExecution**. A pod can have multiple hard rules. Each rule requires a namespace and topology key.
- **Preferred**: This is a soft rule specifying preferences that the scheduler will try to enforce but will not guarantee. It corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Multiple required rules can be set, and scheduling will be performed even if only one or none of them are met. You can set the weight for a preferred rule. A higher weight indicates a higher priority.
- **Namespace**: By default, the namespace of the current pod is used. You can also use another namespace.
- **Topology Key**: Key of the node label that the system uses to denote a topology domain in which scheduling can be performed. Default and custom node labels can be used.
- **Selector**: This corresponds to **matchExpressions** in Kubernetes. Multiple selectors can be set for a scheduling rule, and all of them need to be met for the rule to take effect.
- **Label**: Pod label. You can use the default label **app** or custom labels.
- **Operator**: Four operators are provided for you to configure label matching relationships: In, NotIn, Exists, and DoesNotExist. Operators In and NotIn allow one or more label values. Values are separated with colons (;). Operators Exists and DoesNotExist are used to determine whether a label exists, and do not require a label value.

----End

Using kubectl

This section uses Nginx as an example to describe how to configure node affinity.

Prerequisite

A workload that uses the nginx container image has been deployed on a node.

Procedure

Set **Namespace** to **default** and **Topology Key** to the built-in node label **kubernetes.io/hostname**, which means that the scheduling scope is a node. Set labels **app** and **type** and their value to **redis** and **database**, respectively. Set **Operator** to **In** and click **OK**.

The YAML of the workload with pod affinity is as follows:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```

```
    app: nginx
  spec:
    imagePullSecrets:
      - name: default-secret
    affinity:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
                - key: app
                  operator: In
                  values:
                    - redis
                    - key: type
                  operator: In
                  values:
                    - database
                  namespaces:
                    - default
              topologyKey: kubernetes.io/hostname
```

NOTICE

In this example, only when a candidate workload (for example, workload A) with both labels **app=redis** and **type=database** is found can the workload Nginx be successfully scheduled to the node of the candidate workload.

9.3.3 Pod Anti-Affinity

Using the CCE Console

Pod anti-affinity determines the pods to which the target workload will be deployed in a different topology domain.

Step 1 Log in to the CCE console, and choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** in the navigation pane.

Step 2 Click a workload name in the Deployment or StatefulSet list. On the displayed workload details page, click the **Scheduling Policies** tab and then click **Add Custom Scheduling Policy**.

Step 3 In the **Pod Anti-Affinity** area, set the namespace, topology key, and the label requirements to be met.

NOTE

There are two types of pod anti-affinity rules: Required (hard rule) and Preferred (soft rule), and the label operators include In, NotIn, Exists, and DoesNotExist.

- **Required**: This is a hard rule that must be met for a pod to be scheduled onto a node. The rule is expressed in the format of **requiredDuringSchedulingIgnoredDuringExecution**. A pod can have multiple hard rules. Each rule requires a namespace and topology key.
- **Preferred**: This is a soft rule specifying preferences that the scheduler will try to enforce but will not guarantee. It corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Multiple required rules can be set, and scheduling will be performed even if only one or none of them are met. You can set the weight for a preferred rule. A higher weight indicates a higher priority.
- **Namespace**: By default, the namespace of the current pod is used. You can also use another namespace.
- **Topology Key**: Key of the node label that the system uses to denote a topology domain in which scheduling can be performed. Default and custom node labels can be used.
- **Selector**: This corresponds to **matchExpressions** in Kubernetes. Multiple selectors can be set for a scheduling rule, and all of them need to be met for the rule to take effect.
- **Label**: Pod label. You can use the default label **app** or custom labels.
- **Operator**: Four operators are provided for you to configure label matching relationships: In, NotIn, Exists, and DoesNotExist. Operators In and NotIn allow one or more label values. Values are separated with colons (;). Operators Exists and DoesNotExist are used to determine whether a label exists, and do not require a label value.

----End

Using kubectl

This section uses Nginx as an example to describe how to configure pod anti-affinity.

Prerequisite

A workload that uses the nginx container image has been deployed on a node.

Procedure

Set **Namespace** to **default** and **Topology Key** to the built-in node label **kubernetes.io/hostname**, which means that the scheduling scope is a node. Set the label **app** and its value to **redis**. Set **Operator** to **In** and click **OK**.

The YAML of the workload with pod anti-affinity:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```

```
      app: nginx
    spec:
      imagePullSecrets:
        - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - redis
              namespaces:
                - default
              topologyKey: kubernetes.io/hostname
```

10 Network Management

10.1 Network Overview

CCE provides different workload access types to address diverse scenarios.

Constraints

- A maximum of 6,000 services can be created in each namespace. The service mentioned here refers to the Kubernetes Service resource which defines a logical set of pods and a policy by which to access them.
- If containers running with hostPort or hostNetwork need to be accessed by external networks, the container ports corresponding to the nodes where the containers are located must be enabled.

Network Capability

CCE works seamlessly with the Kubernetes network and VPC to provide a stable and high-performance container network. It provides the following workload access services:

- **Intra-cluster access (ClusterIP)**

  A workload can be accessed from other workloads in the same cluster through a cluster-internal domain name. A cluster-internal domain name is in the format of *<User-defined service name>.<Namespace of the workload>***.svc.cluster.local**, for example, **nginx.default.svc.cluster.local**. For details, see **Intra-Cluster Access (ClusterIP)**.
- **NodePort**

  A service is exposed on each node's IP address at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. By requesting the <NodeIP>:<NodePort>, you can access a NodePort service from the outside of the cluster. For details, see **NodePort**.

  NodePort access is further classified into intra-Virtual Private Cloud (VPC) access or elastic IP address (EIP)-based access.
  - Intra-VPC access: If the workload that the NodePort service targets runs inside a Kubernetes cluster and does not have an EIP, it can still be accessible to workloads in the same VPC by using a node IP address.

  - EIP-based access: If the NodePort service needs to be exposed to public networks, the workload that the NodePort service targets can be accessed at an EIP from public networks. This can happen only after an EIP is bound to any node in the cluster and a service port is mapped to a node port in the 30000–32767 range. For example, the access address could be 10.117.117.117:30000.

- **LoadBalancer**

  A workload can be accessed from public networks through a load balancer. LoadBalancer provides higher reliability than EIP-based NodePort because an EIP is no longer bound to a single node. The LoadBalancer access type is applicable to the scenario in which a service exposed to public networks is required. The access address is in the format of <IP address of public network load balancer>:<access port>. For example, 10.117.117.117:80. For details, see **LoadBalancer**.

- **Layer-7 load balancing (ingress)**

  The difference between ingress and layer-4 load balancing is that ingress supports uniform resource identifier (URI) configuration and distributes access traffic to services based on the URI. In addition, the services implement different functions based on URIs.

  Ingress uses enhanced load balancers. The access address is in the format of <public network load balancer service address>:<access port>/defined URI. For example, 10.117.117.117:80/helloworld. For details, see **Layer-7 Load Balancing (Ingress)**.

Layer-7 Load Balancing (Ingress)

Generally, the IP addresses of services and pods are accessible to workloads in the same cluster. An external request needs to be forwarded by a load balancer to the NodePort exposed by the service on a node and then be forwarded by kube-proxy to corresponding pods through an edge router or be discarded by kube-proxy. Ingress is a set of rules that route external requests to the cluster.

Ingress provides the URL, load balancer, SSL offloading, and HTTP route for external access to the cluster. To configure these ingress rules, the cluster administrator needs to deploy an ingress controller to monitor changes of ingresses and services, configure load balancing based on the rules, and provide access entries.

An ingress object consists of:

- Nginx: implements load balancing among pods.
- Ingress controller: obtains the IP addresses of the pods corresponding to services from cluster APIs and adds the IP addresses to the Nginx configuration file.
- Ingress: creates virtual hosts for Nginx.

For details on how to create and manage ingresses, see **Layer-7 Load Balancing (Ingress)**.

Network Policy

Network policies define communication rules between pods and between a pod and other network endpoints.

The NetworkPolicy resource selects pods by labels and defines the communication rules allowed by the selected pods. For details, see **Network Policy**.

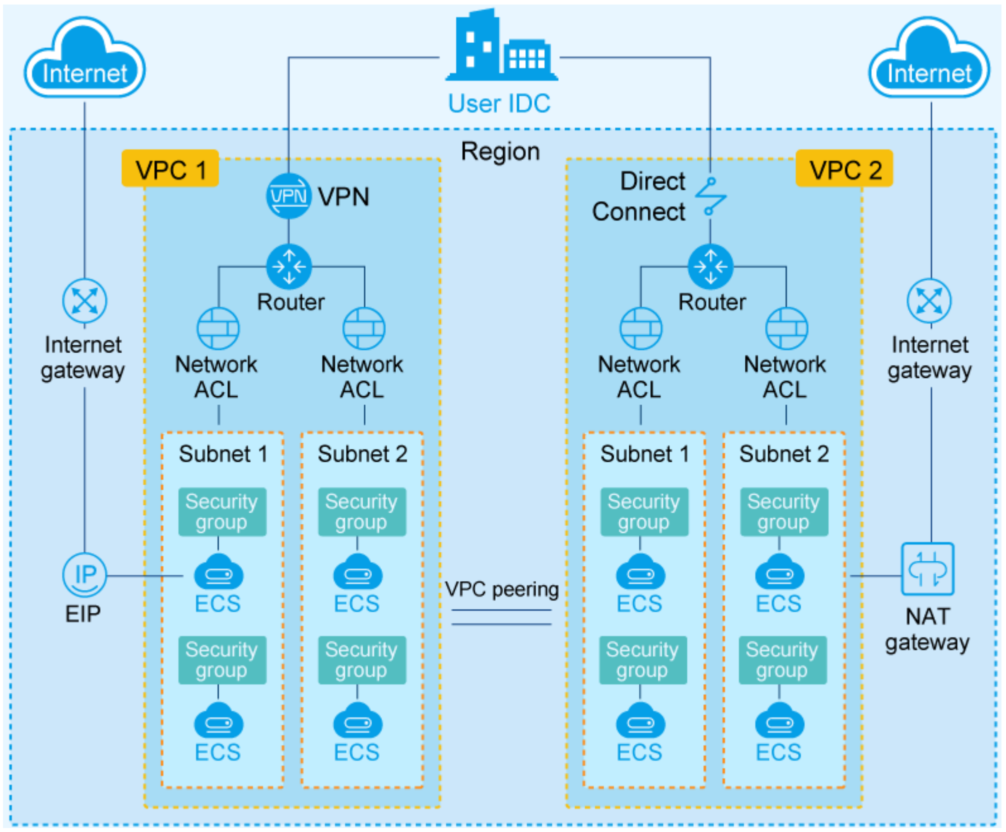
Network Infrastructure

- **VPC**

  A Virtual Private Cloud (VPC) is a private network created based on the cloud. Different private networks are logically isolated. VPC enables you to provision logically isolated, configurable, and manageable virtual networks for Elastic Cloud Servers (ECSs), cloud databases, and load balancers, improving cloud service security and simplifying network deployment.

  You can configure security groups, virtual private networks (VPNs), and IP address segments, and allocate bandwidth in a VPC. With a VPC, you can configure and manage the networks within the VPC, making changes to these networks as needed, quickly and securely. You can also customize the ECS access rules within a security group and between security groups to strengthen ECS security protection.

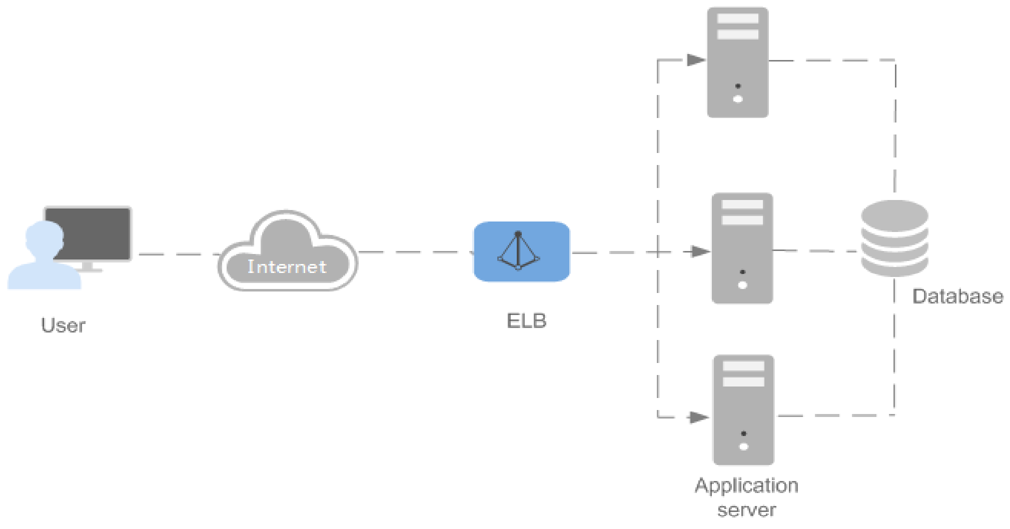
  Figure 10-1 VPC components



- **ELB**

  Elastic Load Balance (ELB) automatically distributes incoming traffic across multiple backend servers based on the rules you configure. This expands service capabilities of applications and improves their availability by eliminating single points of failure (SPOFs).

Figure 10-2 ELB



ELB provides two types of load balancers:

- Classic load balancer: This type of load balancer can well handle web services with low access traffic and simple applications.
- Enhanced load balancer: This type of load balancer is suitable for web services with high access traffic. Requests are forwarded based on domain names or URLs, making request routing more flexible.

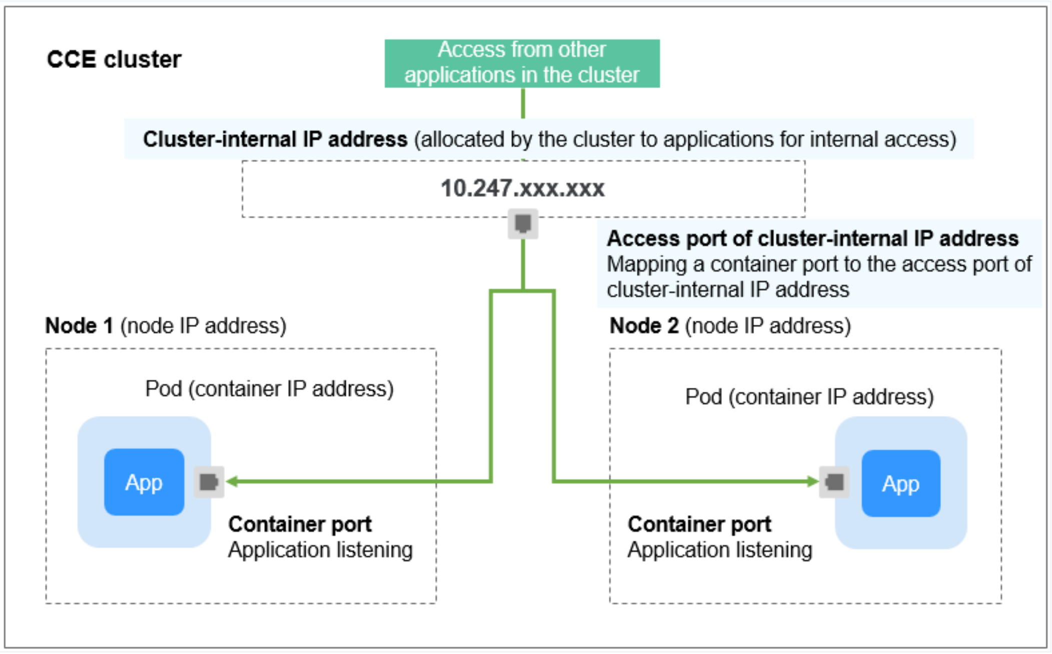
10.2 Intra-Cluster Access (ClusterIP)

A workload is accessible to other workloads in the same cluster through the use of an internal domain name.

The cluster-internal domain name format is *<User-defined Service name>*.*<Namespace of the workload>***.svc.cluster.local:<*Port number*>**, for example, **nginx.default.svc.cluster.local:80**.

Figure 10-3 shows the mapping relationships between access channels, container ports, and access ports.

Figure 10-3 Intra-cluster access (ClusterIP)


CCE cluster
Access from other applications in the cluster
Cluster-internal IP address (allocated by the cluster to applications for internal access)
10.247.xxx.xxx
Access port of cluster-internal IP address
Mapping a container port to the access port of cluster-internal IP address
Node 1 (node IP address)
Node 2 (node IP address)
Pod (container IP address)
App
Container port
Application listening
Pod (container IP address)
App
Container port
Application listening

Using the Console

You can set the access type (Service) when creating a workload on the CCE console.

Step 1 In the **Set Access Mode** step of **Creating a Deployment** or **Creating a StatefulSet**, click **Add Service**, and set the following parameters:

- **Access Type**: Select **ClusterIP**.
- **Service Name**: Specify a Service name, which can be the same as the workload name.
- **Port Settings**
  - **Protocol**: a protocol used by the service.
  - **Container Port**: a port on which the workload listens. The nginx application listens on port 80.
  - **Access Port**: a port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

Step 2 Click **Next: Configure Advanced Settings**. On the page displayed, click **Create**.

Step 3 Click **View Workload Details**. On the **Services** tab page, obtain the access address, for example, 10.247.74.100:2.

Step 4 Log in to any node in the cluster where the workload is located.

Step 5 Run the **curl** command to verify whether access to the workload is successful. You can perform the verification by using the IP address or domain name.

- By using the IP address

  curl *10.247.74.100:2*

  10.247.74.100:2 is the access address obtained in **Step 3**.

If information similar to the following is displayed, the workload is accessible.

```
<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>
```

- By using the domain name

  curl *nginx.default.svc.cluster.local:2*

  nginx.default.svc.cluster.local is the domain name obtained in **Step 3**.

  If information similar to the following is displayed, the workload is accessible.

```
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

Adding a Service Using kubectl

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

Prerequisites

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

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: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        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: 2
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

Table 10-1 Key parameters

Parameter	Type	Description
port	Integer	Port at which the Service is exposed, that is, the access port on the CCE console.
targetPort	String	Container port on the CCE console.

Parameter	Type	Description
type	String	Access type on the CCE console. The options are as follows: ● ClusterIP ● NodePort ● LoadBalancer The default value is **ClusterIP**, indicating the cluster-internal IP address.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                    READY     STATUS             RESTARTS   AGE
etcd-0                  0/1       ImagePullBackOff   0          27m
icagent-m9dkt           0/0       Running            0          3d
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.

```
NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
etcd-svc          ClusterIP   None             <none>        3120/TCP   30m
kubernetes        ClusterIP   10.247.0.1       <none>        443/TCP    3d
nginx-clusterip   ClusterIP   10.247.200.134   <none>        80/TCP     20s
```

Step 5 Log in to any node in the cluster where the workload is located.

Step 6 Run the **curl** command to check whether the workload is accessible. You can perform the verification by using the IP address or domain name.

- By using the IP address

  curl *10.247.200.134:2*

  If information similar to the following is displayed, the workload is accessible.

```
<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>
```

- By using the domain name

  curl *nginx*-**clusterip.default.svc.cluster.local:***2*

  If information similar to the following is displayed, the workload is accessible.

```
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

Setting the Access Type After Creating a Workload on the Console

You can set the Service after creating a workload. This has no impact on the workload status and takes effect immediately. Perform the following operations to check the available space:

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**. On the workload list, click the name of the workload for which you will create a Service.

Step 2 On the **Services** tab page, click **Create Service**.

Step 3 On the **Create Service** page, select **ClusterIP** from the **Access Type** drop-down list.

Step 4 Set ClusterIP parameters.

- **Service Name**: Service name, which can be the same as the workload name.
- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload**: workload for which you want to add a Service. The value is inherited from the workload creation page and cannot be changed.
- **Port Settings**
  - **Protocol**: protocol used by the Service.
  - **Container Port**: port on which the workload listens. The nginx application listens on port 80.
  - **Access Port**: a port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

Step 5 Click **Create**. The ClusterIP Service will be added for the workload.

----End

Updating a Service

After adding a Service, you can update the port configuration of the Service. The procedure is as follows:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Services** tab page, click **Update** for the Service to be updated.

Step 2 On the **Update Service** page, select **ClusterIP** from the **Access Type** drop-down list.

Step 3 Update intra-cluster access parameters.

- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload**: workload for which you want to add a Service. The value is inherited from the workload creation page and cannot be changed.
- **Port Settings**
  - **Protocol**: protocol used by the Service.
  - **Container Port**: port on which the workload listens. The nginx application listens on port 80.
  - **Access Port**: a port mapped to the container port at the cluster-internal IP address. The workload can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

Step 4 Click **Update**. The Service will be updated for the workload.

----End

10.3 NodePort

A service is exposed on each node's IP address at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. By requesting the <NodeIP>:<NodePort>, you can access a NodePort service from the outside of the cluster.

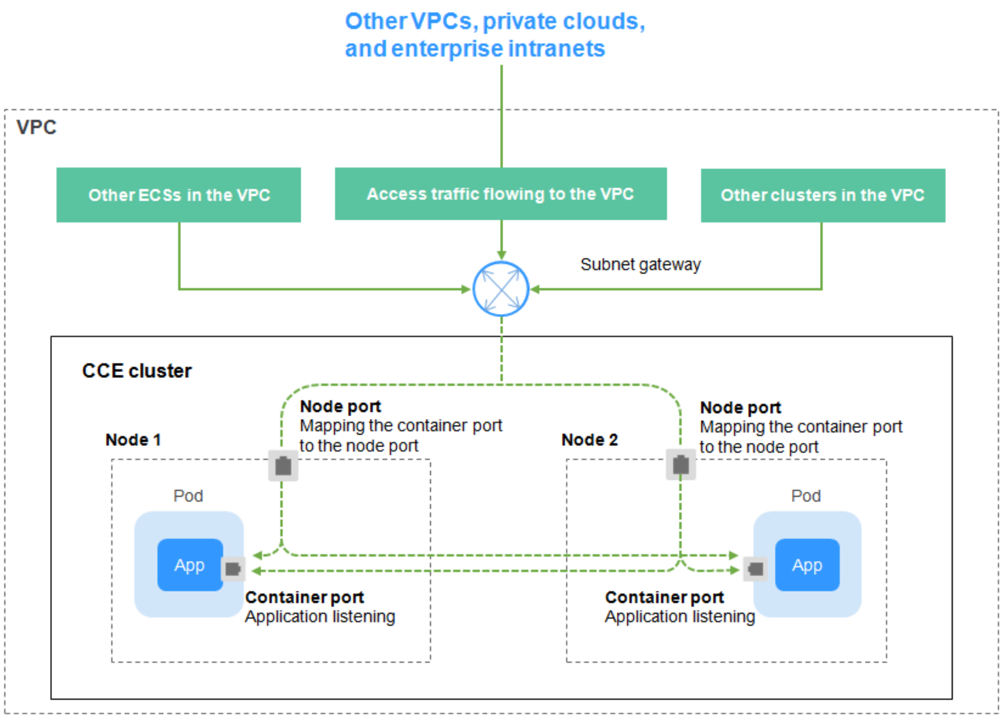
NodePort access is further classified into intra-VPC access and EIP-based access.

- **Intra-VPC Access**

  A workload can be accessed from other workloads in the same VPC through a node IP address.

  Typical scenario: Workloads in a Kubernetes cluster need to be accessed by other workloads in the same VPC.

  Figure 10-4 Intra-VPC access by using a ClusterIP service address

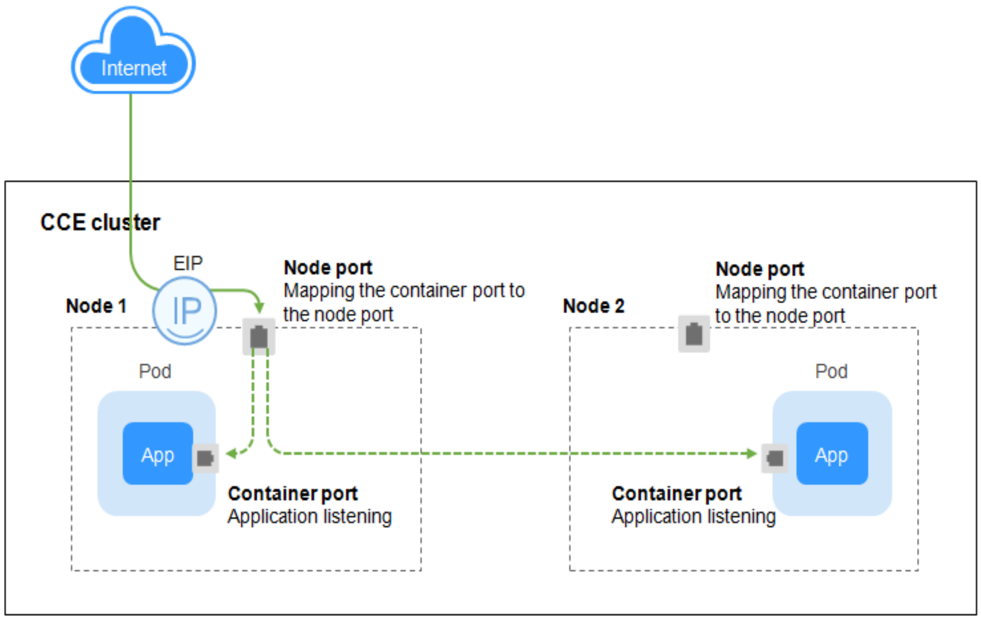

Other VPCs, private clouds, and enterprise intranets
VPC
Other ECSs in the VPC
Access traffic flowing to the VPC
Other clusters in the VPC
Subnet gateway
CCE cluster
Node 1
Node port
Mapping the container port to the node port
Node 2
Node port
Mapping the container port to the node port
Pod
App
Container port
Application listening
Pod
App
Container port
Application listening

- **EIP-based Access**

  If the NodePort Service needs to be exposed to public networks, the workload that the NodePort Service targets can be accessed at an EIP from public networks.

  This can happen only after an EIP is bound to any node in the cluster and a service port is mapped to a node port in the 30000–32767 range. For example, the access address could be 10.117.117.117:30000.

Figure 10-5 EIP-based access



Using the Console

You can set the access type when creating a workload on the CCE console. An Nginx workload is used as an example.

Step 1 In the **Set Application Access** step of **Creating a Deployment** or **Creating a StatefulSet**, click **Add Service** and set the following parameters:

- **Access Type**: In this example, **NodePort** is selected.
- **Service Name**: can be the same as the workload name.
- **Service Affinity:**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the load balancer used by the service is located, without masking clients' source IP addresses.
- **Port Settings**
  - **Protocol**: protocol to be used by the Service.
  - **Container Port**: port on which the workload listens in the container image. The value ranges from 1 to 65535.
  - **Access Port**: node port (with a private IP address) to which the container port will be mapped. You are advised to select **Auto generated**.
    - **Automatically Generated**: The system automatically assigns a port number.
    - **Specified Port**: You have to manually specify a fixed node port number in the range of 30000–32767. Ensure that the port is unique in a cluster.

Step 2 On the **Add Service** page, click **OK**. On the workload creation page, click **Next**. On the **Configure Advanced Settings** page, click **Create**.

Step 3 Click **View Workload Details**. On the **Services** tab page, obtain the access address, for example, 192.168.0.160:30358.

----End

Setting the Intra-VPC Access Type Using kubectl

You can run kubectl commands to set the access type. This section uses an Nginx workload as an example to describe how to set intra-VPC access using kubectl.

Prerequisites

You have configured the kubectl and connected an to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which the kubectl has been configured.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-nodeport-svc.yaml** files.

The files names are user-defined. nginx-deployment.yaml and nginx-nodeport-svc.yaml are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-nodeport-svc.yaml

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

```
  app: nginx
type: NodePort
```

Table 10-2 Key parameters

Parameter	Type	Description
nodePort	Integer	Access port set on the console. The value ranges from 30000 to 32767. If this parameter is left unspecified, the value is automatically generated.
port	Integer	Access port mapped to the cluster-internal IP address. The value ranges from 1 to 65535.
protocol	String	IP protocol used by the port. The value can be **TCP** or **UDP**.
targetPort	String	Container port set on the console. The value ranges from 1 to 65535.
type	String	Access type set on the console. NodePort indicates the private IP address of a node.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                         READY     STATUS             RESTARTS   AGE
etcd-0                       0/1       ImagePullBackOff   0          48m
icagent-m9dkt                0/0       Running            0          3d
nginx-2601814895-qhxqv       1/1       Running            0          9s
```

Step 4 Create a Service.

kubectl create -f nginx-nodeport-svc.yaml

If information similar to the following is displayed, the Service is being created.

```
service "nginx-nodeport" created
```

kubectl get svc

If information similar to the following is displayed, the Service has been created.

```
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
etcd-svc         ClusterIP   None           <none>        3120/TCP       49m
kubernetes       ClusterIP   10.247.0.1     <none>        443/TCP        3d
nginx-nodeport   NodePort    10.247.4.225   <none>        80:30000/TCP   7s
```

Step 5 Run the **curl** command to verify whether the workload is accessible.

curl *192.168.2.240:30000*

192.168.2.240 is the IP address of any node in the cluster, and **30000** is the port opened on all nodes in the cluster.

If information similar to the following is displayed, the workload is accessible.

```
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

----End

Setting the EIP-based Access Type Using kubectl

This section uses an Nginx workload as an example to describe how to implement EIP-based access using kubectl.

Prerequisites

You have configured the kubectl and connected an to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-eip-svc.yaml** files.

The file names are user-defined. nginx-deployment.yaml and nginx-eip-svc.yaml are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-eip-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx-eip
spec:
  ports:
  - name: service0
    nodePort: 30000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
```

Table 10-3 Key parameters

Parameter	Type	Description
nodePort	Integer	Access port set on the console. The value ranges from 30000 to 32767. If this parameter is left unspecified, the value is automatically generated.
port	Integer	Access port mapped to the cluster-internal IP address. The value ranges from 1 to 65535.
protocol	String	IP protocol used by the port. The value can be **TCP** or **UDP**.
targetPort	String	Container port set on the console. The value ranges from 1 to 65535.
type	String	Access type set on the console. EIP-based access is based on a NodePort Service.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                    READY     STATUS             RESTARTS   AGE
etcd-0                 0/1       ImagePullBackOff   0          59m
```

```
icagent-m9dkt          0/0       Running          0         3d
nginx-2601814895-sf71t   1/1       Running          0         8s
```

Step 4 Create a service.

kubectl create -f nginx-eip-svc.yaml

If information similar to the following is displayed, the service has been created.

```
service "nginx-eip" created
```

kubectl get svc

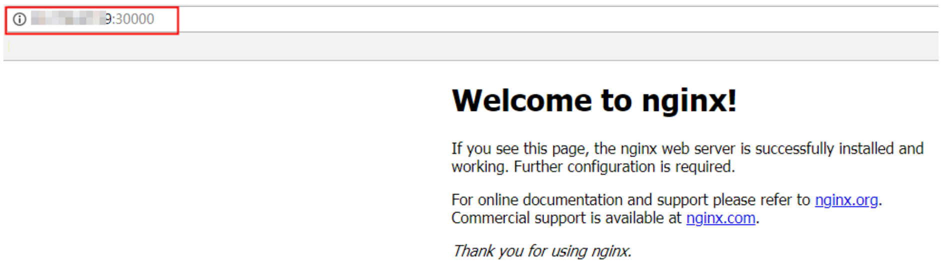
If information similar to the following is displayed, the service access type has been set successfully.

```
NAME        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
etcd-svc     ClusterIP   None             <none>        3120/TCP       59m
kubernetes   ClusterIP   10.247.0.1        <none>        443/TCP        3d
nginx-eip    NodePort    10.247.120.135   <none>        80:30000/TCP   7s
```

Step 5 In the address bar of your browser, enter the access address (for example, 10.78.44.60:30000) and press **Enter**.

10.78.44.60 is the EIP, and **30000** is the node port number obtained in the previous step.

Figure 10-6 Accessing Nginx through an EIP


[redacted]:30000

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

Methods of Verifying the Access Type

- **Verifying Intra-VPC Access**

Step 1 On the homepage of the management console, choose **Computing** > **Elastic Cloud Server**.

Step 2 Select any ECS in the same VPC as the workload that will be accessed, and confirm that the security group is open to the IP address and port to be connected.

Step 3 Click **Remote Login**. On the login page, enter the username and password.

Step 4 Run the **curl** command to check whether access to the workload is successful.

NOTE

An intra-VPC access service will be assigned a cluster-internal IP address. You can use <Service's cluster-internal IP address>:<Access port> to verify whether the workload is reachable from inside the cluster. By default, <Access port> in <Service's cluster-internal IP address>:<Access port> is the same as the container port (for example, 80).

curl *192.168.0.160*:30358

192.168.0.160:30358 is the access address obtained in **Step 3**.

If information similar to the following is displayed, the workload is accessible.

```
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

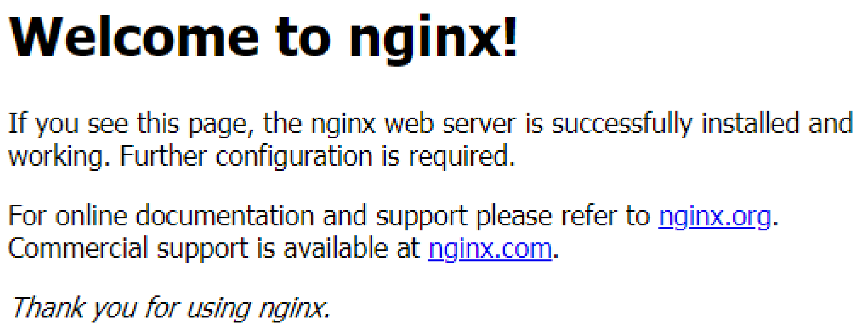
----End

- **Verifying EIP-based Access**

Step 1 After the workload is successfully created, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** on the CCE console. Click the name of the workload you will verify to show the details of the workload. On the workload details page, click the **Services** tab and obtain the access address, for example, 10.78.27.59:30911.

Step 2 Click the access address.

Figure 10-7 Accessing Nginx through an EIP



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

Setting the Access Type After Creating a Workload on the Console

You can set the Service after creating a workload. This has no impact on the workload status and takes effect immediately. Perform the following operations to check the available space:

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**. On the workload list, click the name of the workload for which you will create a Service.

NOTE

If the Service is associated with an ingress, the ingress is unavailable after the port information of the Service is updated. In this case, you need to delete and recreate the Service.

Step 2 On the **Services** tab page, click **Create Service**.

Step 3 On the **Create Service** page, select **NodePort** from the **Access Type** drop-down list.

Step 4 Set node access parameters.

- **Service Name**: can be the same as the workload name.
- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload**: workload for which you want to add a Service.
- **Service Affinity:**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the load balancer used by the service is located, without masking clients' source IP addresses.
- **Port Settings**
  - **Protocol**: Select the protocol used by the Service.
  - **Container Port**: a port on which the workload in the container image listens. The Nginx application listens on port 80.
  - **Access Port**: node port (with a private IP address) to which the container port will be mapped. You are advised to select **Auto generated**.
    - **Auto generated**: The system automatically assigns a port number.
    - **Specified Port**: You have to manually specify a fixed node port number in the range of 30000–32767. Ensure that the port is unique in a cluster.

Step 5 Click **Create**. A NodePort Service will be added for the workload.

----End

Updating a Service

After adding a Service, you can update the port configuration of the Service. The procedure is as follows:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Services** tab page, click **Update** for the Service to be updated.

Step 2 On the **Update Service** page, select **NodePort** from the **Access Type** drop-down list.

Step 3 Update NodePort Service parameters.

- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload**: workload for which you want to add a Service.
- **Service Affinity:**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the load balancer used by the service is located, without masking clients' source IP addresses.
- **Port Settings**
  - **Protocol**: protocol to be used by the Service.
  - **Container Port**: port on which the workload in the container image listens. The Nginx application listens on port 80.
  - **Access Port**: node port (with a private IP address) to which the container port will be mapped. You are advised to select **Auto generated**.
    - **Auto generated**: The system automatically assigns a port number.
    - **Specified Port**: You have to manually specify a fixed node port number in the range of 30000–32767. Ensure that the port is unique in a cluster.

Step 4 Click **Update**. The Service will be updated for the workload.

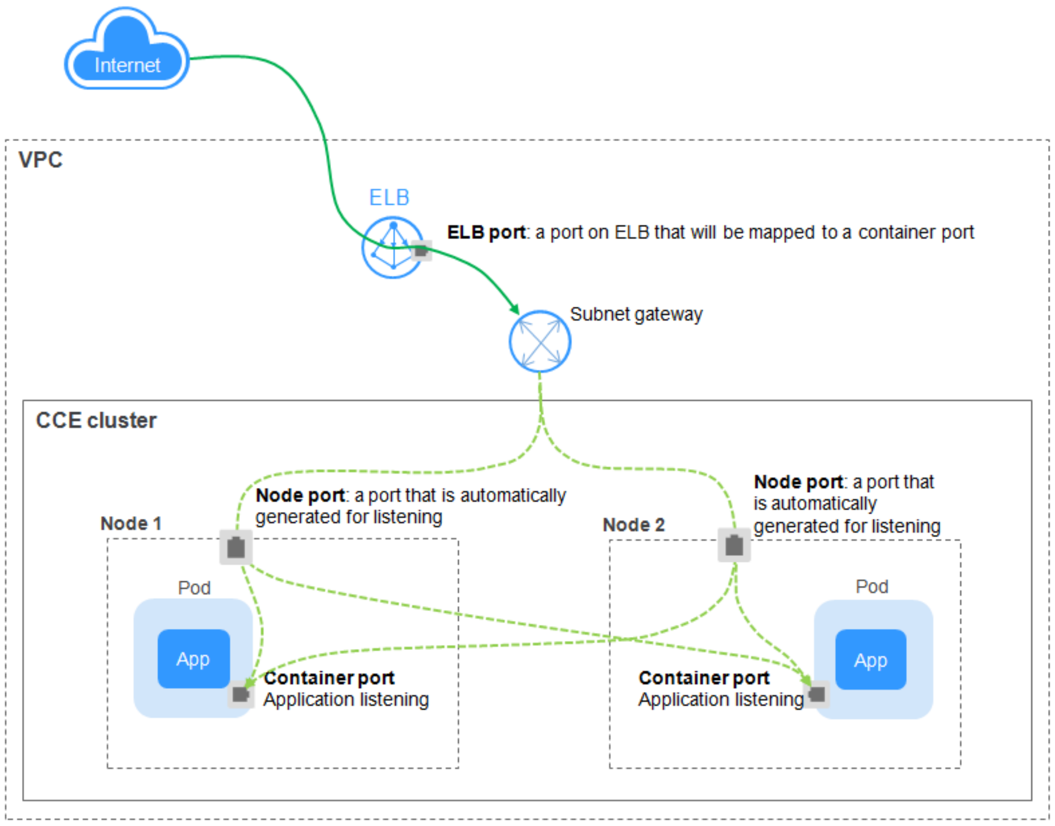
----End

10.4 LoadBalancer

A workload can be accessed from public networks through a load balancer. LoadBalancer provides higher reliability than EIP-based NodePort because an EIP is no longer bound to a single node. The LoadBalancer access type is applicable to the scenario in which a service exposed to public networks is required.

The access address is in the format of <IP address of public network load balancer>:<access port>. For example, 10.117.117.117:80.

Figure 10-8 LoadBalancer


Internet
VPC
ELB
ELB port: a port on ELB that will be mapped to a container port
Subnet gateway
CCE cluster
Node 1
Node port: a port that is automatically generated for listening
Node 2
Node port: a port that is automatically generated for listening
Pod
App
Container port
Application listening
Container port
Application listening
Pod
App

NOTE

The LoadBalancer Service allows workloads to be accessed from public networks through ELB. This access type has the following restrictions:

- It is recommended that automatically created load balancers not be used by other resources. Otherwise, these load balancers cannot be completely deleted, causing residual resources.
- Do not change the listener name for the load balancer in use. Otherwise, the load balancer cannot be accessed.

Using the Console

You can set the Service when creating a workload on the CCE console. An Nginx workload is used as an example.

Step 1 In the **Set Application Access** step of **Creating a Deployment** or **Creating a StatefulSet**, click **Add Service** and set the following parameters:

- **Access Type**: Select **LoadBalancer (ELB)**.
- **Service Name**: can be the same as the workload name.
- **Service Affinity**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the workload targeted by the service is located, without masking clients' source IP addresses.

- **Elastic Load Balancer**: A load balancer automatically distributes Internet access traffic to multiple nodes running the workload.
  - **Public network**: You can select an existing public network load balancer or have the system automatically create a new public network load balancer.

    If you have the system automatically create a public network load balancer, you can click **Change Configuration** to modify its name, EIP type, and bandwidth.
  - **Private network**: You can select an existing private network load balancer or have the system automatically create a new private network load balancer.

  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).

  Other configurations
  - **Specifications**: This field is displayed only when you select **Public network** and **Automatic creation** for **Elastic Load Balancer**. You can click **Change configuration** to modify the name, specifications, and bandwidth of the load balancer.
  - **Health Check**: This option is enabled by default. Configure health check parameters as prompted.
- **Port Settings**
  - **Protocol**: protocol to be used by the Service.
  - **Container Port**: port defined in the container image and on which the workload listens. The Nginx workload listens on port 80.
  - **Access Port**: a port to which the container port will be mapped when the load balancer IP address is used for accessing the workload. The port number range is 1–65535.

Step 2 After the configuration is complete, click **OK**.

Step 3 On the workload creation page, click **Next: Configure Advanced Settings**. On the **Configure Advanced Settings** page, click **Create**.

Step 4 After the workload is successfully created, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets** on the CCE console. Click the name of the workload to show more details of the workload. On the workload details page, click the **Services** tab and obtain the access address.

Step 5 Click the access address.

----End

Using kubectl

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

Prerequisites

You have configured the kubectl and connected an ECS to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to any node in the cluster where the workload is located.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-elb-svc.yaml

NOTE

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

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes. For details, see **Scheduling Policy Overview**.

- Automatically creating load balancer

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/session-affinity-mode: SOURCE_IP
    kubernetes.io/elb.subnet-id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0
    kubernetes.io/elb.autocreate: "{\"type\":\"public\",\"bandwidth_name\":\"cce-
bandwidth-1551163379627\",\"bandwidth_chargemode\":\"traffic\",\"bandwidth_size\":
5,\"bandwidth_sharetype\":\"PER\",\"eip_type\":\"5_bgp\",\"name\":\"james\"}"
  labels:
    app: nginx
  name: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Using existing load balancer

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/session-affinity-mode: SOURCE_IP
    kubernetes.io/elb.id: 3c7caa5a-a641-4bff-801a-feace27424b6
    kubernetes.io/elb.subnet-id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0
  labels:
    app: nginx
  name: nginx
spec:
  loadBalancerIP: 10.78.42.242
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Table 10-4 Key parameters

Parameter	Type	Description
kubernetes.io/elb.class	String	Mandatory and must be set to **union** if an enhanced load balancer is in use.
kubernetes.io/session-affinity-mode	String	Optional. If session stickiness is enabled, add this parameter. The value **SOURCE_IP** indicates that listeners ensure session stickiness based on source IP addresses.
kubernetes.io/elb.session-affinity-option	**elb.session-affinity-option** object	Optional. This parameter indicates the configuration items for the ELB sticky session.
kubernetes.io/elb.id	String	Optional. This parameter is mandatory if an existing load balancer is used. It indicates the ID of an enhanced load balancer.
kubernetes.io/elb.subnet-id	String	Optional. This parameter is mandatory only if a load balancer will be automatically created.

Parameter	Type	Description
kubernetes.io/ elb.enterpriseID	String	Optional. This parameter is mandatory if a public/private network load balancer will be automatically created. This parameter indicates the name of the ELB enterprise project in which the ELB will be created. The value is a string of 1 to 100 characters.
kubernetes.io/elb.lb-algorithm	String	Optional. This parameter indicates the algorithm used the ELB. Default value: **ROUND_ROBIN** Value options: ROUND_ROBIN, LEAST_CONNECTIONS, SOURCE_IP, or left blank
kubernetes.io/elb.health-check-flag	String	Optional. This parameter indicates that whether the ELB health check function is enabled. The default value is **on**. Value options: on, off, or left blank
kubernetes.io/elb.health-check-option	**elb.health-check-option** object	Optional. This parameter indicates the ELB health check configuration items.

Parameter	Type	Description
kubernetes.io/elb.autocreate	**elb.autocreate** object	Optional. This parameter is mandatory if a public network load balancer will be automatically created. The system will create an enhanced load balancer and an EIP. This parameter is also mandatory if a private network load balancer will be automatically created. The system will create an enhanced load balancer. **Example:** ● Value for a public network load balancer that is automatically created: "{\"type\":\"public\",\"bandwidth_name\":\"cce-bandwidth-1551163379627\",\"bandwidth_chargemode\":\"traffic\",\"bandwidth_size\":5,\"bandwidth_sharetype\":\"PER\",\"eip_type\":\"5_bgp\",\"name\":\"james\"}" ● Value for a private network load balancer that is automatically created: "{\"type\":\"inner\"}"
loadBalancerIP	String	Private IP address of a private network load balancer or public IP address of a public network load balancer.
externalTrafficPolicy	String	Optional. If session stickiness is enabled, add this parameter so requests are transferred to a fixed node. If a LoadBalancer Service with this parameter set to **Local** is created for a workload, the workload can be accessed only when the client is installed on the same node as the server.
port	Integer	Access port that is registered on the load balancer and mapped to the cluster-internal IP address.
targetPort	String	Container port on the CCE console.

Table 10-5 elb.autocreate parameters

Parameter	Type	Description
name	String	Name of the load balancer that is automatically created. The value is a string of 1 to 64 characters that consist of letters, digits, underscores (_), and hyphens (-).
type	String	Network type of the load balancer. ● **public**: public network load balancer. ● **inner**: private network load balancer.
bandwidth_name	String	Bandwidth name. The default value is **cce-bandwidth-********. The value is a string of 1 to 64 characters that consist of letters, digits, underscores (_), hyphens (-), and periods (.).
bandwidth_chargemode	String	Bandwidth billing mode. The value is **traffic**, indicating that the billing is based on traffic.
bandwidth_size	Integer	Bandwidth size. Set this parameter based on the bandwidth range supported by the region.
bandwidth_sharetype	String	Bandwidth sharing mode. ● **PER**: dedicated bandwidth. ● **WHOLE**: shared bandwidth.
eip_type	String	EIP type. Set this parameter based on the EIP types supported by ELB.

Table 10-6 elb.session-affinity-option parameters

Parameter	Type	Description
persistence_timeout	String	Sticky session timeout, in seconds. This parameter is valid only when **elb.session-affinity-mode** is set to **SOURCE_IP**. Default value: **60** Value range: 1 to 60

Table 10-7 elb.health-check-option parameters

Parameter	Type	Description
delay	String	Optional. Initial wait time (in seconds) for starting the health check. Default value: **5** Value range: 1 to 50
timeout	String	Optional. Health check timeout, in seconds. Default value: **10** Value range: 1 to 50
max_retries	String	Optional. Maximum number of health check retries. Default value: **3** Value range: 1 to 10
protocol	String	Optional. Protocol used for health check. Default value: protocol of the associated service Value options: TCP, UDP_CONNECT or HTTP
path	String	Optional. Health check URL. This parameter is optional when **protocol** is **HTTP**. Default value: **/** Value range: 1 to 10000

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created.

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                     READY     STATUS             RESTARTS   AGE
etcd-0                   0/1       ImagePullBackOff   0          1h
icagent-m9dkt            0/0       Running            0          3d
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 is being created:

```
service "nginx" created
```

kubectl get svc

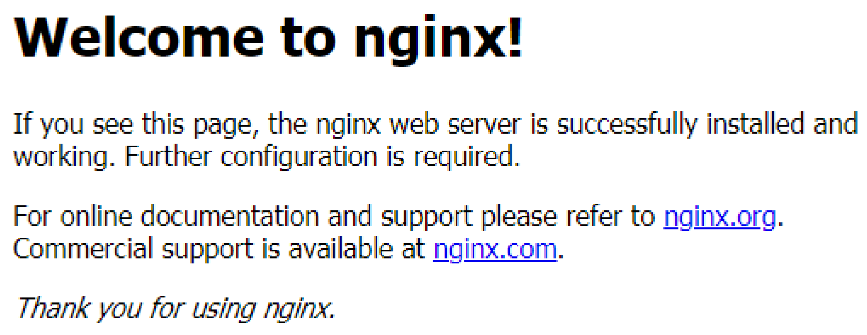
If information similar to the following is displayed, the Service has been set successfully, and the workload is accessible.

```
NAME        TYPE          CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
etcd-svc    ClusterIP     None             <none>        3120/TCP       1h
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, **10.78.42.242:31540**. **10.78.42.242** indicates the IP address of the load balancer, and **31540** indicates the access port displayed on the CCE console.

The Nginx is accessible.

Figure 10-9 Accessing Nginx through load balancing (2)



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

Setting the Access Type After Creating a Workload

You can set the access type after creating a workload. This has no impact on the workload status and takes effect immediately. The procedure is as follows:

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets**. On the workload list, click the name of the workload for which you will add a service.

Step 2 On the **Services** tab page, click **Create Service**.

Step 3 On the **Create Service** page, set **Access Type** to **LoadBalancer (ELB)**.

Step 4 Configure load balancing parameters.

- **Service Name**: can be the same as the workload name.
- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.
- **Workload**: workload for which you want to add a Service. The value is inherited from the workload creation page and cannot be changed.
- **Service Affinity**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.

  - **Node level**: External traffic is routed to the node where the workload targeted by the service is located, without masking clients' source IP addresses.
- **Elastic Load Balancer**: A load balancer automatically distributes Internet access traffic to multiple nodes running the workload.
  - **Public network**: You can select an existing public network load balancer or have the system automatically create a new public network load balancer.

    If you have the system automatically create a public network load balancer, you can click **Change Configuration** to modify its name, EIP type, billing mode, and bandwidth.
  - **Private network**: You can select an existing private network load balancer or have the system automatically create a new private network load balancer.

  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).

  Other configurations
  - **Specifications**: This field is displayed only when you select **Public network** and **Automatic creation** for **Elastic Load Balancer**. You can click **Change configuration** to modify the name, specifications, and bandwidth of the load balancer.
  - **Health Check**: This option is enabled by default. Configure health check parameters as prompted.
- **Port Settings**
  - **Protocol**: protocol to be used by the Service.
  - **Container Port**: port on which the workload in the container image listens. The Nginx workload listens on port 80.
  - **Access Port**: a port mapped to the container port at the load balancer's IP address. The workload can be accessed at <load balancer's IP address>:<access port>. The port number range is 1–65535.

Step 5 Click **Create**. A LoadBalancer Service will be added for the workload.

----End

Updating a Service

After adding a Service, you can update the port configuration of the Service. The procedure is as follows:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Services** tab page, filter services by cluster and namespace, and click **Update** for the service to be updated.

Step 2 On the **Update Service** page, set **Access Type** to **LoadBalancer (ELB)**.

Step 3 Update load balancing parameters.

- **Cluster Name**: name of the cluster where the workload runs. The value is inherited from the workload creation page and cannot be changed.
- **Namespace**: namespace where the workload is located. The value is inherited from the workload creation page and cannot be changed.

- **Workload**: workload for which you want to add a Service. The value is inherited from the workload creation page and cannot be changed.
- **Service Affinity**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the workload targeted by the service is located, without masking clients' source IP addresses.
- **Elastic Load Balancer**: A load balancer automatically distributes Internet access traffic to multiple nodes running the workload.
  - Public network: Select an existing load balancer.
  - Private network: Select an existing load balancer.

  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).

  Other configurations
  - **Sticky Session**: Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be forwarded to the same backend server.
  - **Health Check**: This option is enabled by default. Configure health check parameters as prompted.
- **Port Settings**
  - **Protocol**: protocol to be used by the Service.
  - **Container Port**: port on which the workload in the container image listens. The Nginx workload listens on port 80.
  - **Access Port**: a port mapped to the container port at the load balancer's IP address. The workload can be accessed at <Load balancer's IP address>:<Access port>. The port number range is 1–65535.

Step 4 Click **Update**. The Service will be updated for the workload.

----End

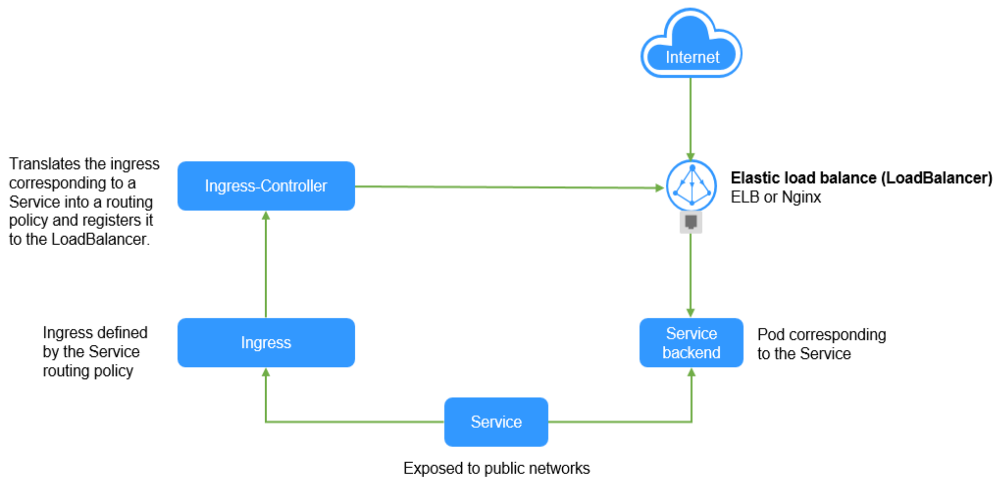
10.5 Layer-7 Load Balancing (Ingress)

Layer-7 load balancing uses enhanced load balancers. Compared with layer-4 load balancing, layer-7 load balancing additionally supports Uniform Resource Identifier (URI) configurations and distributes access traffic to services based on URIs. In addition, different functions are implemented based on various URIs.

The access address is in the format of <IP address of the load balancer>:<Access port>/Defined URI, for example, **10.117.117.117:80/helloworld**.

You can configure load balancers of public and private networks to implement layer-7 route forwarding on public networks and private networks (intra-VPC networks).

Figure 10-10 Layer-7 load balancing (ingress)


Internet
Translates the ingress corresponding to a Service into a routing policy and registers it to the LoadBalancer.
Ingress-Controller
Elastic load balance (LoadBalancer)
ELB or Nginx
Ingress defined by the Service routing policy
Ingress
Service backend
Pod corresponding to the Service
Service
Exposed to public networks

Preparation

An ELB instance has been created on the management console.

Step 1 Log in to the management console and choose **Network** > **Elastic Load Balance** from the service list.

Step 2 Click **Buy Enhanced Load Balancer** in the upper right corner.

NOTE

The LoadBalancer access type allows workloads to be accessed from public networks through **ELB**. This access type has the following restrictions:

- It is recommended that automatically created load balancers not be used by other resources. Otherwise, these load balancers cannot be completely deleted, causing residual resources.
- Do not change the listener name for the load balancer in use. Otherwise, the load balancer cannot be accessed.
- Do not add custom forwarding rules for the listener of the load balancer that is being used. As these forwarding rules are not added to the ingress for management on the **Network** page, they will be deleted when the ingress is updated.

----End

Using the Console

You can set the Service when creating a workload on the CCE console. The ingress-test workload is used as an example.

Step 1 Create a workload. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

- If you have set the access type NodePort during workload creation, go to **Step 3**.
- If the access type is not set during workload creation, go to **Step 2**.

Step 2 (Optional) Set the access type NodePort.

1. Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**.
2. On the **Services** tab page, click **Create Service**. In the **Select Type** dialog box, select **NodePort**.
   - **Service Name**: Specify a Service name, which can be the same as the workload name.
   - **Cluster Name**: Select the cluster for which you want to add a Service.
   - **Namespace**: Select a namespace for which you want to add a Service.
   - **Workload**: Click **Select Workload**, select the name of the workload for which the NodePort Service is to be configured, and click **OK**.
   - **Service Affinity**
     - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
     - **Node level**: External traffic is routed to the node where the workload targeted by the service is located, without masking clients' source IP addresses.
   - **Port Settings**
     - **Protocol**: Select a protocol used by the service.
     - **Container Port**: a port on which the workload in the container image listens. The Nginx workload listens on port 80.
     - **Access Port**: Specify a port to which the container port will be mapped when the node's private IP address is used for accessing the workload. The port number range is 30000–32767. You are advised to select **Automatically generated**.
       - **Automatically Generated**: The system automatically assigns a port number.
       - **Specified Port**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in its cluster.
3. Click **Create**. The access type NodePort is successfully set.

Step 3 Add an ingress service.

1. In the navigation pane, choose **Resource Management** > **Network**.
2. On the **Ingresses** tab page, click **Create Ingress**.
   - **Ingress Name**: Specifies the name of an ingress, for example, **ingress-demo**.
   - **Cluster Name**: Select the cluster to which the ingress is to be added.
   - **Namespace**: Select the namespace to which the ingress is to be added.
   - **Elastic Load Balancer**: A load balancer automatically distributes Internet access traffic to multiple nodes running the workload. 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).
     - **Public network**: You can select an existing public network load balancer or have the system automatically create a new public network load balancer.

If you have the system automatically create a public network load balancer, you can click **Change Configuration** to modify its name, EIP type, billing mode, and bandwidth.

- **Private network**: You can select an existing private network load balancer or have the system automatically create a new private network load balancer.

– **External Port**: port number exposed in the LoadBalancer Service address. The port number can be specified randomly.

– **Front-End Protocol**: HTTP and HTTPS are supported. If you select HTTPS, choose a key certificate..

NOTE

- The key certificate **ingress-test-secret.yaml** is required only when HTTPS is selected. For details on how to create a key, see **Creating a secret**.
- 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 earliest certificate takes effect on the load balancer.

– Domain name: domain name that is actually accessed. Ensure that the entered domain name has been registered. After the ingress is created, bind the domain name to the IP address of the automatically created load balancer, that is, the IP address of the ingress. Once a domain name rule is configured, the domain name must be used for access.

– **Route Configuration**

- Route Matching: Prefix matching, exact matching, and regular matching.
  - 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: Only the URL that is the same as the specified URL can be accessed. For example, if the mapping URL is /healthz, only /healthz can be accessed.
  - Regular expression: The URL rule can be set, for example, **/[A-Za-z0-9_.-]+/test**. All URLs that comply with this rule can be accessed, for example, **/abcA9/test** and **/v1-Ab/test**. Two regular expression standards are supported: POSIX and Perl.
- **URL**: access path to be registered, for example, **/healthz**.
- **Service Name**: Select the Service whose ingress is to be added. The Service type is intra-VPC access.
- **Service Port**: port on which the container in the container image listens. For example, the defaultbackend application listens on port 8080 (container port).

Step 4 Click **Create**.

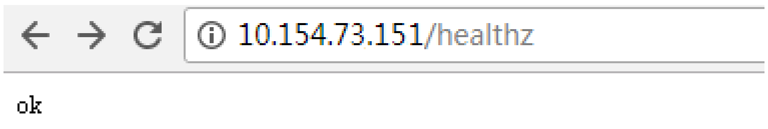
After the creation is complete, you can view the created ingress in the ingress list.

Step 5 Access the /healthz interface of the workload (for example, defaultbackend).

Method 1: By using a load balancer's IP address (The domain name cannot be configured for LoadBalancer services.)

1. Obtain the access address of the /healthz interface of defaultbackend. The access address is in the format of <Load balancer's IP address>:<External port><Mapping URL>. For example, 10.154.73.151:80/healthz.
2. Enter the URL of the /healthz interface, for example, http://10.154.73.151:80/healthz, in the address box of the browser to access the workload.

Figure 10-11 Accessing the /healthz interface of defaultbackend


10.154.73.151/healthz

ok

Method 2: By using a domain name

The following uses the domain name ingress.com configured in the ingress as an example.

1. Obtain the domain name and access address (IP address and port number) of the /iamtest interface of the ingress-demo.
2. Configure the IP address in the access address and the domain name in the **C:\Windows\System32\drivers\etc\hosts** file on the local host.
3. Enter **http://<*Domain name*>:<*Access port*>/<*Mapping URL*>** in the address box of the browser. For example, http://ingress.com:81/iamtest.

----End

Using kubectl

This section uses an Nginx workload as an example to describe how to implement ingress access using kubectl.

Prerequisites

You have configured kubectl and connected an to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create the ingress-test-deployment.yaml, ingress-test-svc.yaml, ingress-test-ingress.yaml, and ingress-test-secret.yaml files.

ingress-test-deployment.yaml, ingress-test-svc.yaml, ingress-test-ingress.yaml, and ingress-test-secret.yaml are user-defined names. You can name them randomly.

NOTE

- The key certificate ingress-test-secret.yaml is required only when HTTPS is selected. For details on how to create a key, see **Creating a secret**.
- 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.

vi ingress-test-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: ingress-test-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ingress-test-deployment
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: ingress-test-deployment
    spec:
      containers:
        # Third-party public image. You can obtain the address by referring to the description or use your own
image.
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi ingress-test-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: ingress-test-svc
  name: ingress-test-svc
spec:
  ports:
  - name: service0
    port: 8888
    protocol: TCP
    targetPort: 8888       #If multiple ports need to be set, fill in the following information in sequence:
  - name: service1
    port: 8081
    protocol: TCP
    targetPort: 8081
  selector:
    app: ingress-test-deployment
  type:  NodePort
```

Table 10-8 Key parameters

Parameter	Type	Description
port	Integer	Access port mapped to the cluster-internal IP address. The value ranges from 1 to 65535.

Parameter	Type	Description
protocol	String	IP protocol used by the port. The value can be **TCP** or **UDP**.
targetPort	String	Container port on which the application listens. The value ranges from 1 to 65535.
type	String	Uses the NodePort Service to connect to the load balancer. NodePort indicates the private IP address of a node.

vi ingress-test-ingress.yaml

- For clusters of v1.15 and later, the value of **apiVersion** is **networking.k8s.io/v1beta1**.
- For clusters of v1.13 or earlier, the value of **apiVersion** is **extensions/v1beta1**.

Automatically creating a load balancer

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.subnet-id: 29a0567e-96f1-4227-91cc-64f54d0b064d
    kubernetes.io/elb.enterpriseID: f6b9fffc-a9b7-4a50-a25e-364f587abe44
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth-1551163379627","bandwidth_chargemode":"traffic","bandwidth_size":
5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}'
    kubernetes.io/elb.port: "80"
    kubernetes.io/ingress.class: cce
  name: ingress-test-ingress
spec:
  tls:
  - secretName: ingress-test-secret
  rules:
  - http:
      paths:
      - backend:
          serviceName: ingress-test-svc
          servicePort: 8888
        property:
          ingress.beta.kubernetes.io/url-match-mode: EQUAL_TO
        path: "/healthz"
    host: ingress.com
```

Using an existing load balancer

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/elb.id: f7891f9a-49f2-4ee2-b1ae-f019cd84eb4f
    kubernetes.io/elb.ip: 192.168.0.39
    kubernetes.io/elb.subnet-id: 29a0567e-96f1-4227-91cc-64f54d0b064d
    kubernetes.io/elb.port: "80"
    kubernetes.io/ingress.class: cce

  name: ingress-test-ingress
spec:
  tls:
```

```
- secretName: ingress-test-secret
rules:
- http:
    paths:
    - backend:
        serviceName: ingress-test-svc
        servicePort: 8888
      property:
        ingress.beta.kubernetes.io/url-match-mode: EQUAL_TO
      path: "/healthz"
  host: ingress.com
```

NOTE

Security policy (kubernetes.io/elb.tls-ciphers-policy) is supported only in clusters of v1.17.9 and later.

Table 10-9 Key parameters

Parameter	Type	Description
kubernetes.io/elb.id	String	Optional. This parameter is mandatory if an existing load balancer is used. ID of an enhanced load balancer. The value contains 1 to 100 characters.
kubernetes.io/elb.ip	String	Optional. Do not specify this parameter if a load balancer will be automatically created. Service address of an enhanced load balancer. The value can be the public IP address of a public network load balancer or the private IP address of a private network load balancer.
kubernetes.io/elb.subnet-id	String	Optional. This parameter is mandatory only if a load balancer will be automatically created. For clusters of v1.11.7-r2 or later, this parameter can be left blank. This parameter indicates the subnet ID. The value can contain 1 to 100 characters.
kubernetes.io/elb.enterpriseID	String	Optional. This parameter is mandatory if a public/private network load balancer will be automatically created. This parameter indicates the name of the ELB enterprise project in which the load balancer will be created. The value contains 1 to 100 characters.

Parameter	Type	Description
kubernetes.io/elb.tls-ciphers-policy	String	Security policy used by the listener. This parameter is optional and takes effect only when the HTTPS protocol is used. This parameter is supported only in clusters of v1.17.9 and later.
kubernetes.io/elb.session-affinity-mode	String	Optional. If sticky session is enabled, add this parameter. The value can be **HTTP_COOKIE** or **APP_COOKIE**.
kubernetes.io/elb.session-affinity-option	**elb.session-affinity-option** object	Optional. This parameter includes the Layer 7 ELB session stickiness configuration.
kubernetes.io/elb.autocreate	**elb.autocreate** object	Optional. This parameter is mandatory if a public network load balancer will be automatically created. The system will create a load balancer and an EIP. This parameter is also mandatory if a private network load balancer will be automatically created. The system will create a load balancer. **Example:** ● Value for a public network load balancer that is automatically created: "{\"type\":\"public\",\"bandwidth_name\":\"cce-bandwidth-1551163379627\",\"bandwidth_chargemode\":\"traffic\",\"bandwidth_size\":5,\"bandwidth_sharetype\":\"PER\",\"eip_type\":\"5_bgp\",\"name\":\"james\"}" ● Value for a private network load balancer that is automatically created: "{\"type\":\"inner\"}"
kubernetes.io/elb.lb-algorithm	String	ELB load balancing algorithm. This parameter is optional. Default value: ROUND_ROBIN Value options: ROUND_ROBIN, LEAST_CONNECTIONS, SOURCE_IP, or a null value
kubernetes.io/elb.health-check-flag	String	Whether to enable the ELB health check. This function is enabled by default. This parameter is optional. Value options: on, off, or a null value

Parameter	Type	Description
kubernetes.io/ elb.health-check-option	**elb.health-check-option** object	ELB health check configuration. This parameter is optional.
kubernetes.io/elb.port	Integer	External port registered with the address of the LoadBalancer Service. This parameter is mandatory.
kubernetes.io/ ingress.class	String	By default, **cce** indicates that an ELB will be used, and **nginx** indicates that the nginx-ingress add-on will be enabled. This parameter is mandatory when an ingress is created by calling the API.
tls	Array of strings	Optional. This parameter is mandatory if HTTPS is used.
secretName	String	Optional. This parameter is mandatory if HTTPS is used. Set this parameter to the name of the created secret certificate.
serviceName	String	Name of the ingress-test-svc.yaml Service.
servicePort	Integer	Port number of the ingress-test-svc.yaml Service.
ingress.beta.kubernetes.io/url-match-mode	String	Route matching policy. The options are **EQUAL_TO** (exact matching), **STARTS_WITH** (prefix matching), and **REGEX** (regular expression matching).
path	String	User-defined route.
host	String	Optional. This parameter specifies the domain name.

Table 10-10 Data structure of the elb.autocreate field

Parameter	Type	Description
name	String	Name of the automatically created load balancer. Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit.

Parameter	Type	Description
type	String	Network type of the load balancer. ● **public**: public network load balancer ● **inner**: private network load balancer
bandwidth_name	String	Bandwidth name. The default value is **cce-bandwidth-********. Value range: a string of 1 to 64 characters, including lowercase letters, digits, and underscores (_). The value must start with a lowercase letter and end with a lowercase letter or digit.
bandwidth_chargemode	String	Bandwidth billing mode. The value is **traffic**, indicating that the billing is based on traffic.
bandwidth_size	Integer	Bandwidth size. Set this parameter based on the bandwidth range supported by the region.
bandwidth_sharetype	String	Bandwidth sharing mode. ● **PER**: dedicated bandwidth ● **WHOLE**: shared bandwidth
eip_type	String	EIP type.

Table 10-11 Data structure description of the elb.session-affinity-option field

Parameter	Type	Description
persistence_timeout	String	Sticky session timeout, in seconds. This parameter is valid only when **elb.session-affinity-mode** is set to **HTTP_COOKIE**. Value range: 1 to 1440. Default value: **1440**
app_cookie_name	String	Sticky session timeout, in seconds. This parameter is valid only when **elb.session-affinity-mode** is set to **APP_COOKIE**. The value can contain 1 to 10,000 characters.

Table 10-12 Data structure description of the elb.health-check-option field

Parameter	Type	Description
delay	String	Initial waiting time (in seconds) for starting the health check. This parameter is optional. Value range: 1 to 50. Default value: **5**
timeout	String	Health check timeout, in seconds. This parameter is optional. Value range: 1 to 50. Default value: **10**
max_retries	String	Maximum number of health check retries. This parameter is optional. Value range: 1 to 10. Default value: **3**
protocol	String	Health check protocol. This parameter is optional. Default value: protocol of the associated Service Value options: TCP, UDP_CONNECT, or HTTP
path	String	Health check URL. This parameter is optional, and needs to be configured when the protocol is HTTP. Default value: **/** The value can contain 1 to 10,000 characters.

vi ingress-test-secret.yaml

```
apiVersion: v1
data:
  tls.crt: LS0******tLS0tCg==
  tls.key: LS0tL******0tLS0K
kind: Secret
metadata:
  annotations:
    description: test for ingressTLS secrets
  name: ingress-test-secret
  namespace: default
type: IngressTLS
```

NOTE

In the preceding information, **tls.crt** and **tls.key** are only examples. Replace them with the actual files.

Step 3 Create a workload.

kubectl create -f ingress-test-deployment.yaml

If the following information is displayed, the workload is being created.

```
deployment "ingress-test-deployment" created
```

kubectl get po

If information similar to the following is displayed, the workload has been created successfully.

```
NAME                         READY     STATUS            RESTARTS   AGE
ingress-test-1627801589-r64pk   1/1       Running           0          6s
```

Step 4 Create a secret.

kubectl create -f ingress-test-secret.yaml

If information similar to the following is displayed, the secret is being created.

```
secret "ingress-test-secret" created
```

kubectl get secrets

If information similar to the following is displayed, the secret has been created successfully.

```
NAME                        TYPE                                  DATA      AGE
dash-dashboard              Opaque                                0         7d
dash-dashboard-token-f2nbk   kubernetes.io/service-account-token   3         7d
default-secret              kubernetes.io/dockerconfigjson        1         8d
default-token-wfn4l          kubernetes.io/service-account-token   3         8d
paas.elb                    cfe/secure-opaque                     2         8d
ingress-test-secret          IngressTLS                            2         13s
```

Step 5 Create a Service.

kubectl create -f ingress-test-svc.yaml

If information similar to the following is displayed, the Service is being created:

```
service "ingress-test-svc" created
```

kubectl get svc

If information similar to the following is displayed, the Service has been created successfully:

```
NAME           TYPE         CLUSTER-IP        EXTERNAL-IP   PORT(S)          AGE
ingress-test   NodePort     10.247.189.207    <none>        8080:30532/TCP   5s
kubernetes     ClusterIP    10.247.0.1        <none>        443/TCP          3d
```

kubectl create -f ingress-test-ingress.yaml

If information similar to the following is displayed, the ingress is being created:

```
ingress "ingress-test-ingress" created
```

kubectl get ingress

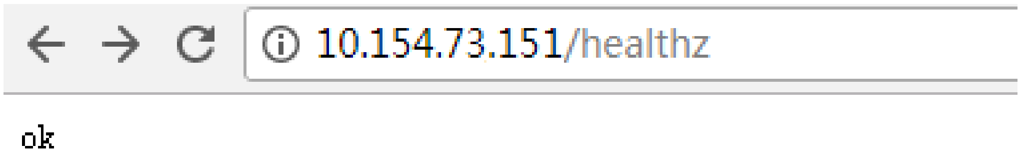
If information similar to the following is displayed, the ingress has been created successfully and the workload is accessible:

```
NAME            HOSTS     ADDRESS          PORTS   AGE
ingress-test    *         10.154.76.63     80      10s
```

Step 6 Enter **http://10.154.76.63/healthz** in the address box of the browser.

10.154.76.63 indicates the IP address of the unified load balancer.

Figure 10-12 Accessing healthz



ok

----End

Updating an Ingress

After adding an ingress, you can update its port, domain name, and route configuration. The procedure is as follows:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Ingresses** tab page, filter ingresses by cluster and namespace, and click **Update** for the ingress to be updated.

Step 2 On the **Update Ingress** page, set the following parameters as follows:

- **External Port**: port number exposed in the LoadBalancer Service address. The port number can be specified randomly.
- **Domain Name** (optional): It indicates the actual domain name to be accessed. You are expected to buy the domain name. Ensure that the domain name can be resolved into the service address of the selected load balancer. If a domain name rule is configured, the domain name must always be used for access.
- **Route Configuration**: You can click **Add Ingress Rule** to add a rule.
  - **Route Matching**: **Prefix match**, **Exact match**, and **Regular expression** are available.
    - 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: Only the URL that is the same as the specified URL can be accessed. For example, if the mapping URL is /healthz, only /healthz can be accessed.
    - Regular expression: The URL rule can be set, for example, **/[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.
  - Mapping URL: Access path to be registered, for example: /healthz.
  - **Service Name**: Select the Service whose ingress is to be added. The Service type is intra-VPC access.

– **Service Port**: port on which the container in the container image listens. For example, the defaultbackend workload listens on port 8080 (container port).

Step 3 Click **Submit**. The ingress will be updated for the workload.

----End

Obtaining Client Source IP Address

If layer-7 load balancing (ingress) is involved, the client source IP address is saved in the **X-Forwarded-For** HTTP header field by default. No other configuration is required.

NOTE

If the Istio service mesh is used, the source IP address cannot be obtained.

In layer-7 load balancing mode, the client IP address cannot be obtained in layer-4 load balancing (that is, the client IP address cannot be obtained by running the **netstat** command). Instead, you need to configure the application server and obtain the client IP address from the **X-Forwarded-For** HTTP header field sent by layer-7 load balancing.

The real IP address is placed in the **X-Forwarded-For** HTTP header field by the load balancer in the following format:

```
X-Forwarded-For: IP address of the client,Proxy server 1-IP address,Proxy server 2-IP address,...
```

Key test point: Obtain the HTTP request header **X-Forward-For** from the container. The obtained IP address is the IP address of the client.

10.6 Network Policy

CCE has enhanced the Kubernetes-based network policy function, allowing network isolation in a cluster by configuring network policies. This means a firewall can be set between instances (pods).

What Are Network Policies?

As the service logic becomes increasingly complex, many applications require network calls between modules. Traditional external firewalls or application-based firewalls cannot meet the requirements. Network policies are urgently needed between modules, service logic layers, or functional teams in a large cluster.

CCE has enhanced the Kubernetes-based network policy function, allowing network isolation in a cluster by configuring network policies. This means a firewall can be set between instances (pods).

For example, to make a payment system accessible only to specified components for security purposes, you can configure network policies.

Precautions

- Network policies are not supported for the VPC Router network model.

- If no network policies have been configured for a workload, such as **workload-1**, other workloads in the same cluster can access **workload-1**.

Creating a Network Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Network**. On the **Network Policies** tab page, click **Create Network Policy**.

- **Network Policy Name**: Specify a NetworkPolicy name.
- **Cluster Name**: Select a cluster to which the network policy belongs.
- **Namespace**: Select a namespace in which the network policy is applied.
- Associate the network policy with a workload:

  Click **Select Workload**. In the dialog box displayed, select a workload for which the network policy is to be created, for example, **workload1**. Then, click **OK**.
- **Rules**: Click **Add Rule**, and set the parameters listed in **Table 10-13**.

Table 10-13 Parameters for adding a rule

Parameter	Description
Direction	Only **Inbound** is supported, indicating that the whitelisted workloads access the current workload (**workload1** in this example).
Protocol	Select a protocol used for workload access.
Destination Container Port	Specify a port on which the workload in the container image listens. The nginx application listens on port 80. If no container port is specified, all ports can be accessed by default.
Remote Node	Select other workloads that can access the current workload. These workloads will access the current workload through the destination container port. – Namespace: All workloads in the selected namespace are added to the whitelist. That is, all workloads in the namespace can access workload A. – Workload: The selected workload can access workload A. Only other workloads in the same namespace as workload A can be selected.

Step 2 Click **Create**.

Step 3 To add more network policies for the current workload when other ports need to be accessed by some workloads, repeat the preceding steps.

After the network policies are created, only the specified workloads or workloads in the specified namespaces can access the current workload.

----End

Configuring a Namespace-level Network Policy

You can configure a namespace-level network policy by enabling network isolation.

By default, **Network Isolation** is disabled for namespaces. For example, if network isolation is off for namespace **default**, **all pods in the current cluster** can access the **pods in namespace default**.

To prevent other pods from accessing the pods in namespace **default**, perform the following steps:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Namespaces**.

Step 2 In the same row as the namespace (for example, **default**) for which you will create a network policy, enable network isolation

After network isolation is enabled, workloads in namespace **default** can access each other but they cannot be accessed by workloads in other namespaces.

----End

11 Storage Management

11.1 Overview

Container storage is a component that provides storage for container workloads. It supports multiple types of storage. A pod can use any amount of storage.

Selecting a Storage Type

You can use the following types of storage when creating a workload: You are advised to store workload data on EVS disks. If you store workload data on a local disk and a fault occurs on the node, the data cannot be restored.

- Local disk: Mount the file directory of the host where a container is located to a specified container path (corresponding to HostPath in Kubernetes). Alternatively, you can leave the source path empty (corresponding to EmptyDir in Kubernetes). If the source path is left empty, a temporary directory of the host will be mounted to the mounting point of the container. A specified source path is used when data needs to be persistently stored on the host, while EmptyDir is used when temporary storage is needed. A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined. For details, see **Using Local Disks as Storage Volumes**.
- EVS: Mount an EVS disk to a container path. When the container is migrated, the mounted EVS disk is migrated together. This storage type is applicable when data needs to be stored permanently. For details, see **Using EVS Disks as Storage Volumes**.
- SFS: Create SFS file systems and mount them to a container path. The file systems created by the underlying SFS service can also be used. SFS file systems are applicable to persistent storage for frequent read/write in multiple workload scenarios, including media processing, content management, big data analysis, and workload analysis. For details, see **Using SFS File Systems as Storage Volumes**.
- Object-based storage: Create OBS object storage volumes and mount them to a container path. Object-based storage applies to scenarios such as cloud

workload, data analysis, content analysis, and hotspot objects. For details, see **Using OBS Buckets as Storage Volumes**.

- SFS turbo: Create SFS turbo file systems and mount them to a container path. SFS turbo file systems are fast, on-demand, and scalable, which makes them suitable for DevOps, containerized microservices, and enterprise office applications. For details, see **Using SFS Turbo File Systems as Storage Volumes**.

11.2 Using Local Disks as Storage Volumes

Use the local disk storage to mount the file directory of the host where the container is located to the specified path of the container, or leave the source path empty.

Local Disk Usage Scenario

Local hard disks are applicable to the following scenarios:

- Host path mounting: Mount the file directory of the host where the container is located to the specified mounting point of the container. If the container needs to access **/etc/hosts**, use **HostPath** to map **/etc/hosts**.
- Temporary path mounting: Used for temporary storage. The lifecycle is the same as that of the container instance. When a container instance disappears, **EmptyDir** will be deleted and the data is permanently lost.
- ConfigMap mounting: Keys in the configuration items of ConfigMap are mapped to a container so that configuration files can be mounted to the specified container directory. For details about how to create a ConfigMap, see **Creating a ConfigMap**. For details about how to use a ConfigMap, see **Using a ConfigMap**.
- Secret mounting: Data in the keys is mounted to a path of the container. A secret is a type of resource that holds sensitive data, such as authentication and key information. All content is user-defined. For details about how to create a secret, see **Creating a secret**. For details about how to use a secret, see **Using a Secret**.

HostPath

In this scenario, the files or directories on the host machine are mounted to a container. Such a volume is usually used to store containerized workload logs that need to be stored permanently or containerized workloads that need to access internal data structure of the Docker engine on the host.

Step 1 Create a workload by referring to **Creating a Deployment** or **Creating a StatefulSet**. When **configuring a container**, expand **Data Storage**, and click **Add Local Volume**.

Step 2 Set parameters for adding a local disk, as listed in **Table 11-1**.

Table 11-1 Setting the volume type to host path mounting

Parameter	Description
Storage Type	Host path.
Host Path	Path of the host to which the local disk is to be mounted, for example, **/etc/hosts**.
Add Container Path	Configure the following parameters: 1. **subPath**: Enter a subpath, for example, **tmp**. A subpath is used to mount a local disk 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. 2. **Container Path**: Enter the path of the container, for example, **/tmp**. This parameter indicates the container path to which a data volume will be mounted. Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Permission: – **Read-only**: You can only read the data volumes mounted to the path. – **Read/Write**: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss. 4. Click **OK**.

----End

EmptyDir

EmptyDir applies to temporary data storage, disaster recovery, and shared running. It will be deleted upon deletion or transfer of workload instances.

Step 1 After adding a container, expand **Data Storage**, and click **Add Local Disk**. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

Step 2 Set the local disk type to temporary path mounting and set parameters for adding a local disk, as shown in **Table 11-2**.

Table 11-2 Setting the volume type to temporary path mounting

Parameter	Description
Storage Type	Temporary path (EmptyDir).
Disk Media	● If you select **Memory**, the running speed is improved, but the storage capacity is limited by the memory size. This mode applies to scenarios where the data volume is small and the read and write efficiency is high. **NOTE** – If you select **Memory**, any files you write will count against your container's memory limit. Pay attention to the memory quota. If the memory usage exceeds the threshold, OOM may occur. – If **Memory** is selected, the size of an emptyDir volume is 50% of the pod specifications and cannot be changed. – If **Memory** is not selected, emptyDir volumes will not occupy the system memory. ● If **Memory** is deselected, data is stored in disks, which is applicable to a large amount of data with low requirements on reading and writing efficiency.
Add Container Path	Configure the following parameters: 1. **subPath**: Enter a subpath, for example, **tmp**. A subpath is used to mount a local disk 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. 2. **Container Path**: Enter the path of the container, for example, **/tmp**. This parameter indicates the container path to which a data volume will be mounted. Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Permission: – **Read-only**: You can only read the data volumes in the path. – **Read/Write**: You can modify the data volumes in the path. Newly written data is not migrated if the container is migrated, which may cause a data loss. 4. Click **OK**.

----End

ConfigMap

The platform provides the separation of workload codes and configuration files. ConfigMap mounting is used to process workload configuration parameters. You need to create workload configurations in advance. For more information, see **Creating a ConfigMap**.

Step 1 After adding a container, expand **Data Storage**, and click **Add Local Disk**. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

Step 2 Set the local disk type to ConfigMap mounting and set parameters for adding a local disk, as shown in **Table 11-3**.

Table 11-3 Setting the volume type to ConfigMap mounting

Parameter	Description
Storage Type	ConfigMap.
Option	Select the desired ConfigMap name. **NOTE** The ConfigMap must be created in advance. For details, see **Creating a ConfigMap**.
Add Container Path	Configure the following parameters: 1. **subPath**: Enter a subpath, for example, **tmp**. A subpath is used to mount a local disk 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. 2. **Container Path**: Enter the path of the container, for example, **/tmp**. This parameter indicates the container path to which a data volume will be mounted. Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Permission: read-only. Data volumes in the path are read-only. 4. Click **OK**.

----End

Secret

Mount the data in the key to the specified container. The content of the key is user-defined. You need to create application configurations in advance. For more information, see **Creating a secret**.

Step 1 After adding a container, expand **Data Storage**, and click **Add Local Disk**. For details, see **Creating a Deployment** or **Creating a StatefulSet**.

Step 2 Set the local disk type to secret mounting and set parameters for adding a local disk, as shown in **Table 11-4**.

Table 11-4 Setting the volume type to secret mounting

Parameter	Description
Storage Type	Secret.
Secret	Select the desired secret name. **NOTE** The secret must be created in advance. For details, see **Creating a secret**.
Add Container Path	Configure the following parameters: 1. **subPath**: Enter a subpath, for example, **tmp**. A subpath is used to mount a local disk 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. 2. **Container Path**: Enter the path of the container, for example, **/tmp**. This parameter indicates the container path to which a data volume will be mounted. Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Permission: read-only. Data volumes in the path are read-only. 4. Click **OK**.

----End

Mounting a Host Path Using kubectl

You can mount a file directory of the host where the container is located to a specified mount path of the container using kubectl.

Step 1 Configure kubectl. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Run the following commands to configure the **hostPath-pod-example.yaml** file, which is used to create a pod.

touch hostPath-pod-example.yaml

vi hostPath-pod-example.yaml

Mount the host path for the Deployment. The following is an example:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: hostpath-pod-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hostpath-pod-example
  template:
    metadata:
      labels:
        app: hostpath-pod-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: hostpath-example
      restartPolicy: Always
      volumes:
      - name: hostpath-example
        hostPath:
          path: /tmp
```

In the preceding information,

- **name**: name of the pod to be created.
- **app**: name of the pod workload.
- **mountPath**: mount path of the container. In this example, the hostPath volume is mounted to the **/tmp** directory.
- **hostPath**: host path. In this example, the host path is **/tmp**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Step 3 Run the following command to create the pod:

kubectl create -f hostPath-pod-example.yaml

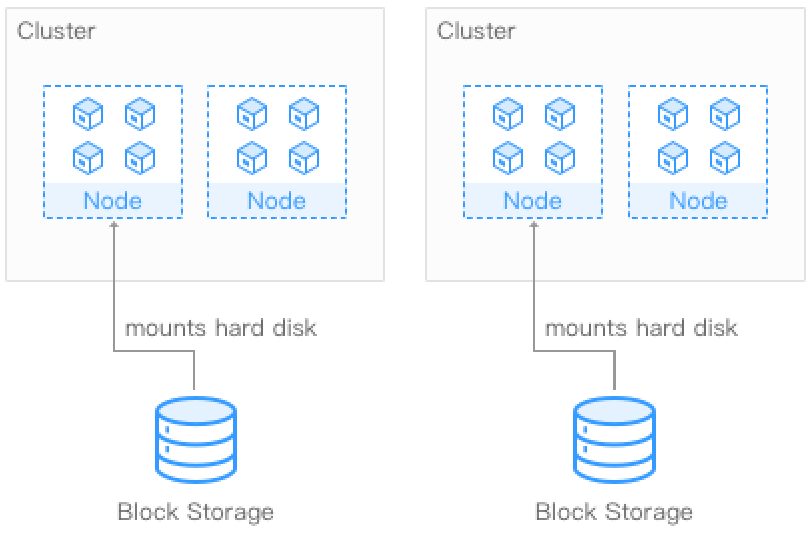
----End

11.3 Using EVS Disks as Storage Volumes

11.3.1 Usage Description for EVS Volumes

To meet data persistency requirements, CCE allows EVS disks to be attached as the storage volumes to containers. By using EVS volumes, you can attach the remote file directory of a storage system to a container so that data in the data volume is permanently preserved. Even if the container is deleted, the data in the data volume is still stored in the storage system.

Figure 11-1 Attaching EVS volumes to CCE



Instructions

- Similar to formatting disks for on-site servers in traditional layouts, you can format block storage (disks) attached to cloud servers, and create file systems on them.
- Data cannot be shared. Each server uses an independent block storage (disk), and data of multiple servers is isolated.
- Private network: Data must be accessed in the internal network of the data center.
- The capacity of a single volume is limited (TB-level), but the performance is excellent (in milliseconds). This type of volumes is mainly used for databases and enterprise office applications.
- EVS disks that have partitions or have non-ext4 file systems cannot be imported.
- This type of volumes applies to single-pod Deployments, jobs, and StatefulSets.

11.3.2 Using EVS Disks

Constraints

- EVS disks cannot be attached across AZs or used by multiple workloads, multiple pods of the same workload, or multiple jobs.
- The data sharing function of a shared disk is not supported between nodes in a CCE cluster. If the same EVS disk is attached to multiple nodes, read and write conflicts and data cache conflicts may occur. Therefore, you can create only one pod when creating a Deployment that uses EVS disks.
- When you create a StatefulSet and add a cloud storage volume, existing EVS volumes cannot be used.
- Container storage in CCE clusters of Kubernetes 1.13 or later version supports encryption. Currently, E2E encryption is supported only in certain regions.

Creating an EVS Volume

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. Click **Create EVS Disk**.

Step 2 Configure basic disk information. **Table 11-5** describes the parameters.

Table 11-5 Configuring basic disk information

Parameter	Description
* PVC Name	Name of the PVC to be created. A storage volume is automatically created when a PVC is created. One PVC corresponds to one storage volume. The storage volume name is automatically generated when the PVC is created.
Cluster Name	Cluster where the EVS disk is deployed.
Namespace	Select the namespace where the EVS disk is deployed. If you do not need to select a namespace, retain the default value.
Volume Capacity (GB)	Capacity of the storage to be created.
Select Data Source	Currently, only snapshots can be used to create EVS disks.
Access Mode	Access permissions of user applications on storage resources (PVs). ● **ReadWriteOnce** (RWO): Data can be read from and written into a volume, but the volume can be attached to only one node. ● **ReadWriteMany** (RWX): Data can be read from and written into a volume, and the volume can be attached to multiple nodes at the same time.
AZ	AZ to which the volume belongs.

Parameter	Description
Type	Type of the new EVS disk.
Storage Format	Storage format of the EVS disk. The default value is **CSI**. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers. This parameter is displayed only for clusters of v1.15.6-r1 and later.
Encryption	**KMS Encryption** is deselected by default. If **KMS Encryption** is selected, set the following parameters: ● **Agency Name**: Agencies can be used to assign permissions to trusted accounts or cloud services for a specific period of time. If no agency is created, click **Create Agency**. The agency name **EVSAccessKMS** indicates that EVS is granted the permission to access KMS and can obtain KMS keys to encrypt and decrypt EVS disks. ● **Key Name**: A key is a type of resource that holds user-defined, sensitive data, such as authentication and key information. This resource object can be loaded into containerized applications. ● **Key ID**: generated by default.

Step 3 Click **Next**. Review order details, click **Submit**, and wait until the creation is successful.

The new storage resource is displayed in the list. When its status becomes **Normal**, the storage resource is created successfully.

Step 4 Click the storage resource name to view detailed information.

----End

Using EVS Disks

Step 1 Create a workload by referring to **Creating a Deployment** or **Creating a StatefulSet**. After you add a container, expand **Data Storage**. On the **Cloud Volume** tab page, click **Add Cloud Volume**.

Step 2 Set the storage volume type to **EVS**.

Table 11-6 Parameters required for mounting EVS disks

Parameter	Description
Type	**EVS**: You can use EVS disks the same way you use traditional hard disks on servers. EVS disks deliver higher data reliability and I/O throughput and are easy to use. They can be used for file systems, databases, or other system software and applications that require block storage resources. **CAUTION** ● **EVS disks can be attached only to single-pod workloads.** ● When you create a StatefulSet and add a cloud storage volume, existing EVS volumes cannot be used. ● EVS disks cannot be attached across AZs or used by multiple workloads, multiple pods of the same workload, or multiple jobs.
Allocation Mode	
Manual	Select an existing storage volume. If no storage volume is available, create one.

Parameter	Description
Automatic	A storage volume is created automatically. You need to enter the storage capacity. 1. If you selected EVS disk as the storage volume type, you need to select an AZ for creating the EVS disk. 2. Select an access mode. – **ReadWriteOnce**: The volume can be attached as read-write by a single node. – **ReadWriteMany**: The volume can be attached as read-write by many nodes. 3. Select a storage sub-type. – High I/O: EVS disks that have high I/O and use SAS – Common I/O: EVS disks that use SATA – Ultra-high I/O: EVS disks that have ultra-high I/O and use SSD 4. Select a data source. Currently, only **Create from snapshot** is supported. 5. Enter the storage capacity, in the unit of GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail. 6. **Storage Format**: The default value is **CSI**. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers. This parameter is displayed only for clusters of v1.15.6-r1 or later. 7. **Encryption**: Select **KMS Encryption** and set the following parameters: – **Agency Name**: Agencies can be used to assign permissions to trusted accounts or cloud services for a specific period of time. If no agency is created, click **Create Agency**. The agency name **EVSAccessKMS** indicates that EVS is assigned the permission to access KMS and can obtain KMS keys to encrypt and decrypt EVS disks. – **Key Name**: A key is a type of resource that holds user-defined, sensitive data, such as authentication and key information. This resource object can be loaded into containerized applications. – **Key ID**: generated by default.

Parameter	Description
Add Container Path	1. Click **Add Container Path**. 2. **Container Path**: Enter the container path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may cause a container error to occur. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. – If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Set permissions. – **Read-only**: You can only read the data volumes mounted to the path. – **Read/Write**: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

Step 3 Click **OK**.

----End

Importing an EVS Disk

CCE allows you to import existing EVS disks.

NOTE

An EVS disk can be imported into only one namespace. If an EVS disk has been imported into a namespace, it is invisible to other namespaces and cannot be imported again. **If you want to import an EVS disk that has file system (ext4) formatted, ensure that no partition has been created for the disk. Otherwise, data may be lost.**

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **EVS** tab page, click **Import**.

Step 2 Select one or more EVS disks that you want to attach and mount. Then, click **OK**.

----End

Unbinding an EVS Disk

After an EVS disk is successfully created or imported, the EVS disk is automatically bound to the current cluster and cannot be used by other clusters. When the EVS disk is unbound from the cluster, it is available to other clusters.

If the EVS disk has been mounted to a workload, it cannot be unbound from the cluster.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the EVS disk list, click **Unbind** next to the target EVS disk.

Step 2 Confirm the unbinding, and click **OK**.

----End

Related Operations

After an EVS disk is created, you can perform operations described in **Table 11-7**.

Table 11-7 Other operations

Operation	Description
Delete an EVS disk	1. Select the EVS disk to be deleted and click **Delete** in the **Operation** column. 2. Follow the prompts to delete the object.

11.3.3 Creating an EVS Disk Using kubectl

Scenario

CCE supports the creation of EVS disks in the form of PersistentVolumeClaim (PVC).

Procedure

Step 1 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Run the following commands to configure the **pvc-evs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-evs-auto-example.yaml

vi pvc-evs-auto-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto-example
  namespace: default
  annotations:
    everest.io/disk-volume-type: SATA
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-disk
```

In this example:

- **everest.io/disk-volume-type**: EVS disk type, which is in uppercase. Currently, high I/O (SAS), ultra-high I/O (SSD), and common I/O (SATA) are supported.
- **failure-domain.beta.kubernetes.io/region**: region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **storage**: storage capacity in the unit of Gi.
- **storageClassName**: name of the Kubernetes storage class associated with the dynamic provisioning of storage volumes. The name of the storage class associated with the CSI used by the clusters of v1.15 is **csi-disk**.
- **accessModes**: read/write mode. Clusters of v1.15 support only non-shared volumes. Set this parameter to **ReadWriteOnce**.

● **Example YAML file for clusters of v1.9, v1.11, and v1.13:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-evs-auto-example
  namespace: default
  annotations:
    volume.beta.kubernetes.io/storage-class: sata
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: EVS disk type. Currently, high I/O (sas), ultra-high I/O (ssd), or common I/O (sata) are supported.
- **failure-domain.beta.kubernetes.io/region**: region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **storage**: storage capacity in the unit of Gi.
- **accessModes**: read/write mode. The value can be **ReadWriteMany** (shared volume) or **ReadWriteOnly** (non-shared volume).

Step 3 Run the following command to create a PVC.

kubectl create -f pvc-evs-auto-example.yaml

After the command is executed, an EVS disk is created in the partition where the cluster is located. Choose **Storage** > **EVS** to view the EVS disk. Alternatively, you can view the EVS disk based on the volume name on the EVS console.

----End

11.3.4 Creating a PVC Using an Existing EVS Disk

Scenario

CCE allows you to create a PersistentVolume (PV) using an existing EVS disk. After the PV is created, you can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Procedure

CCE allows you to create a PersistentVolume (PV) using an existing EVS disk. After the PV is created, you can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Step 1 Log in to the EVS console, create an EVS disk, and record the volume ID, capacity, and disk type of the EVS disk.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a YAML file for creating the PersistentVolume. Assume that the file name is **pv-evs-example.yaml**.

touch pv-evs-example.yaml

vi pv-evs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: disk.csi.everest.io
    fsType: ext4
    volumeAttributes:
      everest.io/disk-mode: SCSI
      everest.io/disk-volume-type: SATA
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-disk
```

  In this example:

  – **failure-domain.beta.kubernetes.io/region**: region where the EVS disk is located.

  – **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.

  – **driver**: storage driver used to mount the volume. Set the driver to **disk.csi.everest.io** for the EVS volume.

  – **volumeHandle**: volume ID of the EVS disk.

- **everest.io/disk-mode**: device type of the EVS disk. The value can be **SCSI**.
- **storage**: EVS volume capacity in the unit of Gi.
- **everest.io/disk-volume-type**: EVS disk type. Currently, high I/O (SAS), ultra-high I/O (SSD), and common I/O (SATA) are supported.
- **storageClassName** specifies the name of the Kubernetes storage class associated with the storage volume. Set this field to **csi-disk** for EVS disks.

- **Example YAML file for clusters of version 1.11.7 or later and version 1.13:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      disk-mode: SCSI
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sata
```

In this example:

- **failure-domain.beta.kubernetes.io/region**: region where the EVS disk is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxivol** for the EVS volume.
- **volumeID**: volume ID of the EVS disk.
- **disk-mode**: EVS disk mode.
- **storage**: EVS volume capacity in the unit of Gi.
- **storageClassName**: EVS disk type. Currently, high I/O (sas), ultra-high I/O (ssd), and common I/O (sata) are supported.

- **Example YAML file for clusters earlier than v1.11.7:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pv-evs-example
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sata
```

In this example:

- **failure-domain.beta.kubernetes.io/region**: region where the EVS disk is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxivol** for the EVS volume.
- **volumeID**: volume ID of the EVS disk.
- **disk-mode**: EVS disk mode.
- **storage**: EVS volume capacity in the unit of Gi.
- **storageClassName**: EVS disk type. Currently, high I/O (sas), ultra-high I/O (ssd), and common I/O (sata) are supported.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pv-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      kubernetes.io/namespace: default
      volumeID: 0992dbda-6340-470e-a74e-4f0db288ed82
  persistentVolumeReclaimPolicy: Delete
  storageClassName: sata
```

In this example:

- **failure-domain.beta.kubernetes.io/region**: region where the EVS disk is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxivol** for the EVS volume.
- **volumeID**: volume ID of the EVS disk.
- **disk-mode**: EVS disk mode.
- **storage**: EVS volume capacity in the unit of Gi.
- **storageClassName**: EVS disk type. Currently, high I/O (sas), ultra-high I/O (ssd), and common I/O (sata) are supported.

Step 4 Create a PV.

kubectl create -f pv-evs-example.yaml

Step 5 (Optional) Add the metadata associated with the cluster to ensure that the EVS disks associated with the attached static PV are not deleted when the node or cluster is deleted.

⚠ CAUTION

If you do not perform **Step 5** when using an existing EVS disk to create a PVC or when creating a static PV or PVC, ensure that the EVS disk associated with the static PV has been detached from the node before you delete the node.

1. Obtain the user token.
2. Obtain the EVS access address **EVS_ENDPOINT**.
3. Add the metadata associated with the cluster to the EVS disk associated with the static PV.

```
curl -X POST ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --insecure \
    -d '{"metadata":{"cluster_id": "${cluster_id}", "namespace": "${pvc_namespace}"}}' \
    -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' \
    -H 'X-Auth-Token:${TOKEN}'
```

In this example:

– **EVS_ENDPOINT**: EVS access address. Set this parameter to the value obtained in **Step 5.2**.

– **project_id**: project ID.

– **volume_id**: ID of the associated EVS disk. Set this parameter to the value of **volume_id** in 2. You can also log in to the EVS console, click the name of the EVS disk to be imported, and obtain the ID from **Summary** on the disk details page.

– **cluster_id**: ID of the cluster where the EVS PV is to be created. On the CCE console, choose **Resource Management** > **Clusters**. Click the name of the cluster to be associated with. On the cluster details page, obtain the cluster ID.

– **pvc_namespace**: namespace to which the PVC is to be bound.

– **TOKEN**: user token. Set this parameter to the value obtained in **Step 5.1**.

For example, run the following commands:

```
curl -X POST https://evs.eu-west-0.prod-cloud-ocb.orange-business.com:443/
v2/060576866680d5762f52c0150e726aa7/volumes/69c9619d-174c-4c41-837e-31b892604e14/
metadata --insecure \
    -d '{"metadata":{"cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442", "namespace":
"default"}}' \
    -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' \
    -H 'X-Auth-Token:MIIPe******IsIm1ldG
```

After the request is executed, run the following command to check whether the EVS disk has been associated with the metadata of the cluster:

```
curl -X GET ${EVS_ENDPOINT}/v2/${project_id}/volumes/${volume_id}/metadata --insecure \
    -H 'X-Auth-Token:${TOKEN}'
```

For example, run the following commands:

```
curl -X GET https://evs.eu-west-0.prod-cloud-ocb.orange-business.com/
v2/060576866680d5762f52c0150e726aa7/volumes/69c9619d-174c-4c41-837e-31b892604e14/
metadata --insecure \
    -H 'X-Auth-Token:MIIPeAYJ***9t1c31ASaQ=='
```

The command output displays the current metadata of the EVS disk.

```
{
    "metadata": {
        "namespace": "default",
        "cluster_id": "71e8277e-80c7-11ea-925c-0255ac100442",
        "hw:passthrough": "true"
    }
}
```

Step 6 Create a YAML file for creating the PVC. Assume that the file name is **pvc-evs-example.yaml**.

touch pvc-evs-example.yaml

vi pvc-evs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  annotations:
    everest.io/disk-volume-type: SATA
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeName:  pv-evs-example
  storageClassName: csi-disk
```

  In this example:

  – **everest.io/disk-volume-type** specifies the EVS disk type. Value options include **SAS**, **SSD**, or **SATA**. The value must be the same as that of the existing PV.
  – **failure-domain.beta.kubernetes.io/region**: region where the cluster is located.
  – **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
  – **storage**: requested capacity of the PVC, which must be the same as the storage size of the existing PV.
  – **volumeName**: name of the PV.
  – **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-disk** for EVS disks.

- **Example YAML file for clusters of v1.11 and v1.13:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sata
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
```

```
 - ReadWriteMany
 resources:
   requests:
     storage: 10Gi
 volumeName: pv-evs-example
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class. The value can be **sas**, **ssd**, or **sata**. The value must be the same as that of the existing PV.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxivol**.
- **failure-domain.beta.kubernetes.io/region**: region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload.
- **storage**: requested capacity of the PVC, which must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sata
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: eu-west
    failure-domain.beta.kubernetes.io/zone: eu-west-0a
  name: pvc-evs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-evs-example
  volumeNamespace: default
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class. The value can be **sas**, **ssd**, or **sata**. The value must be the same as that of the existing PV.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxivol**.
- **failure-domain.beta.kubernetes.io/region**: region where the cluster is located.
- **failure-domain.beta.kubernetes.io/zone**: AZ where the EVS volume is created. It must be the same as the AZ planned for the workload. **storage**: requested capacity of the PVC, which must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

Step 7 Create a PVC.

kubectl create -f pvc-evs-example.yaml

After the operation is successful, choose **Resource Management** > **Storage** to view the created PVC. You can also view the EVS disk by name on the EVS page.

----End

11.3.5 Attaching an EVS Volume Using kubectl

Scenario

After an EVS disk is created or imported to CCE, you can attach it to a workload.

NOTICE

EVS disks cannot be mounted across AZs. Before mounting, you can run the **kubectl get pvc** command to query the available PVCs in the AZ where the current cluster is located.

Procedure

Step 1 Run the following commands to configure the **evs-pod-example.yaml** file, which is used to create a pod.

touch evs-pod-example.yaml

vi evs-pod-example.yaml

- Example of attaching an EVS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: evs-pod-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: evs-pod-example
  template:
    metadata:
      labels:
        app: evs-pod-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-evs-example
      restartPolicy: Always
      volumes:
      - name: pvc-evs-example
        persistentVolumeClaim:
          claimName: pvc-evs-auto-example
```

  In this example:

  – **name**: name of the Deployment.

  – **app**: name of the application running in the pod.

  - **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
  - **claimName**: name of an existing PVC.
  - **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.
- Example of mounting an EVS disk to a StatefulSet (PVCTemplate-based, dedicated volume):
  - **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-evs-sata-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: ***/swr/dockerimage/nginx.png
    description: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-evs-sata-in
      failure-domain.beta.kubernetes.io/region: eu-west
      failure-domain.beta.kubernetes.io/zone: eu-west-0a
  template:
    metadata:
      labels:
        app: deploy-evs-sata-in
        failure-domain.beta.kubernetes.io/region: eu-west
        failure-domain.beta.kubernetes.io/zone: eu-west-0b
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-evs-sata-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: a2cd8e998dca42e98a41f596c636dbda
          resources: {}
          volumeMounts:
            - name: bs-sata-mountoptionpvc
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: bs-sata-mountoptionpvc
        namespace: default
        annotations:
          everest.io/disk-volume-type: SATA
        enable: true
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
        storageClassName: csi-disk
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

– **Example YAML file for clusters of version 1.13 or earlier:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-evs-sata-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: ***/swr/dockerimage/nginx.png
    description: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-evs-sata-in
      failure-domain.beta.kubernetes.io/region: eu-west
      failure-domain.beta.kubernetes.io/zone: eu-west-0a
  template:
    metadata:
      labels:
        app: deploy-evs-sata-in
        failure-domain.beta.kubernetes.io/region: eu-west
        failure-domain.beta.kubernetes.io/zone: eu-west-0b
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-evs-sata-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: a2cd8e998dca42e98a41f596c636dbda
          resources: {}
          volumeMounts:
            - name: bs-sata-mountoptionpvc
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: bs-sata-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: sata
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxivol

      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 10Gi
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 2 Run the following command to create a pod:

kubectl create -f evs-pod-example.yaml

After the creation is complete, log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage** > **EVS**. Then, click the PVC name. On the PVC details page, you can view the binding relationship between the EVS disk and the PVC.

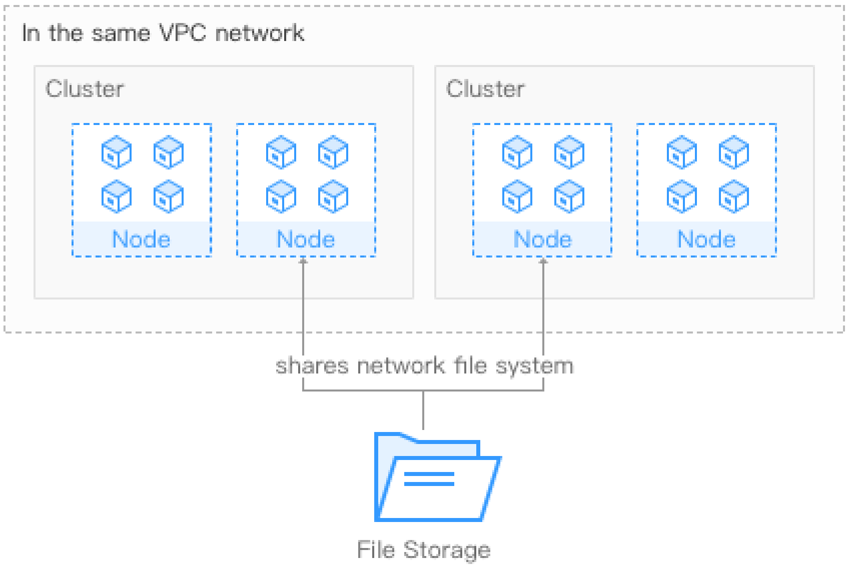
----End

11.4 Using SFS File Systems as Storage Volumes

11.4.1 Usage Description for SFS Volumes

File storage is applicable to various scenarios, such as media processing, content management, big data jobs, and workload analysis.

Figure 11-2 Attaching SFS volumes to CCE



Instructions

- Complying with standard file protocols, file systems can be mounted 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: Data must be accessed in the internal network of the data center.
- The capacity of a single file system is high (at PB level) and the performance is excellent (at ms level), suitable to media editing, HPC, and file sharing scenarios.
- This mode is applicable to Deployments, StatefulSets, and jobs in ReadWriteMany scenarios.

11.4.2 Using SFS Volumes

Constraints

- CCE clusters earlier than v1.7.3-r2 do not support SFS. Therefore, you need to create a cluster that supports SFS first.
- E2E encryption is supported for container storage in CCE clusters of Kubernetes 1.13 or later in certain regions.

Creating an SFS Volume

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.

Step 2 On the **SFS** tab, click **Create SFS File System**.

Step 3 Configure basic information, as shown in **Table 11-8**.

Table 11-8 Parameters for creating an SFS volume

Parameter	Description
* PVC Name	Name of the new PVC, which is different from the volume name. The actual volume name is automatically generated when the PVC is created.
Cluster Name	Cluster to which the file system volume belongs.
Namespace	Namespace in which the volume is created.
Total Capacity	The total capacity is the capacity of a single volume. Fees are charged by actual usage.
Access Mode	**ReadWriteMany**: The volume can be attached as read-write by many nodes.
Storage Format	Storage format of new files. The default value is **CSI**. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers. This parameter is displayed only for clusters of v1.15.6-r1 and later.
Encryption	**KMS Encryption** is deselected by default. If **KMS Encryption** is selected, set the following parameters: ● **Agency Name**: Agencies can be used to assign permissions to trusted accounts or cloud services for a specific period of time. If no agency is created, click **Create Agency**. The agency name **SFSAccessKMS** indicates that SFS is granted the permission to access KMS. After SFS is authorized successfully, it can obtain KMS keys to encrypt and decrypt file systems. ● **Key Name**: A key is a type of resource that holds user-defined, sensitive data, such as authentication and key information. This resource object can be loaded into containerized applications. ● **Key ID**: generated by default.

Step 4 Click **Submit**.

The new storage resource is displayed in the list. When its status becomes **Normal**, the storage resource is created successfully.

Step 5 Click the storage resource name to view detailed information.

----End

Using SFS Volumes

Step 1 Create a workload by referring to **Creating a Deployment** or **Creating a StatefulSet**. After you add a container, expand **Data Storage**. On the **Cloud Volume** tab page, click **Add Cloud Volume**.

Step 2 Set the storage class to **SFS**.

Table 11-9 Parameters for mounting a file system

Parameter	Description
Type	**File Storage (NFS)**: This type applies to a wide range of scenarios, including media processing, content management, big data, and application analysis.
Allocation Mode	
Manual	**Name**: Select a created file system. You need to create a file system in advance. For details about how to create a file system, see **Creating an SFS Volume**.

Parameter	Description
Automatic	A storage volume is created automatically. You need to enter the storage capacity. 1. Select a storage sub-type. Set the sub-type to **NFS**. 2. Enter the storage capacity, in the unit of GB. Ensure that the storage capacity quota is not exceeded; otherwise, creation will fail. 3. **Storage Format**: The default value is **CSI**. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers. This parameter is displayed only for clusters of v1.15.6-r1 or later. 4. **Encryption**: Select **KMS Encryption** and set the following parameters: – **Agency Name**: Agencies can be used to assign permissions to trusted accounts or cloud services for a specific period of time. If no agency is created, click **Create Agency**. The agency name **SFSAccessKMS** indicates that SFS is granted the permission to access KMS. After SFS is authorized successfully, it can obtain KMS keys to encrypt and decrypt file systems. – **Key Name**: A key is a type of resource that holds user-defined, sensitive data, such as authentication and key information. This resource object can be loaded into containerized applications. – **Key ID**: generated by default.

Parameter	Description
Add Container Path	1. Click **Add Container Path**. 2. **subPath**: Enter the subpath of the file storage. **NOTICE** In Kubernetes, the subpath to which the data volume is mounted refers to the subpath of the referenced volume rather than its root. The subpath must be a relative path and cannot start with a slash (/) or ../. If this parameter is not set, the root directory is used by default. 3. **Container Path**: Enter the container path to which the data volume is mounted. **NOTICE** – Do not mount a data volume to a system directory such as **/** or **/var/run**; this action may result in abnormal containers. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. – If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 4. Set permissions. – **Read-only**: You can only read the data volumes mounted to the path. – **Read/Write**: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

Step 3 Click **OK**.

----End

Importing SFS Volumes

CCE allows you to import existing SFS volumes.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **SFS** tab page, click **Import**.

Step 2 Select one or more SFS volumes that you want to attach.

Step 3 Select the target cluster and namespace. Then, click **OK**.

----End

Unbinding an SFS Volume

When an SFS volume is successfully created or imported, the volume is automatically bound to the current cluster. Other clusters can also use the volume. When the volume is unbound from the cluster, it is still available to other clusters.

If the SFS volume has been attached to a workload, the volume cannot be unbound from the cluster.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the file system list, click **Unbind** next to the target volume.

Step 2 Confirm the unbinding, and click **OK**.

----End

Related Operations

After an SFS volume is created, you can perform the operation described in **Table 11-10**.

Table 11-10 Other operations

Operation	Description
Deleting an SFS volume	1. Select the SFS volume to be deleted and click **Delete** in the **Operation** column. 2. Follow the prompts to delete the object.
Importing an SFS volume	CCE allows you to import existing SFS volumes. 1. On the **SFS** tab page, click **Import**. 2. Select one or more SFS volumes that you want to attach. 3. Select the target cluster and namespace. 4. Click **OK**.

11.4.3 Using kubectl to Create a File System

Scenario

CCE allows you to create a file system in the form of PersistentVolumeClaim (PVC).

Procedure

Step 1 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Run the following commands to configure the **pvc-sfs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-sfs-auto-example.yaml

vi pvc-sfs-auto-example.yaml

- **Example YAML file for clusters of version 1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name:  pvc-sfs-auto-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-nas
```

In this example:

- **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-nas**.
- **name**: name of the PVC to be created.
- **storage**: storage capacity in the unit of Gi.

- **Example YAML file for clusters of version 1.13 or earlier:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
  name: pvc-sfs-auto-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: file storage class. Currently, the standard file protocol type (nfs-rw) is supported.
- **name**: name of the PVC to be created.
- **storage**: storage capacity in the unit of Gi.

Step 3 Run the following command to create a PVC.

kubectl create -f pvc-sfs-auto-example.yaml

After the command is executed, an SFS file system is created in the VPC to which the cluster belongs. Choose **Resource Management** > **Storage** > **SFS** or log in to the SFS console to view the file system.

----End

11.4.4 Using an Existing File System to Create a PVC

Scenario

CCE allows you to use an existing file system to create a PersistentVolume (PV). After the creation is successful, create the corresponding PersistentVolumeClaim (PVC) and bind it to the PV.

Procedure

Step 1 Log in to the SFS console, create a file system, and record the file system ID, shared path, and capacity.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a YAML file for creating the PersistentVolume. Assume that the file name is **pv-sfs-example.yaml**.

touch pv-sfs-example.yaml

vi pv-sfs-example.yaml

- **Example YAML file for clusters of version 1.15 and later:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: nas.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: sfs-nas01.eu-west.ocbcloud.com:/share-436304e8
      storage.kubernetes.io/csi
    ProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 682f00bb-ace0-41d8-9b3e-913c9aa6b695
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-nas
```

  In this example:

  – **driver**: storage driver used to mount the volume. Set the driver to **nas.csi.everest.io** for the file system.
  – **everest.io/share-export-location**: shared path of the file system.
  – **volumeHandle**: file system ID.
  – **storage**: file system size.
  – **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-nas**.

- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: sfs-nas1.eu-west.ocbcloud.com:/share-73bdfafd
      fsType: nfs
      volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
  persistentVolumeReclaimPolicy: Delete
  storageClassName: nfs-rw
```

  In this example:

  – **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxinfs** for the file system.

  - **deviceMountPath**: shared path of the file system.
  - **volumeID**: file system ID.
  - **storage**: file system size.
  - **storageClassName**: read/write mode supported by the file system. Currently, **nfs-rw** and **nfs-ro** are supported.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  flexVolume:
    driver: huawei.com/fuxinfs
    fsType: nfs
    options:
      deviceMountPath: sfs-nas1.eu-west.ocbcloud.com:/share-73bdfafd
      fsType: nfs
      kubernetes.io/namespace: default
      volumeID: f6976f9e-2493-419b-97ca-d7816008d91c
  persistentVolumeReclaimPolicy: Delete
  storageClassName: nfs-rw
```

  In this example:

  - **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxinfs** for the file system.
  - **deviceMountPath**: shared path of the file system.
  - **volumeID**: file system ID.
  - **storage**: file system size.
  - **storageClassName**: read/write mode supported by the file system. Currently, **nfs-rw** and **nfs-ro** are supported.

NOTE

The VPC to which the file system belongs must be the same as the VPC of the ECS VM to which the workload is planned.

Step 4 Create a PV.

kubectl create -f pv-sfs-example.yaml

Step 5 Create a YAML file to create a PVC. Assume that the file name is **pvc-sfs-example.yaml**.

touch pvc-sfs-example.yaml

vi pvc-sfs-example.yaml

- **Example YAML file for clusters of version 1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
```

```
 resources:
   requests:
     storage: 10Gi
 storageClassName: csi-nas
 volumeName: pv-sfs-example
```

In this example:

- **storageClassName**: Set this field to **csi-nas**. The value must be the same as that of the existing PV.
- **storage**: storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: read/write mode supported by the file system. The value can be **nfs-rw** or **nfs-ro**. The value must be the same as that of the existing PV.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxinfs**.
- **storage**: storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: nfs-rw
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
  name: pvc-sfs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-sfs-example
  volumeNamespace: default
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: read/write mode supported by the file system. The value can be **nfs-rw** or **nfs-ro**. The value must be the same as that of the existing PV.

  - **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxinfs**.
  - **storage**: storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
  - **volumeName**: name of the PV.

Step 6 Create a PV.

kubectl create -f pv-sfs-example.yaml

----End

11.4.5 Using an Existing PVC to Create a StatefulSet

Scenario

CCE allows you to use the existing file system (PersistentVolumeClaim) to create a workload (StatefulSet).

Procedure

Step 1 Create an SFS volume by referring to **Creating an SFS Volume** and record the volume name.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a PVC file for creating the workload. Assume that the file name is **sfs-statefulset-example.yaml**.

touch sfs-statefulset-example.yaml

vi sfs-statefulset-example.yaml

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sfs-statefulset-example
  namespace: default
spec:
  podManagementPolicy: OrderedReady
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: sfs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: "true"
      creationTimestamp: null
      labels:
        app: sfs-statefulset-example
    spec:
      affinity: {}
      containers:
      - env:
        - name: PAAS_APP_NAME
          value: sfs-statefulset-example
        - name: PAAS_NAMESPACE
          value: default
        - name: PAAS_PROJECT_ID
          value: b7bb7d77a2974a8fa8985cbfb63f23c0
        image: nginx:latest
        imagePullPolicy: Always
        name: container-0
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /tmp
          name: pvc-sfs-example
      dnsPolicy: ClusterFirst
      imagePullSecrets:
      - name: default-secret
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - name: pvc-sfs-example
          persistentVolumeClaim:
            claimName: cce-sfs-demo
      tolerations:
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
        tolerationSeconds: 300
      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists
        tolerationSeconds: 300
  updateStrategy:
    type: RollingUpdate
```

In this example:

- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.
- **replicas**: number of pods.
- **name**: name of the new workload.
- **image**: image used by the workload.
- **mountPath**: mount path of a container.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **claimName**: name of an existing PVC.

Step 4 Create a StatefulSet.

kubectl create -f sfs-statefulset-example .yaml

----End

11.4.6 Using kubectl to Deploy a Workload with an SFS Volume Attached

Scenario

After an SFS volume is created or imported to CCE, you can attach the volume to a workload.

Procedure

Step 1 Run the following commands to configure the **sfs-pod-example.yaml** file, which is used to create a pod.

touch sfs-pod-example.yaml

vi sfs-pod-example.yaml

- Example of attaching an SFS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sfs-pod-example                                # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sfs-pod-example
  template:
    metadata:
      labels:
        app: sfs-pod-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                              # Mount path
          name: pvc-sfs-example
      restartPolicy: Always
      volumes:
      - name: pvc-sfs-example
        persistentVolumeClaim:
          claimName: pvc-sfs-auto-example              # Mounting the PVC
```

  In this example:

  – **name**: name of the pod to be created.

  – **app**: name of the application running in the pod.

  – **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.

  – **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

- Example of attaching an SFS volume to a StatefulSet (PVCTemplate-based, dedicated volume):

  – **Example YAML file for clusters of version 1.15 and later:**

```
apiVersion: apps/v1
kind: StatefulSet
```

```
metadata:
  name: deploy-sfs-nfs-rw-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: 'https://console.prod-cloud-ocb.orange-business/swr/dockerimage/
nginx.png'
    description: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-sfs-nfs-rw-in
  template:
    metadata:
      labels:
        app: deploy-sfs-nfs-rw-in
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-sfs-nfs-rw-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: 8190a2a1692c46f284585c56fc0e2fb9
          resources: {}
          volumeMounts:
            - name: bs-nfs-rw-mountoptionpvc
              mountPath: /aaa
          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: bs-nfs-rw-mountoptionpvc
        namespace: default
        annotations: {}
        enable: true
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 11Gi
        storageClassName: csi-nas
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

– **Example YAML file for clusters of version 1.13 or earlier:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-sfs-nfs-rw-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: 'https://console.prod-cloud-ocb.orange-business/swr/dockerimage/
nginx.png'
    description: ''
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-sfs-nfs-rw-in
  template:
    metadata:
      labels:
        app: deploy-sfs-nfs-rw-in
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-sfs-nfs-rw-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: 8190a2a1692c46f284585c56fc0e2fb9
          resources: {}
          volumeMounts:
            - name: bs-nfs-rw-mountoptionpvc
              mountPath: /aaa
          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: bs-nfs-rw-mountoptionpvc
        creationTimestamp: null
        annotations:
          volume.beta.kubernetes.io/storage-class: nfs-rw
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxinfs
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 11Gi
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 2 Run the following command to create a pod:

kubectl create -f sfs-pod-example.yaml

After the creation is complete, log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage** > **SFS**. Then, click the PVC name. On the PVC details page, you can view the binding relationship between SFS and PVC.

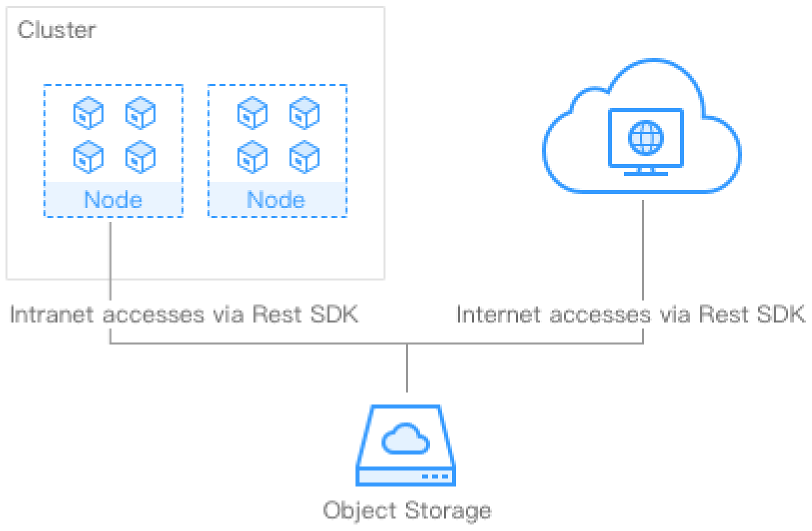
----End

11.5 Using OBS Buckets as Storage Volumes

11.5.1 Usage Description for OBS Volumes

CCE allows you to create OBS buckets and attach them to a path of the container. OBS applies to scenarios such as cloud workload, data analysis, content analysis, and hotspot objects.

Figure 11-3 Attaching OBS volumes to CCE


Cluster
Node
Node
Intranet accesses via Rest SDK
Internet accesses via Rest SDK
Object Storage

Object storage applies to the following scenarios:

- Standard storage

  Standard OBS buckets applies to scenarios where a large number of hotspot files or small-sized files need to be accessed frequently (multiple times per month on average) and data can be quickly obtained. For example, cloud applications, data analysis, content analysis, and hotspot objects.
- Infrequently accessed storage

  This type of OBS buckets applies to scenarios where data is not frequently accessed (less than 12 times per year on average) but fast access response is required. For example, static website hosting, backup/active archiving, storage resource pools or backup storage for cloud services.

Precautions

- You can use OBS buckets and OBS parallel file systems (preferred) in CCE by using OBS SDK or mounting them.
  - OBS SDK is recommended. For details, see the OBS SDK.
  - Parallel File System (PFS) is a high-performance file system provided by OBS, with access latency in milliseconds. PFS can support a bandwidth performance up to the TB/s level and supports millions of IOPS, outperforming OBS buckets. Therefore, you are advised to use OBS parallel file systems instead of OBS buckets (when mounting these objects) in the production environment.
- CCE can be mounted with OBS buckets and OBS parallel file systems (preferred) of third-party tenants.

Storage Class

Object storage offers three storage classes: Standard, Infrequent Access, and Archive, comprehensively meeting various requirements for storage performance and costs.

- The Standard storage class features low access latency and high throughput. It is therefore applicable to storing a large number of hot files (frequently accessed every month) or small files (less than 1 MB). The application scenarios include big data analytics, mobile apps, hot videos, and picture processing on social media.
- The Infrequent Access storage class is ideal for storing data that is semi-frequently accessed (less than 12 times a year), with requirements for quick response. The application scenarios include file synchronization or sharing, and enterprise-level backup. It provides the same durability, access latency, and throughput as the Standard storage class but at a lower cost. However, the Infrequent Access storage class has lower availability than the Standard storage class.
- The Archive storage class is suitable for archiving data that is rarely-accessed (averagely once a year). The application scenarios include data archiving and long-term data backup. The Archive storage class is secure and durable at an affordable low cost, which can be used to replace tape libraries. However, it may take hours to restore data from the Archive storage class.

Description

- With standard 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.
- Unlimited capacity and high performance (I/O read/write latency at 10 ms level) for web/mobile, backup/archiving, and big data/IoT scenarios.
- Applicable to Deployments, StatefulSets, and jobs in ReadOnlyMany scenarios based on OBS UI, tools, and SDKs.

11.5.2 Using OBS Volumes

Constraints

CCE allows you to attach OBS buckets as shared storage to nodes based on s3fs. OBS applies to the scenarios where file objects of different sizes are saved for once and read for multiple times. It does not apply to the scenario where files that have been saved need to be modified frequently. If you want higher access performance, you are advised to use OBS SDK. For details, see the **s3fs website**. An OBS bucket mounted using s3fs cannot provide the same performance or semantics as a local file system. The constraints are as follows:

- Random writes or appends to files require rewriting the entire file.
- Metadata operations such as listing directories have poor performance due to network latency.
- Eventual consistency can temporarily yield stale data.
- No atomic rename is performed on files or directories.
- No coordination is performed between multiple clients mounting the same bucket.

- Hard links are not supported.

Instructions

- With standard 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.
- Unlimited capacity and high performance (10 ms) for web/mobile, backup/archiving, and big data/IoT scenarios.
- Applicable to Deployments, StatefulSets, and jobs in ReadOnlyMany scenarios based on OBS UI, tools, and SDKs.

Preparation

To ensure reliable and stable OBS buckets for storage, ensure that access keys have been configured before you create OBS buckets.

The procedure for configuring the AK/SK is as follows:

1. Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.
2. On the OBS tab page, click **AK/SK** in the notice.
3. Click ..., select a key file, and click **Upload** to upload the key file.
4. Select the corresponding workload and click **Restart**.

NOTICE

When creating an OBS volume, you must use the AK/SK. If the key file is not uploaded, the pod will fail to be started or OBS data access will be abnormal due to the volume attaching failure.

- For details about how to obtain permanent access keys, see **Creating Access Keys (AK and SK)**.
- For details about how to manage access keys, see **Managing Access Keys**.

Creating an OBS Volume

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.

Step 2 Click the **OBS** tab and click **Create OBS Bucket**.

Step 3 Configure basic information, as shown in **Table 11-11**.

Table 11-11 Parameters for creating an OBS bucket

Parameter	Description
* PVC Name	Name of the new PVC, which is different from the volume name. The actual volume name is automatically generated when the PVC is created.
Cluster Name	Cluster of the object storage volume.
Namespace	Namespace of the OBS bucket. The default value is **default**.
* Instance type	Type of the storage instance created on OBS: ● **Parallel file system**: **If the cluster version is v1.15 or later and the everest add-on version is 1.0.2 or later**, parallel file systems that can be mounted by obsfs can be created. Parallel File System (PFS), provided by OBS, is a high-performance file system, with access latency in milliseconds. PFS can support a bandwidth performance up to the TB/s level and millions of IOPS, which makes it ideal for processing high-performance computing (HPC) workloads. ● **Object bucket**: A bucket is a container for storing objects in OBS. OBS provides flat storage in the form of buckets and objects. Unlike the conventional multi-layer directory structure of file systems, all objects in a bucket are stored at the same logical layer.
Storage Class	The following object storage classes are supported: ● **Standard**: applicable to scenarios where 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 to scenarios where data is not frequently accessed (less than 12 times per year on average) but requires fast access response. **NOTICE** Retrieval of data from Infrequent Access OBS is billed separately.
Storage Policy	**Private**: Only the bucket owner has full control over the bucket. Unauthorized users do not have permissions to access the bucket.
Access Mode	Access permissions of user applications on storage resources (PVs). ● **ReadWriteOnce** (RWO): Data can be read from and written into a volume, but the volume can be attached to only one node. ● **ReadWriteMany** (RWX): Data can be read from and written into a volume, and the volume can be attached to multiple nodes at the same time.

Parameter	Description
Storage Format	The default type is **CSI**. The container storage interface (CSI) is used to establish a set of standard storage management interfaces between Kubernetes and external storage systems to provide storage services for containers.

Step 4 Click **Create**.

After the OBS bucket is successfully created, it is displayed in the OBS bucket list. Click the PVC name to view detailed information about the OBS bucket.

----End

Using OBS Volumes

Step 1 Create a workload by referring to **Creating a Deployment** or **Creating a StatefulSet**. After you have added a container, choose **Data Storage** > **Cloud Volume**, and then click **Add Cloud Volume**.

Step 2 Set **Type** to **OBS**.

Table 11-12 OBS bucket parameters

Parameter	Description
Type	OBS buckets: Standard OBS buckets and infrequent access OBS buckets are supported. OBS buckets apply to scenarios such as big data analysis, native cloud application data, static website hosting, and backup/active archiving.
Allocation Mode	
Manual	**Name**: Select a created OBS volume. You need to create an OBS volume in advance.
Automatic	Type of the storage instance created on OBS: ● **Parallel file system**: **If the cluster version is v1.15 or later and the everest add-on version is 1.0.2 or later**, parallel file systems that can be mounted by obsfs can be created. **Storage Format**: The default value is **CSI**. ● **Object bucket**: A bucket is a container for storing objects in OBS. **Sub-Type**: Select **Standard** or **Infrequent access**. **Storage Format**: The default value is **CSI**.

Parameter	Description
Add Container Path	Configure the following parameters: 1. **Container Path**: Enter the path of the container, for example, **/tmp**. The container path must not be a system directory, such as **/** and **/var/run**. Otherwise, an exception occurs. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 2. Set permissions. – **Read-only**: You can only read the data volumes mounted to the path. – **Read/Write**: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss. 3. Click **Add Container Path** to add multiple settings. Then, click **OK**.

Step 3 Click **OK**.

----End

Importing an OBS Volume

CCE allows you to import an existing OBS volume.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **OBS** tab page, click **Import**.

Step 2 Select one or more OBS volumes that you want to import.

Step 3 Select the target cluster and namespace.

Step 4 Click **OK**.

----End

Unbinding an OBS Volume

When an OBS volume is successfully created, it is automatically bound to the current cluster. Other clusters can also use the OBS volume. When the OBS volume is unbound from the cluster, it is still available to other clusters.

If the OBS volume has been mounted to a workload, the OBS volume cannot be unbound from the cluster.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the object storage volume list, click **Unbind** next to the target OBS volume.

Step 2 In the dialog box displayed, click **OK**.

----End

Related Operations

After an OBS volume is created, you can perform the operation described in **Table 11-13**.

Table 11-13 Other operations

Operation	Description
Deleting an OBS bucket	1. Select the OBS bucket to be deleted and click **Delete** in the **Operation** column. 2. Follow the prompts to delete the object.

11.5.3 Automatically Creating an OBS Volume Using kubectl

Scenario

During the use of an OBS EVS disk, the expected OBS bucket can be automatically created and mounted. Currently, standard OBS buckets and infrequent access OBS buckets are supported, corresponding to obs-standard and obs-standard-ia respectively.

Procedure

Step 1 Configure the kubectl commands. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Run the following commands to configure the **pvc-obs-auto-example.yaml** file, which is used to create a PVC.

touch pvc-obs-auto-example.yaml

vi pvc-obs-auto-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
  name: obs-warm-provision-pvc
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
```

In this example:

- **everest.io/obs-volume-type**: OBS bucket type. Currently, standard (STANDARD) and infrequent access (WARM) buckets are supported.
- **name**: name of the PVC to be created.
- **storage**: storage capacity in the unit of Gi. For OBS buckets, this field is used only for verification (cannot be empty or 0). Its value is fixed to **1**, but the value setting does not take effect for OBS buckets.
- **csi.storage.k8s.io/fsType**: file type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to **obsfs**.

- **Example for clusters of version 1.13 or earlier:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
  name: pvc-obs-auto-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: OBS bucket type. Currently, **obs-standard** and **obs-standard-ia** are supported.
- **name**: name of the PVC to be created.
- **storage**: storage capacity in the unit of Gi. For OBS buckets, this field is used only for verification (cannot be empty or 0). Its value is fixed to **1**, but the value setting does not take effect for OBS buckets.

Step 3 Run the following command to create a PVC.

kubectl create -f pvc-obs-auto-example.yaml

After the command is executed, an object storage bucket is created in the VPC to which the cluster belongs. You can click the bucket name in **Storage** > **OBS** to view the bucket or view it on the OBS console.

----End

Procedure

CCE allows you to use an existing OBS bucket to create a PersistentVolume (PV). You can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Step 1 Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a YAML file for creating the PersistentVolume. Assume that the file name is **pv-obs-example.yaml**.

touch pv-obs-example.yaml

vi pv-obs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    fsType: s3fs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: cn-north-7
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

  In this example:

  – **driver**: storage driver used to mount the volume. Set the driver to **obs.csi.everest.io** for the OBS volume.

  – **everest.io/obs-volume-type**: volume type. Value options include **STANDARD** (standard bucket) and **WARM** (infrequent access bucket).

  – **everest.io/region**: region where the OBS bucket is located.

  – **volumeHandle**: OBS bucket name.

    To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.

  – **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

  – **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-obs**.

  – **fsType**: file type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to **obsfs**.

- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiobs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

```
 flexVolume:
   driver: huawei.com/fuxiobs
   fsType: obs
   options:
     fsType: obs
     region: eu-west-0a
     storage_class: STANDARD
     volumeID: test-obs
 persistentVolumeReclaimPolicy: Delete
 storageClassName: obs-standard
```

In this example:

– **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
– **storage_class**: storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).
– **region**: region where the object storage is located.
– **volumeID**: OBS bucket name.

  To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.
– **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.
– **storageClassName**: storage class supported by OBS, including **obs-standard** (standard) and **obs-standard-ia** (infrequent access).

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      kubernetes.io/namespace: default
      region: eu-west-0a
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

In this example:

– **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
– **storage_class**: storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).
– **region**: region where the object storage is located.
– **volumeID**: OBS bucket name.

  To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.

- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.
- **storageClassName**: storage class supported by OBS, including **obs-standard** (standard) and **obs-standard-ia** (infrequent access).

Step 4 Create a PV.

kubectl create -f pv-obs-example.yaml

Step 5 Create a YAML file to create a PVC. Assume that the file name is **pvc-obs-example.yaml**.

touch pvc-obs-example.yaml

vi pvc-obs-example.yaml

- **Example YAML file for clusters of v1.15 and later**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

In this example:

- **volumeName**: name of the PV.
- **storage**: storage capacity in the unit of Gi. The value is fixed to **1Gi**, which must be the same as the storage size of the existing PV.
- **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-obs**.
- **everest.io/obs-volume-type**: OBS volume type, which can be **STANDARD** (standard bucket) and **WARM** (infrequent access bucket).
- **csi.storage.k8s.io/fstype**: file type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to **obsfs**.

- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class supported by the object storage, including **obs-standard** and **obs-standard-ia**.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxiobs**.
- **volumeName**: name of the PV.
- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
  volumeNamespace: default
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class supported by the object storage, including **obs-standard** and **obs-standard-ia**.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxiobs**.
- **volumeName**: name of the PV.
- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

Step 6 Create a PVC.

kubectl create -f pvc-obs-example.yaml

----End

11.5.4 Creating a PVC by Using an Existing OBS Bucket

Scenario

CCE allows you to use an existing OBS bucket to create a PersistentVolume (PV). You can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Procedure

CCE allows you to use an existing OBS bucket to create a PersistentVolume (PV). You can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Step 1 Log in to the OBS console, create an OBS bucket, and record the bucket name and storage class.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a YAML file for creating the PersistentVolume. Assume that the file name is **pv-obs-example.yaml**.

touch pv-obs-example.yaml

vi pv-obs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  csi:
    driver: obs.csi.everest.io
    fsType: s3fs
    volumeAttributes:
      everest.io/obs-volume-type: STANDARD
      everest.io/region: cn-north-7
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: obs-normal-static-pv
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-obs
```

  In this example:

  – **driver**: storage driver used to mount the volume. Set the driver to **obs.csi.everest.io** for the OBS volume.

  – **everest.io/obs-volume-type**: volume type. Value options include **STANDARD** (standard bucket) and **WARM** (infrequent access bucket).

  – **everest.io/region**: region where the OBS bucket is located.

  – **volumeHandle**: OBS bucket name.

    To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.

  – **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

  – **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-obs**.

  – **fsType**: file type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to **obsfs**.

- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiobs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      region: eu-west-0a
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

In this example:

– **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
– **storage_class**: storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).
– **region**: region where the object storage is located.
– **volumeID**: OBS bucket name.

  To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.
– **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.
– **storageClassName**: storage class supported by OBS, including **obs-standard** (standard) and **obs-standard-ia** (infrequent access).

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  flexVolume:
    driver: huawei.com/fuxiobs
    fsType: obs
    options:
      fsType: obs
      kubernetes.io/namespace: default
      region: eu-west-0a
      storage_class: STANDARD
      volumeID: test-obs
  persistentVolumeReclaimPolicy: Delete
  storageClassName: obs-standard
```

In this example:

– **driver**: storage driver used to mount the volume. Set the driver to **huawei.com/fuxiobs** for the OBS volume.
– **storage_class**: storage class, including **STANDARD** (standard bucket) and **STANDARD_IA** (infrequent access bucket).

- **region**: region where the object storage is located.
- **volumeID**: OBS bucket name.

  To obtain the value, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **OBS** tab page, and copy the PV name on the **PV Details** tab page.
- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.
- **storageClassName**: storage class supported by OBS, including **obs-standard** (standard) and **obs-standard-ia** (infrequent access).

Step 4 Create a PV.

kubectl create -f pv-obs-example.yaml

Step 5 Create a YAML file to create a PVC. Assume that the file name is **pvc-obs-example.yaml**.

touch pvc-obs-example.yaml

vi pvc-obs-example.yaml

- **Example YAML file for clusters of v1.15 and later**:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
    everest.io/obs-volume-type: STANDARD
    csi.storage.k8s.io/fstype: s3fs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-obs
  volumeName: pv-obs-example
```

  In this example:
  - **volumeName**: name of the PV.
  - **storage**: storage capacity in the unit of Gi. The value is fixed at **1Gi**, which must be the same as the storage size of the existing PV.
  - **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-obs**.
  - **everest.io/obs-volume-type**: OBS volume type, which can be **STANDARD** (standard bucket) and **WARM** (infrequent access bucket).
  - **csi.storage.k8s.io/fstype**: file type. The value can be **obsfs** or **s3fs**. If the value is **s3fs**, an OBS bucket is created and mounted using s3fs. If the value is **obsfs**, an OBS parallel file system is created and mounted using obsfs. You are advised to set this field to **obsfs**.
- **Example YAML file for clusters of versions 1.11 and 1.13:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class supported by the object storage, including **obs-standard** and **obs-standard-ia**.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxiobs**.
- **volumeName**: name of the PV.
- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

- **Example YAML file for clusters of version 1.9:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: obs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs
  name: pvc-obs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: pv-obs-example
  volumeNamespace: default
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: storage class supported by the object storage, including **obs-standard** and **obs-standard-ia**.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxiobs**.
- **volumeName**: name of the PV.
- **storage**: storage capacity in the unit of Gi. Set this parameter to the fixed value **1Gi**.

Step 6 Create a PVC.

kubectl create -f pvc-obs-example.yaml

----End

11.5.5 Using an Existing PVC to Create a StatefulSet

Scenario

CCE allows you to use an existing OBS volume (PersistentVolumeClaim) to create a workload (StatefulSet).

Procedure

Step 1 Create an object storage volume by referring to **Creating an OBS Volume** and obtain the PVC name.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a PVC file for creating the workload. Assume that the file name is **obs-statefulset-example.yaml**.

touch obs-statefulset-example.yaml

vi obs-statefulset-example.yaml

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: obs-statefulset-example
  namespace: default
spec:
  podManagementPolicy: OrderedReady
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: obs-statefulset-example
  serviceName: qwqq
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: "true"
      creationTimestamp: null
      labels:
        app: obs-statefulset-example
    spec:
      affinity: {}
      containers:
      - env:
        - name: PAAS_APP_NAME
          value: obs-statefulset-example
        - name: PAAS_NAMESPACE
          value: default
        - name: PAAS_PROJECT_ID
          value: b7bb7d77a2974a8fa8985cbfb63f23c0
        image: nginx:latest
        imagePullPolicy: Always
        name: container-0
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /tmp
          name: pvc-obs-example
      dnsPolicy: ClusterFirst
      imagePullSecrets:
      - name: default-secret
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - name: pvc-obs-example
          persistentVolumeClaim:
            claimName: cce-obs-demo
      tolerations:
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
        tolerationSeconds: 300
      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists
        tolerationSeconds: 300
  updateStrategy:
    type: RollingUpdate
```

In this example:

- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.
- **replicas**: number of pods.
- **name**: name of the new workload.
- **image**: image used by the workload.
- **mountPath**: mount path of a container.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **claimName**: name of an existing PVC.

Step 4 Create a StatefulSet.

kubectl create -f obs-statefulset-example.yaml

----End

11.5.6 Creating a Pod with an OBS Volume Attached Using kubectl

Scenario

After an OBS volume is created or imported to CCE, you can attach the volume to a workload.

Procedure

Step 1 Run the following commands to configure the **obs-pod-example.yaml** file, which is used to create a pod.

touch obs-pod-example.yaml

vi obs-pod-example.yaml

- Example of attaching an OBS volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: obs-pod-example                    # Workload name
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: obs-pod-example
  template:
    metadata:
      labels:
        app: obs-pod-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp                        # Mount path
          name: pvc-obs-example
      restartPolicy: Always
      volumes:
      - name: pvc-obs-example
        persistentVolumeClaim:
          claimName: pvc-obs-auto-example       # PVC Name
```

In this example:

- **name**: name of the pod to be created.
- **app**: name of the application running in the pod.
- **mountPath**: mount path of a container.
- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

● Example of attaching an OBS volume to a StatefulSet (PVCTemplate-based, dedicated volume):

  – **Example for clusters of version 1.15 and later:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-obs-standard-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: https://console.prod-cloud-ocb.orange-business.com/swr/
dockerimage/nginx.png
    description: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-obs-standard-in
  template:
    metadata:
      labels:
        app: deploy-obs-standard-in
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-obs-standard-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: a2cd8e998dca42e98a41f596c636dbda
          resources: {}
          volumeMounts:
            - name: obs-bs-standard-mountoptionpvc
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: obs-bs-standard-mountoptionpvc
        namespace: default
        annotations:
          everest.io/obs-volume-type: STANDARD
      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-obs
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

– **Example for clusters of version 1.13 or earlier:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: deploy-obs-standard-in
  namespace: default
  generation: 1
  labels:
    appgroup: ''
  annotations:
    container.io/container-0: https://console.prod-cloud-ocb.orange-business.com/swr/
dockerimage/nginx.png
    description: ''
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-obs-standard-in
  template:
    metadata:
      labels:
        app: deploy-obs-standard-in
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
    spec:
      containers:
        - name: container-0
          image: 'nginx:1.12-alpine-perl'
          env:
            - name: PAAS_APP_NAME
              value: deploy-obs-standard-in
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: a2cd8e998dca42e98a41f596c636dbda
          resources: {}
          volumeMounts:
            - name: obs-bs-standard-mountoptionpvc
              mountPath: /tmp
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      imagePullSecrets:
        - name: default-secret
      affinity: {}
      schedulerName: default-scheduler
  volumeClaimTemplates:
    - metadata:
        name: obs-bs-standard-mountoptionpvc
        annotations:
          volume.beta.kubernetes.io/storage-class: obs-standard
          volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiobs

      spec:
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
  serviceName: wwww
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: RollingUpdate
  revisionHistoryLimit: 10
```

In this example:

- **name**: name of the created workload.
- **image**: image of the workload.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.volumeClaimTemplates.metadata.name** must be consistent because they have a mapping relationship.

Step 2 Run the following command to create a pod:

kubectl create -f obs-pod-example.yaml

After the creation is complete, choose **Storage** > **OBS** on the CCE console and click the PVC name. On the PVC details page, you can view the binding relationship between the OBS service and the PVC.

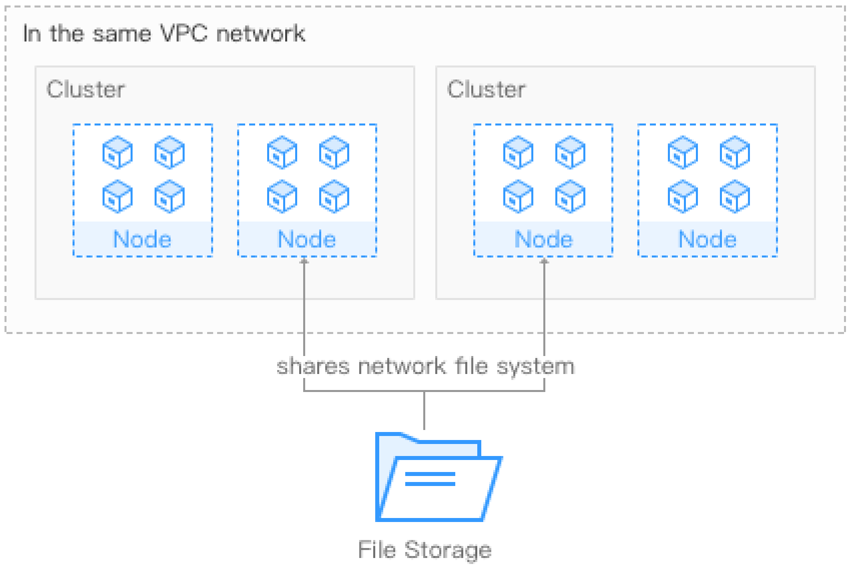
----End

11.6 Using SFS Turbo File Systems as Storage Volumes

11.6.1 Usage Description for SFS Turbo Volumes

SFS Turbo file systems are fast, on-demand, and scalable, which are suitable for DevOps, containerized microservices, and enterprise office applications.

Figure 11-4 Mounting SFS Turbo file systems to CCE



Instructions

- Complying with standard file protocols, file systems can be mounted 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: Data must be accessed in the internal network of the data center.
- Existing IaaS services in the cloud are used to build exclusive cloud file storage, providing data isolation and IOPS performance assurance for users. SFS Turbo is mainly used in DevOps, containerized microservices, and enterprise office scenarios.
- This mode is applicable to Deployments, StatefulSets, and jobs in ReadWriteMany scenarios.

11.6.2 Using SFS Turbo Volumes

Constraints

Currently, SFS Turbo volumes cannot be directly created on CCE. You need to create an SFS Turbo file system on the SFS Turbo console before using it as a storage volume on CCE.

Importing an SFS Turbo Volume

CCE allows you to import existing SFS Turbo volumes.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. On the **SFS Turbo** tab page, click **Import**.

Step 2 Select one or more SFS Turbo file systems that you want to import.

Step 3 Select the cluster and namespace to which you want to attach and mount the file system.

Step 4 Click **OK**.

----End

Using SFS Turbo Volumes

Step 1 Create a workload by referring to **Creating a Deployment** or **Creating a StatefulSet**. After you have added a container, choose **Data Storage** > **Cloud Volume**, and then click **Add Cloud Volume**.

Step 2 Set the storage type to **SFS Turbo**.

Table 11-14 Parameters for configuring an SFS Turbo volume

Parameter	Description
Type	**SFS Turbo**: applicable to DevOps, containerized microservices, and enterprise office applications.
Allocation Mode	
Manual	Select an existing SFS Turbo file system. You need to import SFS Turbo file systems in advance. For details, see **Importing an SFS Turbo Volume**.

Parameter	Description
Add Container Path	Configure the following parameters: 1. **subPath**: Enter the subpath of the file storage, for example, **/tmp**. In Kubernetes, the subpath to which the data volume is mounted refers to the subpath of the referenced volume rather than its root. If this parameter is not specified, the root path of the data volume is used by default. Currently, only file storage is supported. The value must be a relative path and cannot start with a slash (/) or ../. 2. **Container Path**: Enter the path of the container, for example, **/tmp**. The container path must not be a system directory, such as **/** and **/var/run**. Otherwise, an exception occurs. You are advised to mount the volume to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload. **NOTICE** If the volume is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container; otherwise, high-risk files on the host machine may be damaged. 3. Set permissions. – **Read-only**: You can only read the data volumes mounted to the path. – **Read/Write**: You can modify the data volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause data loss.

Step 3 Click **OK**.

----End

Unbinding an SFS Turbo Volume

When an SFS Turbo volume is successfully imported to a cluster, the volume is bound to the cluster. Other clusters can use the volume. When the volume is unbound from the cluster, it is still available to other clusters.

If the SFS Turbo volume has been attached to a workload, it cannot be unbound from the cluster.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**. In the SFS Turbo volume list, click **Unbind** next to the target volume.

Step 2 In the dialog box displayed, click **OK**.

----End

11.6.3 Creating a PVC by Using an Existing SFS Turbo File System

Scenario

CCE allows you to use an existing SFS Turbo file system to create a PersistentVolume (PV). After the creation is successful, you can create a PersistentVolumeClaim (PVC) and bind it to the PV.

Procedure

Step 1 Log in to the SFS console, create a file system, and record the file system ID, shared path, and capacity.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Kubectl Usage Guide**.

Step 3 Create a YAML file for creating the PV. Assume that the file name is **pv-efs-example.yaml**.

touch pv-efs-example.yaml

vi pv-efs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: everest-csi-provisioner
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: sfsturbo.csi.everest.io
    fsType: nfs
    volumeAttributes:
      everest.io/share-export-location: 192.168.0.169:/
      storage.kubernetes.io/csiProvisionerIdentity: everest-csi-provisioner
    volumeHandle: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: csi-sfsturbo
```

  In this example:

  – **driver**: storage driver for the mounting. Set it to **sfsturbo.csi.everest.io**.
  – **everest.io/share-export-location**: shared path of the SFS Turbo file system.
  – **volumeHandle**: SFS Turbo file system ID.

    To obtain the ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **SFS** tab page, and copy the **PVC UID** on the PVC details page.
  – **storage**: file system size.
  – **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-sfsturbo** for SFS Turbo file systems.

- **Example YAML file for clusters of version 1.13 or earlier:**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-efs-example
  annotations:
    pv.kubernetes.io/provisioned-by: flexvolume-huawei.com/fuxiefs
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 100Gi
  flexVolume:
    driver: huawei.com/fuxiefs
    fsType: efs
    options:
      deviceMountPath: 192.168.0.169:/
      fsType: efs
      volumeID: 8962a2a2-a583-4b7f-bb74-fe76712d8414
  persistentVolumeReclaimPolicy: Delete
  storageClassName: efs-standard
```

  In this example:

  – **driver**: storage driver for the mounting. Set it to **huawei.com/fuxiefs**.
  – **deviceMountPath**: shared path for storing files.
  – **volumeID**: file system ID.
    To obtain the ID, log in to the CCE console, choose **Resource Management** > **Storage**, click the PVC name on the **SFS** tab page, and copy the **PVC UID** on the PVC details page.
  – **storage**: file system size.
  – **storageClassName**: Volume type supported by the SFS Turbo file system. **efs-standard** and **efs-performance** are supported. Currently, SFS Turbo does not support dynamic creation; therefore, this parameter is not used for now.

NOTE

The VPC to which the SFS Turbo file system belongs must be the same as the VPC of the ECS VM planned for the workload. Ports 111, 445, 2049, 2051, and 20048 can be accessed.

Step 4 Create a PV.

kubectl create -f pv-efs-example.yaml

Step 5 Create a YAML file to create a PVC. Assume that the file name is **pvc-efs-example.yaml**.

touch pvc-efs-example.yaml

vi pvc-efs-example.yaml

- **Example YAML file for clusters of v1.15 and later:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: everest-csi-provisioner
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: csi-sfsturbo
  volumeName: pv-efs-example
```

In this example:

- **storageClassName**: name of the Kubernetes storage class. Set this field to **csi-sfsturbo**.
- **storage**: storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

- **Example YAML file for clusters of version 1.13 or earlier:**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: efs-standard
    volume.beta.kubernetes.io/storage-provisioner: flexvolume-huawei.com/fuxiefs
  name: pvc-efs-example
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  volumeName: pv-efs-example
```

In this example:

- **volume.beta.kubernetes.io/storage-class**: read/write mode supported by the file system. The value can be **efs-standard** or **efs-performance**. The value must be the same as that of the existing PV.
- **volume.beta.kubernetes.io/storage-provisioner**: must be set to **flexvolume-huawei.com/fuxiefs**.
- **storage**: storage capacity, in the unit of Gi. The value must be the same as the storage size of the existing PV.
- **volumeName**: name of the PV.

Step 6 Create a PVC.

kubectl create -f pvc-efs-example.yaml

----End

11.6.4 Using an Existing PVC to Create a StatefulSet

Scenario

CCE allows you to use an existing SFS Turbo file system to create a workload (StatefulSet).

Procedure

Step 1 Create an SFS Turbo file system and record the name of the file system.

Step 2 Run kubectl commands to connect to the cluster. For details, see **Connecting to a Cluster Through kubectl**.

Step 3 Create a YAML file for creating the workload. Assume that the file name is **efs-statefulset-example.yaml**.

touch efs-statefulset-example.yaml

vi efs-statefulset-example.yaml

Configuration example:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: efs-statefulset-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-statefulset-example
  template:
    metadata:
      annotations:
        metrics.alpha.kubernetes.io/custom-endpoints: '[{"api":"","path":"","port":"","names":""}]'
        pod.alpha.kubernetes.io/initialized: 'true'
      labels:
        app: efs-statefulset-example
    spec:
      containers:
        - image: 'nginx:1.0.0'
          name: container-0
          resources:
            requests: {}
            limits: {}
          env:
            - name: PAAS_APP_NAME
              value: efs-statefulset-example
            - name: PAAS_NAMESPACE
              value: default
            - name: PAAS_PROJECT_ID
              value: b18296881cc34f929baa8b9e95abf88b
          volumeMounts:
            - name: efs-statefulset-example
              mountPath: /tmp
              readOnly: false
              subPath: ''
      imagePullSecrets:
        - name: default-secret
      terminationGracePeriodSeconds: 30
      volumes:
        - persistentVolumeClaim:
            claimName: cce-efs-import-jnr481gm-3y5o
          name: efs-statefulset-example
      affinity: {}
      tolerations:
        - key: node.kubernetes.io/not-ready
          operator: Exists
          effect: NoExecute
          tolerationSeconds: 300
        - key: node.kubernetes.io/unreachable
          operator: Exists
          effect: NoExecute
          tolerationSeconds: 300
  podManagementPolicy: OrderedReady
  serviceName: test
  updateStrategy:
    type: RollingUpdate
```

In this example:

- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.
- **replicas**: number of pods.
- **name**: name of the new workload.
- **image**: image used by the workload.
- **mountPath**: mount path of a container.
- **serviceName**: Service corresponding to the workload. For details about how to create a Service, see **Creating a StatefulSet**.
- **claimName**: name of an existing PVC.

Step 4 Create a StatefulSet.

kubectl create -f efs-statefulset-example.yaml

----End

11.6.5 Attaching an SFS Turbo Volume Using kubectl

Scenario

After an SFS Turbo volume is created or imported to CCE, you can attach the volume to a workload.

Procedure

Step 1 Run the following commands to configure the **efs-pod-example.yaml** file, which is used to create a pod:

touch efs-pod-example.yaml

vi efs-pod-example.yaml

Example of attaching an SFS Turbo volume to a Deployment (PVC-based, shared volume):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: efs-pod-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: efs-pod-example
  template:
    metadata:
      labels:
        app: efs-pod-example
    spec:
      containers:
      - image: nginx
        name: container-0
        volumeMounts:
        - mountPath: /tmp
          name: pvc-efs-example
      restartPolicy: Always
      volumes:
```

```
    - name: pvc-efs-example
      persistentVolumeClaim:
        claimName: pvc-sfs-auto-example
```

In this example:

- **name**: name of the pod to be created.
- **app**: name of the application running in the pod.
- **mountPath**: mount path of the container. In this example, the mount path is **/tmp**.
- **spec.template.spec.containers.volumeMounts.name** and **spec.template.spec.volumes.name** must be consistent because they have a mapping relationship.

Step 2 Run the following command to create a pod:

kubectl create -f efs-pod-example.yaml

After the creation is complete, choose **Storage** > **SFS Turbo** on the CCE console and click the PVC name. On the PVC details page, you can view the binding relationship between SFS Turbo and PVC.

----End

11.7 Snapshot and Backup

CCE works with EVS to provide the snapshot function. A snapshot is a complete copy or image of EVS disk data at a certain point of time, which is of great help to data DR.

You can create snapshots to rapidly save the disk data at specified time points. In addition, you can use snapshots to create new disks so that the created disks will contain the snapshot data in the beginning.

Precautions

- The snapshot function is available only for clusters of **v1.15 or later** and requires the CSI-based **Everest (System Resource Add-on, Mandatory)**.
- The subtype (common I/O, high I/O, or ultra-high I/O), disk mode (SCSI or VBD), data encryption, sharing status, and capacity of an EVS disk created from a snapshot must be the same as those of the disk associated with the snapshot. These attributes cannot be modified after being queried or set.

Scenarios

The snapshot feature helps address your following needs:

- **Routine data backup**

  You can create snapshots for EVS disks regularly and use snapshots to recover your data in case that data loss or data inconsistency occurred due to misoperations, viruses, or attacks.
- **Rapid data restoration**

  You can create a snapshot or multiple snapshots before an OS change, application software upgrade, or a service data migration. If an exception occurs during the upgrade or migration, service data can be rapidly restored to the time point when the snapshot was created.

For example, a fault occurred on system disk A of ECS A, and therefore ECS A cannot be started. Because system disk A is already faulty, the data on system disk A cannot be restored by rolling back snapshots. In this case, you can use an existing snapshot of system disk A to create EVS disk B and attach it to ECS B that is running properly. Then, ECS B can read data from system disk A using EVS disk B.

NOTE

The snapshot capability provided by CCE is the same as the CSI snapshot function provided by the Kubernetes community. EVS disks can be created only based on snapshots, and snapshots cannot be rolled back to source EVS disks.

- **Rapid deployment of multiple services**

  You can use a snapshot to create multiple EVS disks containing the same initial data, and these disks can be used as data resources for various services, for example, data mining, report query, and development and testing. This method protects the initial data and creates disks rapidly, meeting the diversified service data requirements.

Operation Overview

Step 1 You can **create a snapshot** to rapidly save the disk data at specified time points.

Step 2 You can **create an EVS disk from a snapshot** so that the EVS disk contains the snapshot data in the beginning. When data is lost, the snapshot can be used to restore the data to point in time when the snapshot was taken.

Step 3 When a snapshot is no longer needed, **delete it** to release the virtual resources.

----End

Creating a Snapshot

You can create EVS snapshots to save the disk data at specific time points.

NOTE

Snapshots can be created only for EVS disks that are available or in use. A maximum of seven snapshots can be created for a single EVS disk. For details about how to create an EVS disk, see **Using EVS Disks as Storage Volumes**.

Snapshot data of encrypted disks is stored encrypted, and that of non-encrypted disks is stored non-encrypted.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.

Step 2 On the **Snapshot and backup** tab page, click **Create Snapshot**.

Step 3 On the **Create Snapshot** page displayed, configure parameters listed in **Table 11-15**.

Table 11-15 Parameters for creating a snapshot

Parameter	Description
Snapshot Name	Name of the snapshot to be created. The name starts with **cce-disksnap-** by default, for example, **cce-disksnap-01**.
Cluster Name	Cluster where the snapshot is deployed.
Namespace	Namespace with which the snapshot is associated.
Select Storage	Select the EVS disk for which the snapshot is to be created. The disk must be available or in use. For details about how to create a disk, see **Using EVS Disks as Storage Volumes**.

Step 4 Click **Create Now**.

Step 5 Click **Back to Snapshot List**. When the snapshot status changes to Normal, the snapshot is successfully created.

Click ^ on the left of the snapshot name to view the snapshot status and details.

----End

Creating a Disk from a Snapshot

On the snapshot list page, you can select a snapshot to create an EVS disk.

In this mode, pay attention to the following constraints:

- The disk type, disk mode, and encryption setting of the created disk are consistent with those of the snapshot's source EVS disk.
- A snapshot can be used to create a maximum of 128 EVS disks.
- Batch disk creation is not supported, and the quantity parameter must be set to **1**.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.

Step 2 In the snapshot list of the **Snapshot and backup** tab page, locate the target snapshot and click **Create Data Volume** in the **Operation** column.

Step 3 Set the parameters of the EVS disk. For details, see **Using EVS Disks as Storage Volumes**.

When you use a snapshot to create an EVS disk, the capacity must be greater than or equal to the snapshot size. In the condition that you do not specify the disk capacity, if the snapshot size is smaller than 10 GB, the default capacity 10 GB will be used as the disk capacity; if the snapshot size is greater than 10 GB, the disk capacity will be consistent with the snapshot size.

Step 4 Click **Create Now**, and click **Submit**.

Step 5 On the page indicating that the task is submitted successfully, click **Go to Storage**. On the **EVS** tab page, view the status of the EVS disk.

When the EVS disk status changes to **Available**, the EVS disk is created successfully.

----End

Deleting a Snapshot

If a snapshot is no longer used, you can release the virtual resources by deleting the snapshot from the system. Snapshot deletion has the following constraints:

- A snapshot can be deleted only when its status is **Available** or **Error**.
- If an EVS disk is deleted, all the snapshots created for this disk will also be deleted.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Storage**.

Step 2 In the snapshot list of the **Snapshot and backup** tab page, locate the target snapshot and click **Delete** in the **Operation** column.

Step 3 (Optional) If multiple snapshots are to be deleted, select ☐ in front of each snapshot and click **Delete** in the upper area of the list.

Step 4 In the dialog box displayed, enter **DELETE**, and click **Yes**.

----End

Reference Documents

Volume Snapshots

12 Chart Management

12.1 Introduction

CCE uses Kubernetes Helm, a package manager, to simplify deployment and management of packages (also called charts). A chart is a collection of files that describe a related set of Kubernetes resources. The use of charts handles all the complexity in Kubernetes resource installation and management, making it possible to achieve unified resource scheduling and management.

NOTE

Helm is a tool for packaging Kubernetes applications. For more information, see **Helm documentation**.

Custom charts simplify workload deployment.

This section describes how to create a workload using a customized chart. You can use multiple methods to create an orchestration template on the CCE console.

Constraints

- A user can upload a maximum of 20 charts.
- A chart with multiple versions consumes the same amount of portion of chart quota.

12.2 Preparing a Template Package

You can prepare a template package using one of the following methods:

- **Customizing a Template Package**
- **Using a Kubernetes Official Template Package**

NOTICE

If the created workload requires the EVS disk and ELB functions, you need to modify the template package. For details, see **Using EVS Disks** and **Using ELB**.

Customizing a Template Package

Step 1 Customize the content of a template package as required.

For details about how to create a template package, see **https://github.com/kubernetes/helm/blob/master/docs/charts.md**.

Step 2 Set the template package directory structure and name the template package based on the requirements defined in **Template Package Specifications**.

----End

Using a Kubernetes Official Template Package

Step 1 Access **https://github.com/kubernetes/charts** to obtain the required community template package.

Step 2 Log in to a Linux machine.

Step 3 Upload the template package obtained in **Step 1**.

Step 4 Run the following command to compress the template package.

- If the Helm client is not installed on the Linux machine, run the following command:

  tar pzcf {name}-{version}.tgz {name}/

  In the preceding command,

  {name} indicates the actual name of the template package.

  {version} indicates the actual version of the template package.

  NOTICE

  The values of {name} and {version} must be the same as the values of name and version in the Chart.yaml file in the template package.

- If the Helm client is installed on the Linux machine, run the following command:

  helm package {name}/

  In the preceding command, **{name}** indicates the actual name of the template package.

Step 5 Set the template package directory structure and name the template package based on the requirements defined in **Template Package Specifications**.

----End

Template Package Specifications

The following uses the Redis as an example. Prepare the Redis template package according to the template package specifications.

- **Naming Requirement**

The template package is named in the following format: Workload name-main version number.minor version number.revision number.tgz, for example, **redis-0.4.2.tgz**, **redis-0.4.2-beta.tgz**, and **redis-0.4.2-alpha.1.tgz**.

NOTE

The version number of the template package must comply with the **semantic versioning** rules.

- The main version number and minor version number are mandatory, and the revision number is optional.
- The version number contains a maximum of 64 characters.
- The values of the main and minor version numbers are integers. They must be ≥0 and ≤99.
- The revision number consists of digits, uppercase letters, lowercase letters, and hyphens (-), that is, [0-9A-Za-z-].

- **Directory Structure**

  The directory structure of a template package is as follows:

```
redis/
  templates/
  values.yaml
  README.md
  Chart.yaml
  .helmignore
```

As shown in **Table 12-1**, the parameters marked with * are mandatory.

Table 12-1 Parameters of the directory structure of a template package

Parameter	Description
* templates	All templates
* values.yaml	Configuration parameters essential to describe the template. **NOTICE** When setting parameters in the template file, ensure that the image address is the same as the corresponding image address in the container image repository. Otherwise, an exception occurs when you create a workload, and the system displays a message indicating that the image fails to be pulled. To obtain the image address, perform the following operations: Log in to the CCE console. In the navigation pane, choose **Image Repository** to access the SWR console. On the SWR console, choose **My Images**. On the **Private Images** tab page, click the name of an uploaded image. On the **Image Version** tab page, you can view the image address in the **Intranet Image Address** column.
README.md	A markdown file, including: ● The workload or services provided by Chart. ● Prerequisites for running Chart. ● Configurations in the values.yaml file. ● Information about Chart installation and configuration.
* Chart.yaml	Basic information about the template.

Parameter	Description
.helmignore	Files or data that will be ignored during workload installation.

12.3 Uploading a Chart Package

Upload a chart to **Charts** > **Uploaded Charts** for subsequent workload creation.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Charts** > **Uploaded Charts** and click **Upload Chart**.

Step 2 Click ⋯, select the chart package to be uploaded, and click **Upload**.

----End

Follow-up Procedure

After a chart is created, you can perform operations listed in **Table 12-2** on the **Uploaded Charts** page.

Table 12-2 Other operations

Operation	Description
Installing a chart	Click **Install** to install the chart for creating workloads. For details, see **Creating a Workload Using a Chart**.
Updating a chart	The chart content will be updated while the chart version remains unchanged. The procedure is similar to that of uploading a chart.
Downloading a chart	Click **More** > **Download** to download the chart to the local host.
Deleting a chart	Click **More** > **Delete** to delete the installed chart. Caution: Once a chart is deleted, it cannot be restored.

12.4 Creating a Workload Using a Chart

Creating a Chart-based Workload

Step 1 Log in to the CCE console. In the navigation pane, choose **Chart** > **Uploaded Chart**.

Step 2 In the list of uploaded charts, click **Install**.

Step 3 Set the installation parameters listed in **Table 12-3**. The parameters marked with an asterisk (*) are mandatory.

Table 12-3 Parameter description

Parameter	Parameter description
* Release Name	Unique name of the chart release.
* Chart Version	Chart version by default.
* Cluster	Cluster where the workload will be deployed.
* Namespace	Namespace to which the workload will be deployed.
Advanced Configuration	You can import and replace the **values.yaml** file or directly edit the chart parameters online. **NOTE** The imported **values.yaml** files 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 **Import Configuration File**. 2. Select the corresponding **values.yaml** file and click **Open**.

Step 4 After the configuration is complete, click **Custom Installation**.

Step 5 Confirm the order and click **Submit**.

Step 6 Click **Back to Release List** to view the running status of the chart-based workload (also called release), or click **View Release Details** to view details about the release.

----End

Upgrading a Chart-based Workload

Step 1 Log in to the CCE console. In the navigation pane, choose **Chart** > **Release**.

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.

Step 4 Modify the template parameters as prompted.

Step 5 Select an upgrade mode.

- If no more configuration is required, click **Upgrade at One Click**.
- To design the access mode, click **Upgrade**. For details about how to set the access mode, see **Network Management**. Click **Next** and then click **Submit**.

Step 6 Go back to the release list. If the release 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. In the navigation pane, choose **Chart** > **Release**.

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. In the navigation pane, choose **Chart** > **Release**.

Step 2 Click **More** > **Uninstall** next to the template instance to be uninstalled, and click **OK**. Exercise caution when performing this operation because workloads cannot be restored after the release is uninstalled.

----End

12.5 Using EVS Disks

CCE connects to EVS disks through its own add-on to support persistent storage.

The following example shows how to define an EVS disk in a chart. When creating a workload from the chart, the container dynamically creates a 10 Gi EVS disk and mounts it to the container.

NOTICE

Currently, CCE supports only creating EVS disks in a dynamic way.

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: helm-test-slave
spec:
  updateStrategy:
    type: "RollingUpdate"
  serviceName: helm-test-slave-headless
  replicas: 1
  template:
    metadata:
      labels:
        app: helm-test-slave
        type: slave
        release: "helm-test"
        failure-domain.beta.kubernetes.io/region: eu-west
        failure-domain.beta.kubernetes.io/zone: eu-west-0a
    spec:
```

```
    containers:
      - name: helm-test-slave
        image: nginx:alpine-per1
        volumeMounts:
        - mountPath: /redis-data
          name: helm-test-slave
        - mountPath: /opt/rancher/
          name: utility
        - mountPath: /etc/redis/
          name: redis-conf
        ports:
          - containerPort: 6379
volumeClaimTemplates:
- metadata:
    labels:
      app: helm-test-slave
      type: slave
      release: "helm-test"
    name: helm-test-slave
    annotations:
      "volume.beta.kubernetes.io/storage-class": sata
      "volume.beta.kubernetes.io/storage-provisioner": flexvolume-huawei.com/fuxivol
  spec:
    accessModes: [ "ReadWriteMany" ]
    resources:
      requests:
        storage: 10Gi
```

Table 12-4 Key parameters

Parameters	Parameter description
* annotations	Used for console display. **volume.beta.kubernetes.io/storage-class** indicates the EVS disk type, which can be **sas**, **sata** or **ssd**. For details, see the definition of the EVS service.
* accessModes	EVS disk access mode. ● **ReadWriteOnce**: The volume can be mounted as read-write by a single node. **NOTE** This function is supported only when the cluster version is v1.13.10 and the storage-driver version is 1.0.19. ● **ReadWriteMany**: The volume can be mounted as read-write by many nodes.
* resource.request.storage	Size of the EVS disk, in Gi. The minimum value is **10**.
* failure-domain.beta.kubernetes.io/region	Region where an EVS disk is located.
* failure-domain.beta.kubernetes.io/zone	Partition where an EVS disk is located.

12.6 Using ELB

Charts support the LoadBalancer Services. The definition method is the same as that of the Kubernetes community.

To display the LoadBalancer Service information on the CCE console, add the annotation to the chart of the corresponding resource type.

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: {{ .Release.Name }}-master
  annotations:
    "service.portal.kubernetes.io/access-ip": "10.4.4.14:8888"
    "service.portal.kubernetes.io/type": LoadBalancer
spec:
......
```

Table 12-5 Key parameters

Parameters	Parameter description
*annotations	Used for console display. **service.portal.kubernetes.io/access-ip** indicates the IP address and exposed port number of the ELB instance. The value of **service.portal.kubernetes.io/type** is fixed at **LoadBalancer**.

13 Add-on Management

13.1 coredns (System Resource Add-on, Mandatory)

Introduction

The coredns add-on is a DNS server that provides domain name resolution services for Kubernetes clusters. coredns chains plug-ins to provide additional features.

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. coredns chains plug-ins. Each plug-in provides a particular DNS function. You can integrate coredns with only the plug-ins you need to make coredns 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. Kubernetes v1.11 and later back CoreDNS as the official default DNS for all clusters going forward.

NOTE

- By default, CoreDNS is installed in clusters of Kubernetes v1.11 and later.
- Whenever there is an update or bug fix to coredns, you only need to install or upgrade the add-on instead of upgrading or creating the cluster.
- For details, see **Using Kubernetes In-Cluster DNS** or **Configuring High Availability of kube-dns/CoreDNS Using kubectl**.

Installing the Add-on

By default, coredns is installed in CCE clusters of Kubernetes v1.11 and later.

If coredns is not installed in a cluster, perform the following steps to install it:

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **coredns**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next**.

Step 3 In the **Configuration** step, set the following parameters:

Table 13-1 coredns add-on parameters

Parameters	Description
Add-on specifications	Concurrent domain name resolution ability. Select add-on specifications that best fit your needs.
Pods	Number of pods that will be created to match the selected add-on specifications. The number cannot be modified.
Container	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.
Warning	Add-on precautions. Read the precautions before you proceed with the step.
stub domain	A domain name server for a user-defined domain name. The format is a key-value pair. The key is a suffix of DNS domain name, and the value is one or more DNS IP addresses. For example, **acme.local -- 1.2.3.4,6.7.8.9** means that DNS requests with the **.acme.local** suffix are forwarded to a DNS listening at 1.2.3.4,6.7.8.9.

Step 4 After the preceding configurations are complete, click **Install**.

When the installation is complete, an instance of the add-on is installed on each node in the cluster.

----End

Configuring the Stub Domain for CoreDNS

Cluster administrators can modify the ConfigMap for the CoreDNS Corefile to change how service discovery works. coredns has the ability to configure stub domains using the proxy plug-in.

Assume that a cluster administrator has a Consul DNS server located at 10.150.0.1 and all Consul names have the suffix **.consul.local**. To configure Consul in coredns, the cluster administrator needs to write the following information in the coredns ConfigMap:

```
consul.local:5353 {
        errors
        cache 30
        proxy . 10.150.0.1
    }
```

ConfigMap after modification:

```
apiVersion: v1
data:
  Corefile: |-
    .:5353 {
        cache 30
        errors
        health
        kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          upstream /etc/resolv.conf
          fallthrough in-addr.arpa ip6.arpa
        }
        loadbalance round_robin
        prometheus 0.0.0.0:9153
        proxy . /etc/resolv.conf
        reload
    }

    consul.local:5353 {
        errors
        cache 30
        proxy . 10.150.0.1
    }
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
```

How Does Domain Name Resolution Work in Kubernetes?

DNS policies can be set on a per-pod basis. Currently, Kubernetes supports four types of DNS policies: **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. For details, see **https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/**. These policies are specified in the **dnsPolicy** field of the pod-specific.

- **Default**: Pods inherit the name resolution configuration from the node that runs the pods. 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-domain and upstream DNS servers configured.
- **ClusterFirstWithHostNet**: For pods running with hostNetwork, set its DNS policy **ClusterFirstWithHostNet**.
- **None**: It allows a pod to ignore DNS settings from the Kubernetes environment. All DNS settings are supposed to be provided using the **dnsPolicy** field in the pod-specific.

NOTE

- Clusters of Kubernetes v1.10 and later support **Default**, **ClusterFirst**, **ClusterFirstWithHostNet**, and **None**. Clusters earlier than Kubernetes v1.10 support only **Default**, **ClusterFirst**, and **ClusterFirstWithHostNet**.
- **Default** is not the default DNS policy. If **dnsPolicy** is not explicitly specified, **ClusterFirst** is used.

Routing

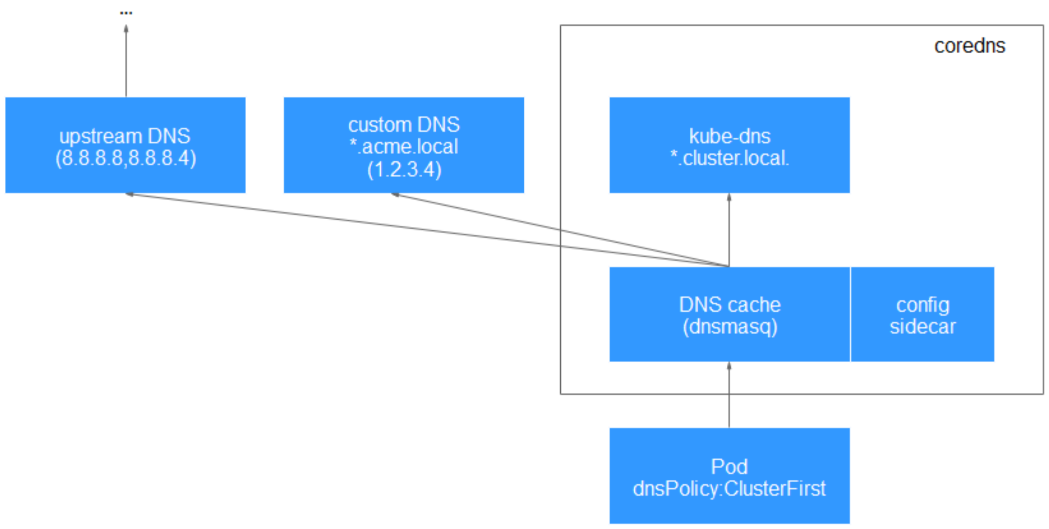
Without stub domain configurations: Any query that does not match the configured cluster domain suffix, such as **www.kubernetes.io**, is forwarded to the upstream DNS server inherited from the node.

With stub domain configurations: If stub domains and upstream DNS servers are configured, DNS queries are routed according to the following flow:

1. The query is first sent to the DNS caching layer in coredns.

2. From the caching layer, the suffix of the request is examined and then forwarded to the appropriate DNS, based on the following cases:
   - 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, listening for example at 1.2.3.4.
   - Names that do not match the suffix (for example, widget.com): The request is forwarded to the upstream DNS.

Figure 13-1 Routing



Upgrading the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Upgrade** under **coredns**.

NOTE

- If the Upgrade button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- When the upgrade is complete, the original coredns version on cluster nodes will be replaced by the latest version.

Step 2 On the **Basic Information** page, select the add-on version and click **Next**.

Step 3 Configure the parameters listed in **Table 13-2**. After the configuration is completed, click **Upgrade** to upgrade the coredns add-on.

Table 13-2 Parameters for installing coredns

Parameter	Parameter description
Add-on Specifications	Concurrent domain name resolution ability. Select add-on specifications based on service requirements.

Parameter	Parameter description
Stub domain	A domain name server for a user-defined domain name. The format is a key-value pair. The key is a suffix of DNS domain name, and the value is one or more DNS IP addresses. For example, **acme.local -- 1.2.3.4,6.7.8.9** means that DNS requests with the **.acme.local** suffix are forwarded to a DNS listening at 1.2.3.4,6.7.8.9.

----End

Uninstalling the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **coredns**.

Step 2 In the dialog box that is displayed, click **OK** to uninstall the add-on.

----End

13.2 storage-driver (System Resource Add-on, Mandatory)

Introduction

storage-driver is a FlexVolume driver used to support IaaS storage services, such as EVS and SFS.

NOTE

- The storage-driver is a system resource add-on. It is provided in the operating system images of the new nodes in CCE.
- When an upgrade or bug fix is available for the storage function, you only need to install or upgrade the storage-driver add-on to upgrade the function. Upgrading the cluster or creating a cluster is not required.

Installing the Add-on

By default, storage-driver is installed in CCE clusters of Kubernetes v1.11 and later.

If storage-driver is not installed in a cluster, perform the following steps to install it:

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **storage-driver**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next**.

Step 3 The storage-driver has no configurable parameters and can be directly installed by clicking **install**.

When the installation is complete, an instance of the add-on is installed on each node in the cluster.

----End

Upgrading the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Update** under **storage-driver**.

NOTE

- If the Upgrade button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- When the upgrade is complete, the original storage-driver version on cluster nodes will be replaced by the latest version.

Step 2 On the **Basic Information** page, select the add-on version and click **Next**.

Step 3 Click **Upgrade** to upgrade the storage-driver add-on. Note that the storage-driver has no configurable parameters and can be directly upgraded.

----End

Uninstalling Add-ons

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Uninstall** under **storage-driver**.

Step 2 In the dialog box that is displayed, click **OK** to uninstall the add-on.

----End

13.3 Everest (System Resource Add-on, Mandatory)

Introduction

Everest is a cloud-native container storage system. Based on CSI, clusters of Kubernetes v1.15.6 or later can interconnect with storage services such as CLOUD EVS, OBS, SFS, and SFS Turbo.

Everest is a system resource add-on. It is installed by default when a cluster of Kubernetes v1.15 or later is created.

Restrictions

This add-on can be installed only in hybrid clusters of v1.15 or later. By default, the **storage-driver (System Resource Add-on, Mandatory)** add-on is mandatory when a cluster of v1.13 or earlier is created.

Installing the Add-on

This add-on has been installed by default. If it is uninstalled due to some reasons, you can reinstall it by performing the following steps:

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **everest**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

Step 3 Select **Single** or **HA** for **Add-on Specifications**, and click **Install**.

After the add-on is installed, click **Back to Add-on List**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

Upgrading the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Upgrade** under **everest**.

NOTE

- If the **Upgrade** button is unavailable, the current add-on is already up-to-date and no upgrade is required.
- When the upgrade is complete, the original everest version on cluster nodes will be replaced by the latest version.

Step 2 On the **Basic Information** page, select the add-on version and click **Next**.

Step 3 Select **Single** or **HA** for **Add-on Specifications**, and click **Upgrade**.

----End

Uninstalling Add-ons

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Uninstall** under **everest**.

Step 2 In the dialog box displayed, click **OK** to uninstall the add-on.

----End

13.4 autoscaler

Introduction

The autoscaler add-on is used to automatically scale in or out nodes in a cluster based on the pod scheduling status and resource usage.

CCE simplifies the creation, upgrade, and manual scaling of Kubernetes clusters, in which traffic loads change over time. To balance resource usage and workload performance of nodes, Kubernetes introduces the autoscaler add-on to automatically resize a cluster based on the resource usage required for workloads deployed in the cluster.

The autoscaler is applied in the following two scenarios:

- Automatic scale-up is triggered if there are pods that failed to be scheduled onto any nodes in a cluster due to insufficient node resources. The add-on follows the "No Less, No More" policy. If three cores are required for creating a pod and there are four-core and eight-core nodes, a four-core node is preferentially created.

  NOTE

  An auto scale-up will be performed only when either of the following conditions has been met:

  - Node resources are insufficient.
  - No node affinity settings are included in other scheduling configurations. For more information about node affinity configuration, see **Affinity and Anti-Affinity Scheduling**.
- If a node in a cluster is not fully used for a period of time and the pods on the node can be scheduled to other nodes, a scale-down is automatically performed to remove the node.

Constraints

- Only clusters of version 1.9.7-r1 and later support autoscaler.
- This autoscaler add-on cannot be used with the CPU-usage-based node autoscaler.

Installing the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **autoscaler**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next**.

Step 3 Configure add-on installation parameters listed in **Table 13-3**.

Table 13-3 Basic settings

Parameter	Add-on Version	Description
Add-on Specifications	All versions	Possible values: ● **Single**: The add-on has only one instance. ● **HA**: The add-on has multiple instances for higher availability. This mode requires more compute resources.
Instances	All versions	Number of instances that will be created to match the selected add-on specifications. The number cannot be modified.
Container	All versions	CPU and memory quotas of the container allowed for the selected add-on specifications. The quotas cannot be modified.

Parameter	Add-on Version	Description
Login Mode	Available only in certain versions	Select a login mode for auto scale-out nodes.

Parameter	Add-on Version	Description
Auto Scale-In	All versions	**Off**: Auto scale-down is not allowed. Only auto scale-up is allowed. **On**: Auto scale-down is allowed for both existing and added nodes. ● **Idle Time (min)**: Time for which a node should be unneeded before it is eligible for scale-down. Default value: 10 minutes. ● **Resource Usage**: If the percentage of both CPU and memory usage on a node is below this threshold, auto scale-down will be triggered to delete the node from the cluster. The default value is 0.5, which means 50%. ● **Scaledown Delay After Scaleup**: The time after scale-up that the scale-down evaluation will resume. Default value: 10 minutes. ● **Scaledown Delay After Node Deletion**: The time after node deletion that the scale-down evaluation will resume. Default value: 10 minutes. ● **Scaledown Delay After Failure**: The time after a scale-down failure that the scale-down evaluation will resume. Default value: 3 minutes. ● **Max empty bulk delete**: The maximum number of empty nodes that can be deleted at the same time. Default value: 10. ● **Node Recheck Timeout**: The timeout before autoscaler checks again the node that could not be previously removed. Default value: 5 minutes.

Parameter	Add-on Version	Description
		NOTE If a node is consistently unneeded for a significant amount of time (default: 10 min), it will be considered for removal. However, the following pods cannot be removed from a cluster: ● Pods with restrictive **PodDisruptionBudget** ● Pods with local storage ● Pods that cannot be moved elsewhere due to various constraints (lack of resources, non-matching node selectors or affinity, matching anti-affinity, etc) ● Pods that have the **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** annotation ● Pods (except those created by kube-system DaemonSet) that exist in the kube-system namespace on the node ● Pods that are not created by deployments, ReplicaSets, jobs, StatefulSets, or other controllers

Parameter	Add-on Version	Description
Node Pool Configuration	Available only in certain versions	A node pool is a group of compute nodes with the same node type (VM or BMS), specifications, and labels. When the cluster needs to be scaled up, autoscaler will automatically select nodes from these node pools to the cluster. If no custom node pool is available, autoscaler will use the default node pool. Click **Add Node Pool Configuration** and set the following parameters: ● **AZ**: A physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected through the internal network. ● **OS**: Select the operating system (OS) of the nodes to be created. ● **Taints**: No taints are added by default. Taints allow nodes to repel a set of pods. You can add a maximum of 10 taints for each node pool. Each taint contains the following parameters: – **Key**: A key must contain 1 to 63 characters starting with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. A DNS subdomain name can be used as the prefix of a key. – **Value**: A value must start with a letter or digit and can contain a maximum of 63 characters, including letters, digits, hyphens (-), underscores (_), and periods (.). – **Effect**: Available options are **NoSchedule**, **PreferNoSchedule**, and **NoExecute**. **NOTICE** – If taints are used, you must configure tolerations in the YAML files of pods. Otherwise, scale-up may fail or pods cannot be scheduled onto the added nodes. – Taints cannot be modified after configuration. Incorrect taints may cause a scale-up failure or prevent pods from being scheduled onto the added nodes. ● **Resource Tags**: Resource tags can be added to classify resources.

Parameter	Add-on Version	Description
		NOTE You can create predefined tags in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency. ● **Specifications**: CPU and memory of the added nodes.

To configure more add-on parameters, click **Advanced Settings** at the bottom of this page.

Table 13-4 Advanced Settings

Parameter	Add-on Version	Description
Total Nodes	All versions	The total number of nodes up to which the cluster can be scaled.
Total Cores	All versions	The total number of cores up to which the cluster can be scaled.
Total Memory (GB)	All versions	The total memory up to which the cluster can be scaled.
Auto Scale-Out	Available only in certain versions	**Triggered when there are pods unscheduled**: Selected by default and cannot be deselected.

Step 4 When the configuration is complete, click **Install**.

After the installation is complete, you can view the add-on instance on the **Add-on Instance** tab page. This indicates that the add-on has been installed in the current cluster.

----End

Uninstalling the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Uninstall** under **autoscaler**.

Step 2 In the dialog box that is displayed, click **OK** to uninstall the add-on.

----End

13.5 metrics-server

From version 1.8 onwards, Kubernetes provides resource usage metrics, such as the container CPU and memory usage, through the Metrics API. These metrics can be directly accessed by users (for example, by using the **kubectl top** command) or used by controllers (for example, Horizontal Pod Autoscaler) in a cluster for decision-making. The specific component is metrics-server, which is used to substitute for heapster for providing the similar functions. heapster has been gradually abandoned since v1.11.

metrics-server is an aggregator for monitoring data of core cluster resources. You can quickly install the add-on on the CCE console.

The official community project and documentation are available at **https://github.com/kubernetes-sigs/metrics-server**.

Restrictions

This add-on can be installed only in hybrid clusters of v1.13 or later.

Installing the Add-on

After metrics-server is installed, you can create an HPA policy on the **Workload Scaling** tab page of the **Auto Scaling** page. For details, see **Workload Scaling**.

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **metrics-server**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next: Configuration**.

Step 3 Select **Single** or **HA** for **Add-on Specifications**, and click **Install**.

After the add-on is installed, click **Back to Add-on List**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each node in the cluster.

----End

Uninstalling Add-ons

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, click **Uninstall** under **metrics-server**.

Step 2 In the dialog box displayed, click **OK** to uninstall the add-on.

----End

13.6 gpu-beta

Introduction

gpu-beta is a device management add-on that supports GPUs in containers. It supports only NVIDIA drivers.

Constraints

- The cluster where the gpu-beta add-on is installed must contain GPU nodes.
- The driver to be downloaded must be a **.run** file.

NOTICE

- If the download link is a public network address, for example, NVIDIA official website, bind an EIP to each GPU node.
- If the download link is an OBS URL, you do not need to bind an EIP to GPU nodes.
- The latest version of the NVIDIA driver supported by the gpu-beta add-on is 396.37. You are not advised to install a driver of a later version.
- After the driver version is changed, restart the node for the change to take effect.

Installing the Add-on

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Marketplace** tab page, click **Install Add-on** under **gpu-beta**.

Step 2 On the **Install Add-on** page, select the cluster and the add-on version, and click **Next**.

Step 3 In the **Configuration** step, select one way to configure the driver address.

CAUTION

The driver will not be uninstalled during gpu-beta add-on uninstall. If the driver is reinstalled, you must restart all GPU nodes.

When installing the gpu-beta add-on of version 1.1.8, you can only customize the driver address for the NVIDIA driver.

1. If you select **Verified source** for **Nvidia Driver**, select the NVIDIA driver that has been verified.
2. If you select **Custom source** for **Nvidia Driver**, enter the NVIDIA driver link you have obtained. For details about how to obtain the link, see **Obtaining the Driver Link**.

   For example, the NVIDIA 396.37 driver can be downloaded at *https://us.download.nvidia.com/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run*.

Step 4 Click **Install**.

After the add-on is installed, click **Back to Add-on List**. On the **Add-on Instance** tab page, select the corresponding cluster to view the running instance. This indicates that the add-on has been installed on each GPU node in the cluster.

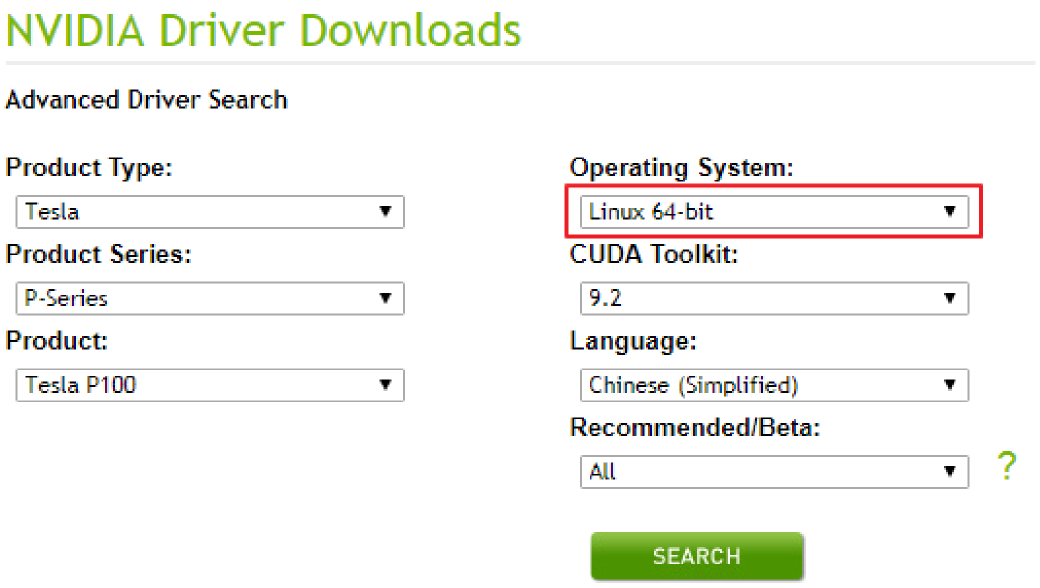
----End

Obtaining the Driver Link

Step 1 Log in to ***https://www.nvidia.cn/Download/Find.aspx?lang=en***.

Step 2 Select the driver information on the **NVIDIA Driver Downloads** page, as shown in **Figure 13-2**. **Operating System** must be **Linux 64-bit**.

Figure 13-2 Setting parameters


NVIDIA Driver Downloads

Advanced Driver Search

Product Type: Tesla

Product Series: P-Series

Product: Tesla P100

Operating System: Linux 64-bit

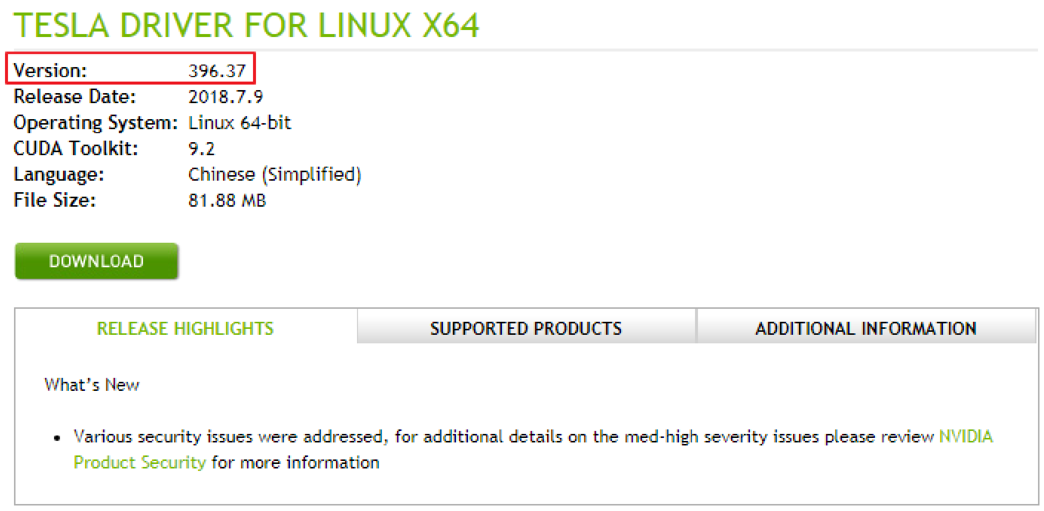
CUDA Toolkit: 9.2

Language: Chinese (Simplified)

Recommended/Beta: All

Step 3 After confirming the driver information, click **SEARCH**. A page is displayed, showing the driver information, as shown in **Figure 13-3**. Click **DOWNLOAD**.

Figure 13-3 Driver information



TESLA DRIVER FOR LINUX X64

Version:	396.37
Release Date:	2018.7.9
Operating System:	Linux 64-bit
CUDA Toolkit:	9.2
Language:	Chinese (Simplified)
File Size:	81.88 MB

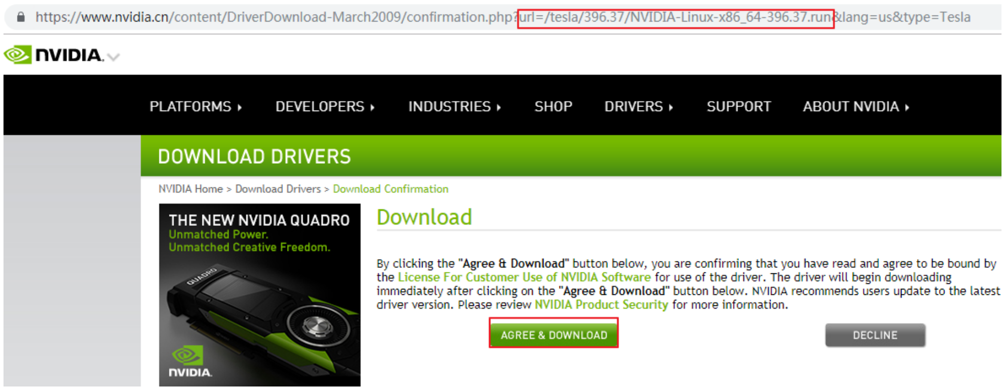
What's New

- Various security issues were addressed, for additional details on the med-high severity issues please review NVIDIA Product Security for more information

Step 4 Obtain the driver link in either of the following ways:

- Method 1: As shown in **Figure 13-4**, find *url=/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run* in the browser address box. Then, supplement it to obtain the driver link **https://us.download.nvidia.com/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run**. By using this method, you must bind an EIP to each GPU node.
- Method 2: As shown in **Figure 13-4**, click **Download** to download the driver and upload it to OBS to obtain the software link. In this method, you do not need to bind an EIP to the node. However, you must ensure that the OBS bucket is public and readable..

Figure 13-4 Obtaining the link



https://www.nvidia.cn/content/DriverDownload-March2009/confirmation.php?url=/tesla/396.37/NVIDIA-Linux-x86_64-396.37.run&lang=us&type=Tesla

By clicking the "Agree & Download" button below, you are confirming that you have read and agree to be bound by the License For Customer Use of NVIDIA Software for use of the driver. The driver will begin downloading immediately after clicking on the "Agree & Download" button below. NVIDIA recommends users update to the latest driver version. Please review NVIDIA Product Security for more information.

----End

Uninstalling Add-ons

Step 1 Log in to the CCE console. In the navigation pane, choose **Add-ons**. On the **Add-on Instance** tab page, select the cluster and click **Uninstall** under **gpu-beta**.

Step 2 In the dialog box displayed, click **Yes** to uninstall the add-on.

NOTE

The driver will not be uninstalled during gpu-beta add-on uninstall. If the driver is reinstalled, you must restart all GPU nodes.

----End

14 Auto Scaling

14.1 Overview

CCE supports auto scaling in two ways:

1. (Recommended) **Using add-ons (autoscaler/hpa)**
2. **Using AOM** (The AOM service monitors the scaling metrics and triggers scaling.)

Auto Scaling by Using CCE Add-ons (autoscaler/hpa)

How to use

1. For workload scaling:
   a. In the navigation pane, choose **Auto Scaling** > **Workload Scaling**.
   b. In the navigation pane, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets**. On the workload details page, click the **Scaling** tab and then configure/view scaling policies in the **Auto Scaling by HPA/CustomedHPA** area.
2. For node scaling: In the navigation pane, choose **Auto Scaling** > **Node Scaling**.

Constraints

- Install the required add-ons in the cluster. For details, see **Workload Scaling** and **Node Scaling**.
- Node scaling applies only to nodes in a node pool. You must enable the auto scaling function when creating a node pool or modifying its configuration, so that nodes in the pool can be automatically added or reduced.

Notes

- Workload scaling can be triggered by metric-based and scheduled policies.
- Node scale-out can be triggered by pod scheduling failure (automatically adding nodes in the node pool when pods cannot be scheduled), metric-based policies, and scheduled policies.

- Node scale-in can be triggered only by the resource allocation rate. When CPU and memory allocation rates in a cluster are lower than the specified thresholds (set when the autoscaler add-on is installed or modified), scale-in is triggered for nodes in the node pool (this function can be disabled).
- The default node pool DefaultPool is designed to group all the nodes that do not belong to any user-created node pool.

Auto Scaling by Using AOM

How to use

1. For workload scaling: In the navigation pane, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets**. On the workload details page, click the **Scaling** tab and then configure/view the scaling settings in the **Auto Scaling by AOM** area. For details, see **Auto Scaling - AOM**.
2. For node scaling: In the navigation pane on the left, choose **Resource Management** > **Clusters**. On the cluster details page, click the **Auto Scaling** tab. For details, see **Cluster Auto Scaling**.

Constraints

Clusters of v1.17 and later do not support AOM-based auto scaling.

Notes

- Workload scaling can be triggered by metric-based and scheduled policies.
- Node scale-out can be triggered by metric-based policies or scheduled policies. Node scale-in is not supported.
- If you configure node scaling on the **Auto Scaling** tab page in the cluster details, only a few parameters can be configured and one configuration applies to all scenarios.

14.2 Workload Scaling

Workload scaling allows you to create HPA policies.

HPA: a policy that implements horizontal scaling of pods in Kubernetes. This policy adds the HPA-level cooldown time window and scaling threshold functions based on the HPA function provided by the Kubernetes community.

Prerequisites

To use the HPA policy function, you need to install the **metrics-server** add-on to collect the running indicators of the workloads associated with the HPA policy.

Constraints

HPA policies can be created only for clusters of v1.13 or later.

Creating an HPA Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, check whether the **metrics-server** add-on has been installed and is running properly.

Step 2 After installing the metrics-server add-on, click **Create HPA Policy**.

Step 3 On the **Create HPA Policy** page displayed, set the policy parameters listed in **Table 14-1**.

Table 14-1 HPA policy parameters

Parameter	Parameter description
Policy Name	Name of the policy to be created. Set this parameter as required.
Cluster Name	Cluster to which the workload belongs.
Namespace	Namespace to which the workload belongs.
Workload	Workload with which the HPA policy is associated.
Pod Range	Minimum and maximum numbers of pods. When a policy is triggered, the workload pods are scaled within this range.
Cooldown Period	Interval between two consecutive scaling tasks, in minutes. **This parameter is available only for clusters of v1.15 or later. For clusters of v1.13, the default scaling cooldown period is 0 minutes.** This parameter indicates the interval between consecutive scaling operations. The cooldown period ensures that a scaling operation is initiated only when the previous one is completed and the system is running stably.
Execution Rules	• **Metric**: You can select **CPU usage** or **Memory usage**. Usage = CPUs or memory used by pods/Requested CPUs or memory. • **Expected Value**: Enter an expected value. The expected value of the selected metric works with the threshold. The actual number of pods for scaling is calculated using the following formula: Expected value/Threshold x Number of current pods • **Threshold**: When the metric value is less than the scale-in threshold, scale-in is triggered. When the metric value is greater than the scale-out threshold, scale-out is triggered. You can click **Add Rule** again to add more scaling policies.

Step 4 After the configuration is complete, click **Create**. If the system displays a message indicating that the request to create workload policy *** is successfully submitted, click **Back to Workload Scaling**.

Step 5 On the **Workload Scaling** tab page, you can view the newly created HPA policy.

----End

Update the HPA policy.

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the Workload Scaling tab page, click **Update** in the **Operation** column of the policy to be updated.

Step 2 On the **Update HPA Policy** page displayed, set the policy parameters listed in **Table 14-1**.

Step 3 Click **Update**.

----End

Cloning an HPA Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the Workload Scaling tab page, click **Clone** in the **Operation** column of the policy to be updated.

Step 2 For example, for an HPA policy, on the **Create HPA Policy** page, you can view that parameters such as **Pod Range**, **Cooldown Period**, and **Rules** have been cloned. Add or modify other policy parameters as needed.

Step 3 Click **Create**.

After the clone is complete, you can view the new clone policy in the policy list on the **Workload Scaling** tab page.

----End

Editing a YAML file

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, choose **More** > **Edit YAML** in the **Operation** column of the policy to be updated.

Step 2 In the **Edit YAML** dialog box displayed, edit or download the YAML file.

Step 3 Click the close button in the upper right corner.

----End

Deleting an HPA Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, choose **More** > **Delete** in the **Operation** column of the policy to be updated.

Step 2 In the **Delete HPA Policy** dialog box displayed, confirm whether to delete the HPA policy.

Step 3 Click **Yes** to delete the policy.

----End

Checking an HPA Policy

You can view the rules, status, and events of an HPA policy and handle exceptions based on the error information displayed.

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Workload Scaling** tab page, click ⌄ in front of the target policy.

Step 2 In the expanded area, you can view the **Rules**, **Status**, and **Events** tab pages. If the policy is abnormal, locate and rectify the fault based on the error information.

NOTE

You can also view the created HPA policy on the workload details page. Log in to the CCE console, choose **Workloads > Deployments** or **Workloads > StatefulSets** in the navigation pane, and choose **More** > **Scaling** in the **Operation** column. On the workload details page, click the **Scaling** tab. You can see the **Auto Scaling-HPA** pane, as well as the HPA policy you have configured on the **Auto Scaling** page.

Table 14-2 Event types and names

Event type	Event Name	Description
Success response	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 obtain compute metrics.
	FailedGetScaleWindow	Failed to obtain ScaleWindow.
	FailedRescale	Failed to scale the service.

----End

14.3 Node Scaling

This section describes how to perform node scaling.

Prerequisites

Before using the node scaling function, you must install the **autoscaler** add-on of v1.13.8 or later.

Constraints on Scale-in

CCE cannot trigger scale-in by using node scaling policies. You can set a scale-in policy when installing the add-on **autoscaler**.

Node scale-in can be triggered only by the resource allocation rate. When CPU and memory allocation rates in a cluster are lower than the specified thresholds (set when the autoscaler add-on is installed or modified), scale-in is triggered for nodes in the node pool (this function can be disabled).

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, check whether the add-on has been installed and is running properly.

Step 2 After installing the autoscaler add-on, click **Create Node Scaling Policy**.

Step 3 On the **Create Node Scaling Policy** page, set policy parameters listed in **Table 14-3**.

Table 14-3 Node scaling policy parameters

Parameter	Description
Policy Name	Name of the policy to be created. Set this parameter as required.
Associated Node Pool	Add a node pool. You can associate multiple node pools to use the same scaling policy.

Parameter	Description
Execution Rules	Click **Add Rule**. In the **Add Rule** dialog box displayed, set the following parameters: **Rule Name**: Enter a custom rule name. **Type**: You can select **Metric-based** or **Periodic**. The differences between the two types are as follows: ● **Metric-based**: – **Condition**: Select **CPU usage** or **Memory usage** and enter a value. The value must be greater than the scale-in percentage configured in the autoscaler add-on. – **Action**: Set an action to be performed when the trigger condition is met. ● **Periodic**: – **Triggered At**: You can select a specific time point every day, every week, every month, or every year. – **Action**: Set an action to be performed when the **Triggered At** value is reached. You can click **Add Rule** again to add more node scaling policies. You can add a maximum of one CPU usage-based rule and one memory usage-based rule. The total number of rules cannot exceed 10.

Step 4 After the configuration is complete, click **Create**. If the system displays a message indicating that the request to create a node scaling policy is submitted successfully, click **Back to Node Scaling Policy List**.

Step 5 On the **Node Scaling** tab page, you can view the created node scaling policy.

----End

Deleting a Node Scaling Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **Delete** in the **Operation** column of the policy to be deleted.

Step 2 In the **Delete Node Policy** dialog box displayed, confirm whether to delete the policy.

Step 3 Enter **DELETE** in the text box.

Step 4 Click **OK** to delete the policy.

----End

Editing a Node Scaling Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **Edit** in the **Operation** column of the policy to be edited.

Step 2 On the **Create Node Scaling Policy** page displayed, modify policy parameters listed in **Table 14-3**.

Step 3 After the configuration is complete, click **OK**.

----End

Cloning a Node Scaling Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **More** > **Clone** in the **Operation** column of the policy to be cloned.

Step 2 On the **Create Node Scaling Policy** page displayed, certain parameters have been cloned. Add or modify other policy parameters based on service requirements.

Step 3 Click **Create Now** to clone the policy. The cloned policy is displayed in the policy list on the **Node Scaling** tab page.

----End

Enabling or Disabling a Node Scaling Policy

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click **More** > **Disable** or **Enable** in the **Operation** column of the policy.

Step 2 In the dialog box displayed, confirm whether to disable or enable the node policy.

Step 3 Click **Yes**. The policy status is displayed in the node scaling list.

----End

Viewing a Node Scaling Policy

You can view the associated node pool, rules, and scaling history of a node scaling policy and rectify faults according to the error information displayed.

Step 1 Log in to the CCE console. In the navigation pane, choose **Auto Scaling**. On the **Node Scaling** tab page, click ⌄ in front of the policy to be viewed.

Step 2 In the expanded area, the **Associated Node Pool**, **Execution Rules**, and **Scaling Records** tab pages are displayed. If the policy is abnormal, locate and rectify the fault based on the error information.

NOTE

You can also enable or disable auto scaling in **Node Pools**. Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Node Pools**, and click **Edit** in the upper right corner of the node pool to be operated. In the **Edit Node Pool** dialog box displayed, you can enable **Autoscaler** and set the limits of the number of nodes and the cool-down interval for auto scaling.

----End

FAQs

1. **If multiple rules meet the conditions at the same time, how will the rules be executed?**

   This issue occurs in either of the following scenarios:

   – If rules based on the **CPU allocation rate** and **memory allocation rate** are configured and two or more rules meet the scale-out conditions, the rule that will add the most nodes will be executed.

   – If a rule based on the **CPU allocation rate** and **a periodic rule** are configured and they both meet the scale-out conditions, one of them will be executed randomly. The rule executed first (rule A) changes the node pool to the scaling state. As a result, the other rule (rule B) cannot be executed properly. After rule A is executed and the node pool status becomes normal, rule B will not be executed.

2. **How long is the detection period of a policy after the CPU or memory-based rule is configured? Is node scale-out triggered once conditions are met?**

   The period is not fixed and varies according to the processing logic of each loop of the autoscaler add-on. Scale-out is triggered once the conditions are met, but it is constrained by other factors such as the cool-down interval and node pool status.

3. **How does node scaling work with the autoscaler add-on?**

   If a node scaling policy and the configuration in the autoscaler add-on take effect at the same time, for example, there are pods that cannot be scheduled and the value of a metric reaches the threshold at the same time, scale-out is performed first for the unschedulable pods.

   – If the scale-out succeeds for the unschedulable pods, the system skips the metric-based rule logic and enters the next loop.

   – If the scale-out fails for the unschedulable pods, the metric-based rule is executed.

15 Permissions Management

15.1 Permissions Management by IAM

This chapter describes Identity and Access Management (IAM)'s fine-grained permissions management for your Cloud Container Engine (CCE) service. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials, providing access to CCE resources.
- Grant only the permissions required for users to perform a task.
- Entrust an account or cloud service to perform professional and efficient O&M on your CCE resources.

If your account does not need individual IAM users, then you may skip over this chapter.

The following sections describe the common IAM operations, including creating a user and user group, granting permissions to a user group, and creating a custom policy.

Domain

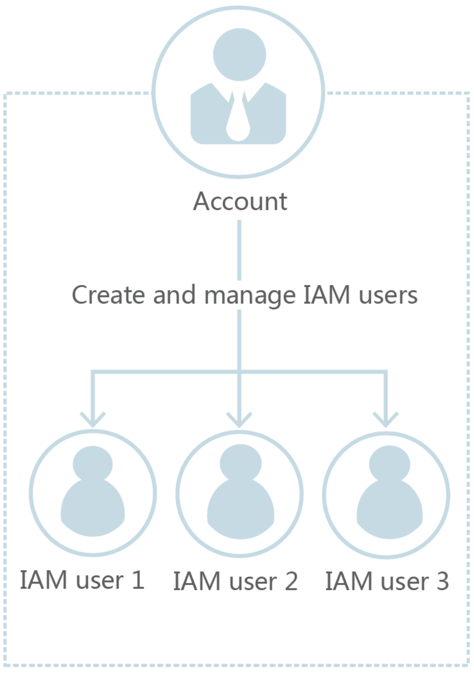
To use cloud services, you need to register an account using your mobile number. The account owns your cloud resources and has full access permissions for the resources. You can use the account to reset user passwords and assign permissions. Your account receives and pays all bills generated by your IAM users' use of resources. To log in to the management console using an account, choose **Account Login**.

IAM user

IAM users are created by an account. Each IAM user has its own password and access keys to access the cloud, and uses cloud resources based on assigned permissions. IAM users do not own resources and cannot make payments.

Relationship between the account and IAM users

An account and its IAM users are like a parent and children. The account owns the resources and makes payments for IAM users' resource usage. It has full access permissions for these resources. IAM users are created using the account, and only have the permissions granted by the account. The account can be used to modify or cancel the IAM users' permissions at any time. Fees generated by IAM users' use of resources are paid by the administrator.



Identity Credentials

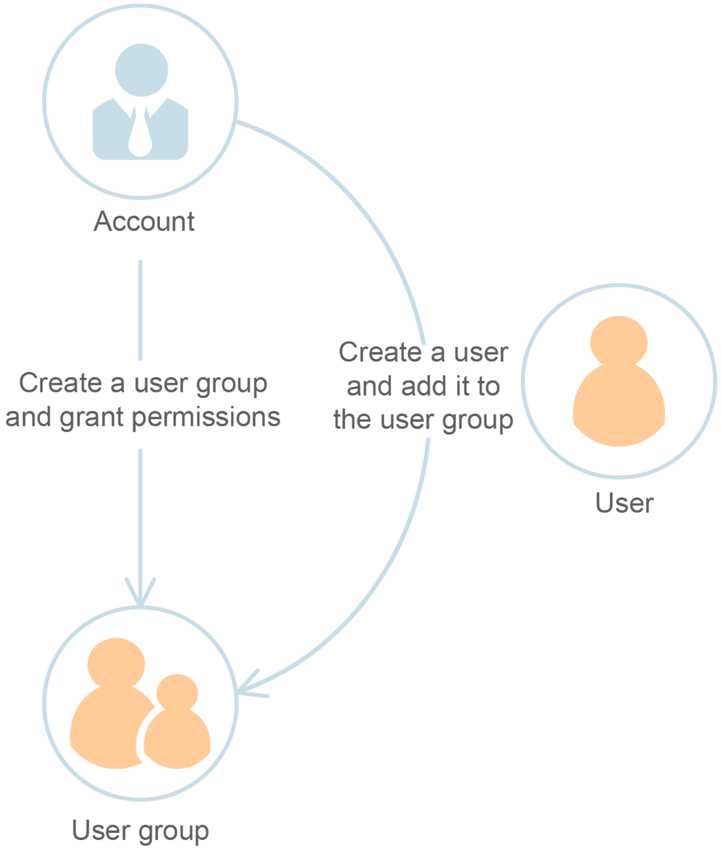
Credentials confirm the identity of a user that accesses cloud services through the console or APIs. Credentials include a password and access keys, which are managed in IAM.

- Password: A common credential for logging in to the management console or calling APIs.
- Access key: An access key ID/secret access key (AK/SK) pair, which can only be used to call APIs. Each access key provides a signature for cryptographic authentication to ensure that access requests are secret, complete, and correct.

User group

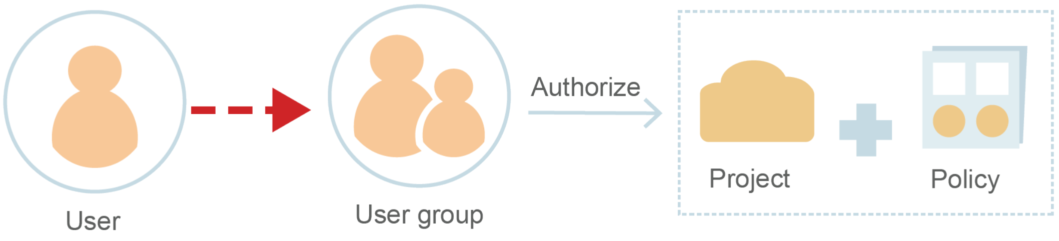
User groups facilitate centralized user management and streamlined permissions management. Users in the same user group inherit the same permissions. IAM users must be added to a user group to obtain the permissions required for accessing specified resources or cloud services under the account. A user can be added to multiple groups, which allows them to inherit different permissions.

The default user group **admin** has all of the permissions required to use all of the cloud resources. Users in this group can perform operations on all the resources, including but not limited to creating user groups and users, assigning permissions, and managing resources.


Account
Create a user group and grant permissions
Create a user and add it to the user group
User
User group

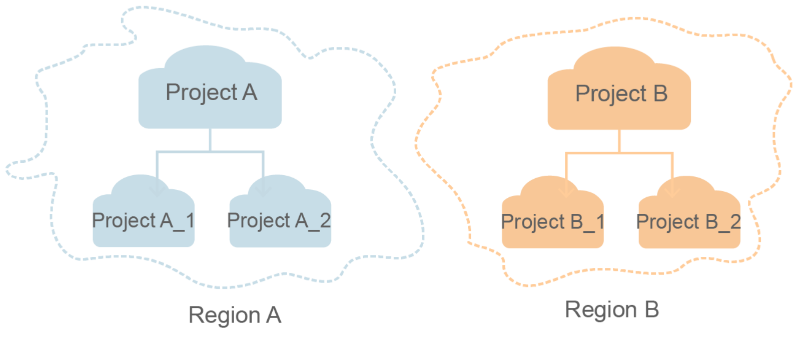
Authorization

Authorization is the process of granting required permissions for a user to perform a task. After a system or custom policy is assigned to a user group, users in the group inherit the permissions defined by the policy to manage resources. For example, managing Elastic Cloud Servers (ECSs).



Inspection Item

Projects group and isolate resources (including compute, storage, and network resources) across physical regions. A default project is provided for each region, and subprojects can be created under each default project. Users can be granted permissions to access all resources in a specific project. If you need more refined access control, you can create subprojects under a default project and purchase resources in subprojects. Then you can assign required permissions for users to access only the resources in specific subprojects.

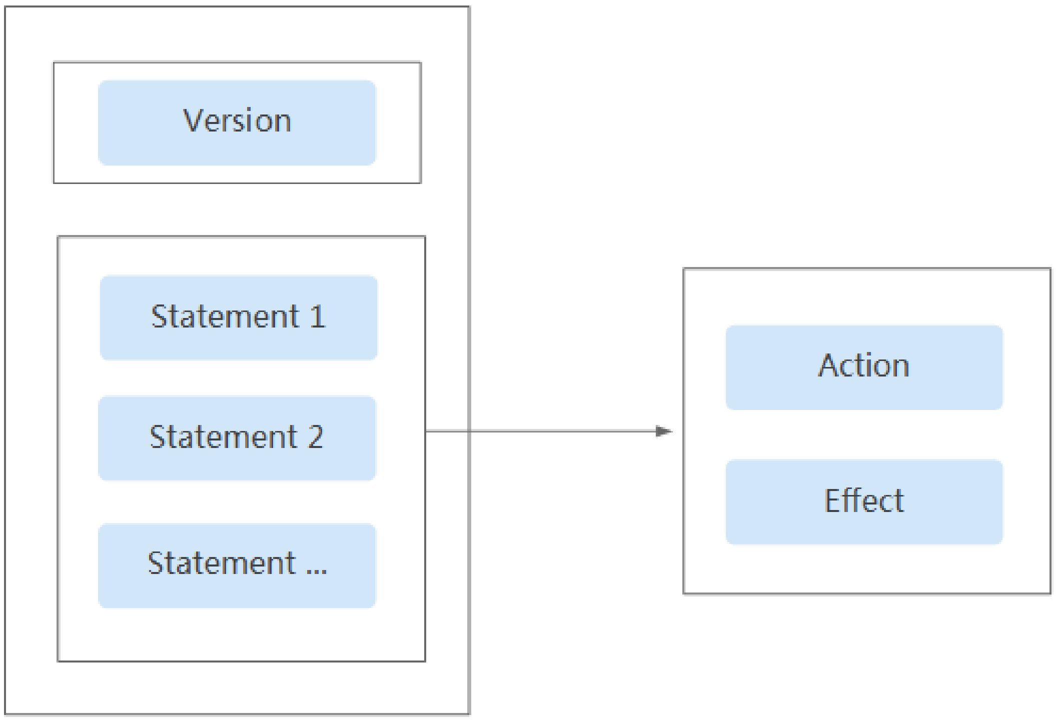


15.2 Policy Syntax

Policy Structure

A permissions policy consists of a Version and a Statement. Each policy can have multiple statements.

Figure 15-1 Policy structure



Policy Syntax

The **CCE Viewer** policy is used as an example to describe policy syntax.

```
{
        "Version": "1.1",
        "Statement": [
                {
                        "Effect": "Allow",
                        "Action": [
                    "cce:*:get",
                    "cce:*:list",
                    "cce:kubernetes:*",
                    "ecs:*:get",
                    "ecs:*:list",
                    "evs:*:get",
                    "evs:*:list",
                    "evs:*:count",
                    "vpc:*:get",
                    "vpc:*:list",
                    "elb:*:get",
                    "elb:*:list",
                    "sfs:*:get",
                    "sfs:*:list",
                    "aom:*:get",
                    "aom:*:list",
                    "aom:autoScalingRule:*"
               ]
          }
     ]
}
```

- **Version**: Distinguishes between role-based access control (RBAC) and fine-grained policies.
  - **1.0**: RBAC policies. An RBAC policy consists of permissions for an entire service. Users in a group with such a policy assigned are granted all of the permissions required for that service.
  - **1.1**: Fine-grained policies. A fine-grained policy consists of API-based permissions for operations on specific resource types. Fine-grained policies, as the name suggests, allow for more fine-grained control than RBAC policies. Users assigned permissions of such a policy can only perform specific operations on the corresponding service. Fine-grained policies are classified into system-defined and custom policies.
- Statement: Permissions defined by a policy, including Effect and Action.
  - Effect

    The valid values for Effect are Allow and Deny. System policies contain only Allow statements. For custom policies containing both Allow and Deny statements, the Deny statements take precedence.
  - Action

    Permissions in the format of *Service name:Resource type:Operation*. A policy can contain one or more permissions. The wildcard (*) is allowed to indicate all of the services, resource types, or operations depending on its location in the action.

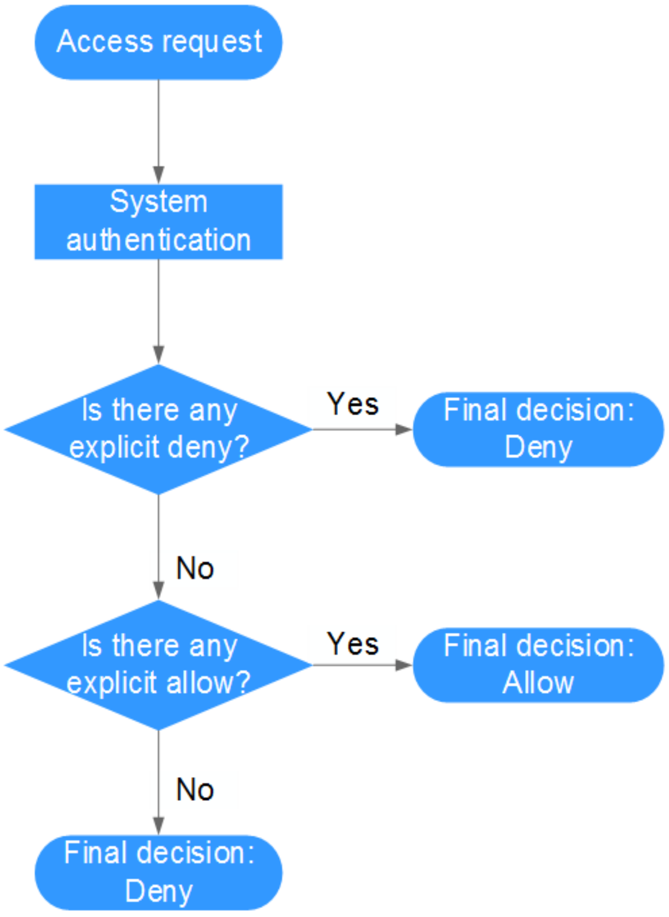
    Examples:

    - **cce:*:get**: Permissions for querying all types of resources in CCE.
    - **cce:kubernetes:***: Permissions for performing all operations on Kubernetes resources in CCE.

Authentication Logic

If a user is assigned permissions of multiple policies or of only one policy containing both Allow and Deny statements, then authentication starts from the Deny statements. The following figure shows the authentication logic for resource access.

Figure 15-2 Authentication logic



NOTE

The actions in each policy bear the OR relationship.

Step 1 A user accesses the system and makes an operation request.

Step 2 The system evaluates all the permissions policies assigned to the user.

Step 3 In these policies, the system looks for explicit deny permissions. If the system finds an explicit deny that applies, it returns a decision of Deny, and the authentication ends.

Step 4 If no explicit deny is found, the system looks for allow permissions that would apply to the request. If the system finds an explicit allow permission that applies, it returns a decision of Allow, and the authentication ends.

Step 5 If no explicit allow permission is found, IAM returns a decision of Deny, and the authentication ends.

----End

15.3 Permissions Management for CCE

15.3.1 Overview

CCE permissions management allows tenants to grant permissions to IAM users and user groups under the tenant accounts. It combines the advantages of Kubernetes Role-based Access Control (RBAC) authorization and Identity and Access Management (IAM) to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, cluster-scoped authorization, and namespace-wide authorization.

CCE permissions management is classified as:

- **Cluster-level permissions management**: On IAM, tenants can configure fine-grained policies to describe which IAM user groups can perform which operations on cluster resources. For example, tenants can grant user group A to create and delete cluster X while granting user group B to view cluster X information. Users created in IAM inherit permissions from the groups to which they belong.
- **Namespace-level permissions management**: Tenants regulate user or user group access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

15.3.2 Cluster-Level Permissions Management (By Using IAM Fine-Grained Authorization)

Cluster-level permissions management evolves out of the fine-grained authorization feature of IAM. User groups facilitate centralized user management and streamlined permissions management. Users in the same user group have the same permissions. Currently, two default roles are available: **CCE Admin** and **CCE Viewer**. In this chapter, role and permissions policy are interchangeable. Both represent a set of permissions.

Permissions Granted by the CCE Admin Role

The default role **CCE Admin** has the following permissions:

Table 15-1 Permissions granted by the CCE Admin role

Action	Specific Action	Description
cce:*:*	cce:cluster:create	Create a cluster
	cce:cluster:delete	Delete a cluster.
	cce:cluster:update	Update a cluster, for example, update cluster node scheduling parameters and provide RBAC support to clusters.
	cce:cluster:upgrade	Upgrade a cluster.
	cce:cluster:start	Wake up a cluster.
	cce:cluster:stop	Hibernate a cluster.
	cce:cluster:list	List all clusters.

Action	Specific Action	Description
	cce:cluster:get	Query cluster details.
	cce:node:create	Add a cluster node.
	cce:node:delete	Delete one or more nodes.
	cce:node:update	Update a cluster node, for example, update the node name.
	cce:node:get	Query node details.
	cce:node:list	List all nodes.
	cce:job:list	List all cluster jobs.
	cce:job:delete	Delete one or more cluster jobs.
	cce:job:get	Read a specified cluster job.
	cce:storage:create	Create a storage volume.
	cce:storage:delete	Delete a storage volume.
	cce:kubernetes:*	Perform operations on all Kubernetes resources. For details, see **Namespace-level Permissions**.
ecs:*:*	-	Perform all operations on Elastic Cloud Servers (ECSs).
evs:*:*	-	Perform all operations on Elastic Volume Service (EVS). EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.
vpc:*:*	-	Perform all operations on Virtual Private Cloud (VPC), including ELBs. A cluster must run in a VPC. When creating a namespace, you need to create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.
sfs:*:get*	-	View Scalable File Service (SFS) resource details.
aom:*:get	-	View Application Operations Management (AOM) resource details.
aom:*:list	-	List AOM resources.
aom:autoScalingRule:*	-	Perform all operations on AOM auto scaling rules.

Permissions Granted by the CCE Viewer Role

The default role **CCE Viewer** has the following permissions:

Table 15-2 Permissions granted by the CCE Viewer role

Action	Action	Description
cce:*:get	cce:cluster:get	Query cluster details.
	cce:node:get	Query node details.
	cce:job:get	Query a specified cluster job.
cce:*:list	cce:cluster:list	List all clusters.
	cce:node:list	List all nodes.
	cce:job:list	List all cluster jobs.
cce:kubernetes:*	-	Perform operations on all Kubernetes resources. For details, see **Namespace-level Permissions**.
ecs:*:get	-	View details of all Elastic Cloud Servers (ECSs). In CCE, a node is an ECS with multiple EVS disks.
ecs:*:list	-	List ECS resources.
evs:*:get	-	View details of all EVS disk resources. EVS disks can be attached to cloud servers and scaled to a higher capacity whenever needed.
evs:*:list	-	List all EVS resources.
evs:*:count	-	-
vpc:*:get	-	View details of all VPC resources (including ELBs). A cluster must run in a VPC. When creating a namespace, you need to create or associate a VPC for the namespace so that all containers in the namespace will run in the VPC.
vpc:*:list	-	List all VPC resources (including enhanced load balancers).
sfs:*:get*	-	View Scalable File Service (SFS) resource details.
aom:*:get	-	View AOM resource details.
aom:*:list	-	List all AOM resources.

Action	Action	Description
aom:autoScalingRule:*	-	Perform all operations on AOM auto scaling rules.

15.3.3 Namespace-Level Permissions Management (By Using Kubernetes RBAC Authorization)

RBAC is a method of regulating access to computer or network resources based on the roles of individual users within an organization. The RBAC API declares four kinds of Kubernetes objects: Role, ClusterRole, RoleBinding, and ClusterRoleBinding, which are described as follows:

- **Role and ClusterRole**: An RBAC Role or ClusterRole contains rules that represent a set of permissions. If you want to define a role within a namespace, use a Role; if you want to define a role cluster-wide, use a ClusterRole.
- **RoleBinding** and **ClusterRoleBinding**: A role binding grants the permissions defined in a role to a user or set of users. It holds a list of subjects (users, groups, or service accounts), and a reference to the role being granted. A RoleBinding grants permissions within a specific namespace whereas a ClusterRoleBinding grants that access cluster-wide. If you want to bind a ClusterRole to all the namespaces in your cluster, use a ClusterRoleBinding.

Table 15-3 Four objects declared by the RBAC API

Type	Description
Role	A Role can be used to define permissions within a particular namespace.
ClusterRole	A ClusterRole can be used to grant the same permissions as a Role. Because ClusterRoles are cluster-scoped, you can also use them to grant access to: ● cluster-scoped resources (such as nodes) ● non-resource endpoints (such as /healthz) ● namespaced resources (such as pods) across all namespaces. For example, you can use a ClusterRole to allow a particular user to run **kubectl get pods --all-namespaces**.
RoleBinding	A RoleBinding maps a Role to a subject (a user or set of users), granting that Role's permissions to those users for accessing the resources in that namespace.
ClusterRoleBinding	A ClusterRoleBinding allows users to be granted a ClusterRole for authorization across the entire cluster (all namespaces).

For more information about Kubernetes RBAC authorization, see **Using RBAC Authorization**.

CCE Namespace Permissions

Tenants regulate users' or user groups' access to Kubernetes resources in a single namespace based on their Kubernetes RBAC roles.

Currently, there are three roles: **admin**, **edit**, and **view**. For details, see **Table 15-4**.

Table 15-4 User/user group roles

Default ClusterRole	Description
admin	Allows admin access, intended to be granted within a namespace using a RoleBinding. If used in a RoleBinding, allows read/write access to most resources in a namespace, including the ability to create roles and role bindings within the namespace. It does not allow write access to resource quota or to the namespace itself.
edit	Allows read/write access to most objects in a namespace. It does not allow viewing or modifying roles or role bindings.
view	Allows read-only access to see most objects in a namespace. It does not allow viewing roles, role bindings, or secrets.

15.4 Granting IAM Users the Permissions to Access CCE

15.4.1 Granting Cluster-Level Permissions (IAM Fine-Grained Authorization)

This topic describes how to use a group to grant permissions to an IAM user. **Process Flow** shows the process for granting permissions. By way of example, the CCE Viewer role will be granted to the user group Developers and the IAM user James will be added to the user group so that James will inherit the read-only access to CCE.

NOTICE

CCE provides two default roles: CCE Admin and CCE Viewer. It is advisable to grant default roles rather than custom roles. Note that role and permissions policy are interchangeable in this chapter. Both represent a set of permissions.

Configuration

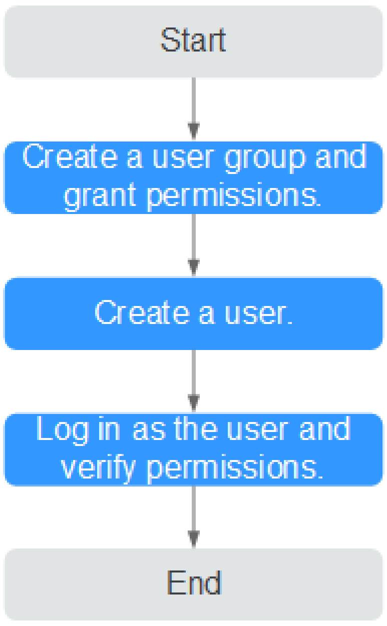
- You need a master account. Only the master account user or an IAM user who has been granted the CCE Administrator permission and has the global Security Administrator policy configured has the permission to access the CCE **Permissions Management** page to grant permissions to other IAM users.
- Only accounts can grant IAM users the permissions required for reading and writing IAM resources.
- To ensure IAM user security, if permissions management on the CCE console involves changing permissions of an IAM user, you must manually change the permissions through the IAM console according to on-screen instructions.

Process Flow

User groups facilitate centralized user management and streamlined permissions management. Users in the same user group have the same permissions. IAM users inherit permissions from the groups to which they belong.

To create a user group and grant it permissions, perform the following steps:

Figure 15-3 Process for granting cluster-level permissions (fine-grained policies)


Start
Create a user group and grant permissions.
Create a user.
Log in as the user and verify permissions.
End

Step 1: Create a User Group and Grant Permissions

Create a user group on the IAM console, and grant the **CCE Viewer** role to the group.

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

Step 2 Click the **Cluster-Level Permissions** tab and then **Create User Group**.

Step 3 Enter the user group name and optimally group description. Then, click **OK**. The name **Developers** is used as an example.

The user group is displayed in the user group list.

Step 4 In the same row as the user group, click **Modify**. In the **Group Permissions** area, click **Attach Policy** next to the region for which you want to assign permissions to the user group.

CCE is a project-level service. Therefore, you need to assign permissions in the projects in which users in the group need to access CCE resources.

Step 5 In the **Attach Policy** dialog box, search for and select **CCE Viewer**.

Step 6 Click **OK**.

----End

Step 2: Create an IAM User and Add It to the Group

IAM users can be created for employees or applications of an enterprise. Each IAM user has their own security credentials, and inherits permissions from the groups it is a member of. To create an IAM user, perform the following steps:

Step 1 Log in to the IAM console. In the navigation pane, choose **Users**. Then, click **Create User**.

Step 2 Set user information and click **Next**.

- **Username**: Used for logging in to the cloud. For this example, enter **James001**.
- **Credential Type**: Identity credential for authentication. For this example, select **Password**.
  - **Password**: Used for accessing the cloud using the console or development tools (including APIs, CLI, and SDKs).
  - **Access Key**: Used for logging to the cloud using development tools. This credential type is more secure, and is recommended if the IAM user does not need to use the console.
- (Optional) **User Groups**: Select **Developers**. The IAM user will inherit the permissions granted to the user group. The default user group is **admin**, which has the administrator permissions and all of the permissions required to use all cloud resources.
- (Optional) **Description**: Description of the IAM user.

Step 3 Click **Next**. In the dialog box that is displayed, set the parameters.

- **Password Type**: Three password types are available. In this example, **Set manually** is selected.
  - **Set by user**: Select this option if you are not the entity using the IAM user **James001**. James will receive a one-time login URL by email and can set a password at first login.
  - **Automatically generated**: Select this option if James accesses the cloud using a development tool. The cloud will generate a random 10-digit password.
  - **Set manually**: Select this option if you are the entity using the IAM user **James001**. Then set a password for login.
- **Password Reset Required**: By default, this option is selected, which means that the user will be required to set a new password at next login. In this example, this option is selected.

- **Email**: This parameter is configurable only if **Password Type** is **Set now**.
- (Optional) **Mobile Number**: Enter a mobile number.
- **Confirm Password**: Enter the initial password of the IAM user James001.

Step 4 Click **OK**.

----End

Step 3: Log In and Verify Permissions

After the IAM user is created, use the username and identity credential to log in to CCE console, and verify that the IAM user has the permissions defined by the **CCE Viewer** policy.

Step 1 On the login page, click **API Login** in the upper right corner.

Step 2 Enter the account name, username, and password, and click **Log In**.

- The account name is the name of the cloud account to which the IAM user belongs.
- The username and password are those set by the account when creating the IAM user James001. You will be prompted to change the initial password at initial login.

If the login fails, contact the entity owning the account to verify the username and password. Alternatively, you can reset the password.

Step 3 After successful login, switch to a region where you have been granted permissions on the console.

Step 4 Choose **Service List** > **Cloud Container Engine** to launch the CCE console. Then verify James001' s cluster permissions.

----End

15.4.2 Granting Namespace-Level Permissions (Kubernetes RBAC Authorization)

This section describes how to grant CCE users and user groups the permissions to various namespaced resources. Namespace-level authorization is more fine-grained than cluster-level authorization. Users and user groups with both a cluster-wide role and a namespace-wide role can perform operations on a cluster as well as on specified namespaces in the cluster. **Process Flow** describes the process for granting permissions.

Configuration

- You need a cloud account that has created one or more user groups and IAM users.
- In this example, both a user group and a user are granted permissions to access namespaced resources. You have the choice to grant permissions to either users or user groups.
- The process flow is used only to add namespace permissions policies for users or user groups who never have such policies. To edit permissions policies of

users or user groups, click **Edit** on the **Permissions Management** > **Namespace-Level Permissions** page.

- If multiple permissions policies are granted to a user or user group, all of these policies will take effect at the same time. The permissions policies granted to a user group apply to all users in the user group.

Constraints

Kubernetes RBAC authorization can be used for CCE clusters of v1.13.

Only CCE clusters of v1.11.7-r2 and later support Kubernetes RBAC authorization. By default, Kubernetes RBAC authorization is enabled in container clusters of v1.11.7-r2. For details on cluster version upgrade, see **Upgrading a Cluster**.

Table 15-5 User permissions comparison

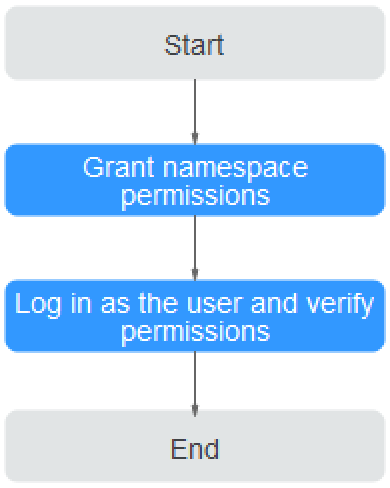
User Type	Cluster Version Earlier than v1.11.7-r2	Cluster Version of v1.11.7-r2 or Later
Account	All permissions	All permissions
IAM user with the CCE Administrator role	All permissions	All permissions
IAM user with the CCE Admin or CCE Viewer role	All permissions	Namespace permissions configured on the **Permissions Management** page
IAM user who is granted the Tenant Guest role and belongs to an account for which fine-grained access control is disabled	Read-only permission	Read-only permission
IAM user who is granted the Tenant Guest role and belongs to an account for which fine-grained access control is enabled	All permissions **NOTE** Enabling fine-grained access control will change read-only permissions to all permissions.	Namespace permissions configured on the **Permissions Management** page

Process Flow

A namespace is an abstract collection of resources and objects. Multiple namespaces can be created in a cluster. Data is isolated between namespaces so namespaces can share the same cluster service while not affecting each other. A namespace can act as a virtual cluster to meet diversified requirements.

This section describes how to grant namespace-level permissions to the IAM user James001 and the user group Developers created in **Granting IAM Users the Permissions to Access CCE**.

Figure 15-4 Process for granting namespace permissions


Start
Grant namespace permissions
Log in as the user and verify permissions
End

Step 1: Grant Namespace Permissions to an IAM User Or User Group

In this step, the IAM user James001 and user group Developers will be granted permissions to operate on resources in a cluster's namespace.

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

Step 2 On the displayed page, click the **Namespace-Level Permissions** tab. In the upper right corner of the namespace permissions list, select the cluster that contains the namespace whose access will be managed, and click **Add Permissions**.

Step 3 On the **Add Permissions** page, confirm the cluster name, and select a namespace whose access will be managed. By way of example, namespace **default** is selected.

Step 4 Click **Add Policy for Users**, and add the **admin** permissions policy for the user group **Developers**.

- **User/User Group**: Select **User group** from the drop-down list and then select **Developers** from the user group list.
- **Permissions Policy**: Select **admin**.

Step 5 Click **Add Policy for Namespaces**. Click **Add Policy for Users**, and add the **dev-test** permissions policy for the user James001.

- **Namespaces**: Select **dev-test**.
- **User/User Group**: Select **User** from the drop-down list and then select **James001** from the user list.
- **Permissions Policy**: Select **view**.

Step 6 Click **Create**. The namespace permissions granted to the user group and user appear in the permissions list.

NOTE

To sum up, the authorization result is as follows:

- The user group Developers has **admin** access to the namespace **default**. This **admin** access also applies to the IAM user James001 in the user group Developers.
- The IAM user James001 has **view** access to the namespace **dev-test**.

----End

Step 2: Log In and Verify Permissions

Use the username James001 and identity credential to log in to the CCE console, and verify that the IAM user James001 has the namespace permissions.

Step 1 Enter the account name, username, and password, and click **Log In**.

- The account name is the name of the cloud account to which the IAM user belongs.
- The username and password are those set by the account when creating the IAM user James001. You will be prompted to change the initial password at initial login.

If the login fails, contact the entity owning the account to verify the username and password. Alternatively, you can reset the password.

Step 2 After successful login, switch to a region where you have been granted permissions on the console.

Step 3 Choose **Service List** > **Cloud Container Engine** to launch the CCE console. Then verify James001' s namespace permissions.

----End

16 Configuration Center

16.1 Creating a ConfigMap

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 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 your configurations files to containers.

Preparations

Cluster and node resources have been created. For more information, see **Creating a Hybrid Cluster**. If clusters and nodes are available, you need not configure them again.

Creating a ConfigMap

Step 1 Log in to the CCE console. In the navigation pane, choose **Configuration Center** > **ConfigMaps**, and click **Add ConfigMap**.

Step 2 You can create a ConfigMap directly or based on YAML. If you want to create a ConfigMap based on YAML, go to **Step 4**.

Step 3 Method 1: Create a ConfigMap directly.

Set the parameters listed in **Table 16-1**.

Table 16-1 Parameters for creating a ConfigMap

Parameter	Description
Configuration Name	Name of a ConfigMap, which must be unique in a namespace.
Cluster	Cluster where the ConfigMap will be used.
Cluster Namespace	Namespace to which the ConfigMap belongs. If this parameter is left unspecified, the namespace **default** is used.
Description	Description of the ConfigMap.
Configuration data	The workload configuration data can be used in a container or used to store the configuration data. **Key** indicates the file name, and **Value** indicates the file content. 1. Click **Add ConfigMap Data**. 2. Set **Key** and **Value**.
Labels	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. 1. Click **Add Label**. 2. Set **Key** and **Value**.

Step 4 Method 2: Create a ConfigMap based on YAML.

NOTE

To create a ConfigMap by uploading a file, ensure that a resource description file has been created. CCE supports files in JSON or YAML format. For more information, see **ConfigMap Requirements**.

Click **Create YAML** on the right of the page.

- Method 1: Import an orchestration file.

  Click **Add File** to import the file in YAML or JSON format. The orchestration content can be directly displayed.
- Method 2: Directly orchestrate the content.

  In the orchestration content area, enter the content of the YAML or JSON file.

Step 5 After the configuration is complete, click **Create**.

The new ConfigMap is displayed in the ConfigMap list.

----End

ConfigMap Requirements

A ConfigMap resource file can be in JSON or YAML format, and the file size cannot exceed 2 MB.

- JSON format

  The following shows a configuration example of a ConfigMap resource file named **configmap.json**:

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "paas-broker-app-017",
    "namespace": "test",
   "enable": true
  },
  "data": {
    "context": "{\"applicationComponent\":{\"properties\":{\"custom_spec\":{}},\"node_name\":\"paas-broker-app\",\"stack_id\":\"0177eae1-89d3-cb8a-1f94-c0feb7e91d7b\"},\"softwareComponents\":[{\"properties\":{\"custom_spec\":{}},\"node_name\":\"paas-broker\",\"stack_id\":\"0177eae1-89d3-cb8a-1f94-c0feb7e91d7b\"}]}"
 }
}
```

- YAML format

  The following shows a configuration example of a ConfigMap resource file named **configmap.yaml**:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
data:
  data-1: value-1
  data-2: value-2
```

Creating a ConfigMap Using kubectl

Step 1 Configure kubectl. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Create and edit the **cce-configmap.yaml** file.

vi cce-configmap.yaml

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

Step 3 Creating a ConfigMap

kubectl create -f cce-configmap.yaml

kubectl get cm

```
NAME              DATA          AGE
cce-configmap       3           3h
cce-configmap1      3            7m
```

----End

Related Operations

After creating a ConfigMap, you can update or delete it as described in **Table 16-2**.

Table 16-2 Other operations

Operation	Description
Viewing the YAML file	Click **YAML** in the row where the target ConfigMap resides to view its YAML file.
Updating the ConfigMap	1. Click **Update** in the row where the target ConfigMap resides. 2. Modify the configuration according to **Table 16-1**. 3. Click **Update**.
Deleting the ConfigMap	Click **Delete** in the row where the target ConfigMap resides. Delete the ConfigMap as prompted.

16.2 Using a ConfigMap

After a ConfigMap is created, it can be used in three workload scenarios: environment variables, command line parameters, and data volumes.

- **Using a ConfigMap to Set Workload Environment Variables**
- **Using a ConfigMap to Set Command Line Parameters**
- **Mounting a ConfigMap to the Workload Data Volume**

The following example shows how to use a ConfigMap.

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

NOTICE

When a ConfigMap is used in a pod, the pod and ConfigMap must be in the same cluster and namespace.

Using a ConfigMap to Set Workload Environment Variables

When creating a workload, you can use a ConfigMap to set an environment variable. The **valueFrom** parameter indicates the key-value pair to be referenced.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-1
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:                          ## Use valueFrom to specify the value to be referenced.
            configMapKeyRef:
              name: cce-configmap                ## Name of the referenced configuration file.
              key: SPECIAL_LEVEL                ## Key of the referenced ConfigMap.
  restartPolicy: Never
```

If you need to define the values of multiple configuration items as the environment variables of the pods, add multiple environment parameters to the pods.

```
env:
- name: SPECIAL_LEVEL_KEY
  valueFrom:
?configMapKeyRef:
          name: cce-configmap
          key: SPECIAL_LEVEL
- name: SPECIAL_TYPE_KEY
  valueFrom:
?configMapKeyRef:
          name: cce-configmap
          key: SPECIAL_TYPE
```

To add all data in a configuration item to environment variables, use the envFrom parameter. The keys in the configuration item will become names of environment variables in pods.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-2
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
      - configMapRef:
          name: cce-configmap
  restartPolicy: Never
```

Using a ConfigMap to Set Command Line Parameters

You can use a ConfigMap to set commands or parameter values for a container by using the environment variable substitution syntax $(VAR_NAME). The following shows an example.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-3
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: cce-configmap
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: cce-configmap
          key: SPECIAL_TYPE
  restartPolicy: Never
```

After the pod runs, the following information is displayed:

```
Hello CCE
```

Mounting a ConfigMap to the Workload Data Volume

You can mount a ConfigMap to a workload when creating the workload so that the ConfigMap can be used in the data volume. After the mounting is complete, a configuration file with key as the file name and value as the file content is generated.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod-4
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ &quot;/bin/sh&quot;, &quot;-c&quot;, &quot;ls /etc/config/&quot; ]   ## Lists the names of 
the files in the directory.
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config                    ##Mounting to the /etc/config directory
  volumes:
    - name: config-volume
      configMap:
        name: cce-configmap
  restartPolicy: Never
```

After the pod is run, the **SPECIAL_LEVEL** and **SPECIAL_TYPE** files are generated in the **/etc/config** directory. The contents of the files are Hello and CCE, respectively. Also, the following file names will be displayed.

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

To mount a ConfigMap to a data volume, you can also perform operations on the CCE console. When creating a workload, set advanced settings for the container, choose **Data Storage** > **Local Volume**, click **Add Local Volume**, and select **ConfigMap**. For details, see **ConfigMap**.

16.3 Creating a secret

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.

Preparations

Cluster and node resources have been created. For more information, see **Creating a Hybrid Cluster**. If clusters and nodes are available, you need not configure them again.

Creating a Secret

Step 1 Log in to the CCE console. In the navigation pane, choose **Configuration Center** > **Secrets**. Click **Create Secret**.

Step 2 You can create a secret directly or based on YAML. If you want to create a secret based on YAML, go to **Step 4**.

Step 3 Method 1: Create a secret directly.

Set the basic information by referring to **Table 16-3**.

Table 16-3 Parameter description

Parameter	Description
Name	Name of a secret, which must be unique in the same namespace.
Cluster	Cluster where the secret will be used.
Namespace	Namespace to which the secret belongs. The default value is **default**.
Description	Description of the secret.
Secret Type	Type of the secret. ● Opaque: common secret. ● kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository. ● IngressTLS: a secret that stores the certificate required by ingresses (layer-7 load balancing Services). ● Other: another type of secret, which is specified manually.
Secret Data	Workload secret data can be used in containers. ● If the secret is of the Opaque type: 1. Click **Add Data**. 2. Enter a key and a value. The value must be based on the Base64 coding method. For details about the method, see **Base64 Encoding**. ● If the secret type is kubernetes.io/dockerconfigjson, enter the account and password of the private image repository. ● If the secret type is 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.

Parameter	Description
Secret Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define the identifiable attributes of these objects and are used to manage and select the objects. 1. Click **Add Label**. 2. Enter a key and a value.

Step 4 Method 2: Create a secret based on the YAML file.

NOTE

To create a secret by uploading a file, ensure that a resource description file has been created. CCE supports files in JSON or YAML format. For more information, see **Secret Resource File Configuration**.

You can import or directly write the file content in YAML or JSON format.

- Method 1: Import an orchestration file.

  Click **Add File** to import the file in YAML or JSON format. The orchestration content can be directly displayed.
- Method 2: Directly orchestrate the content.

  In the orchestration content area, enter the content of the YAML or JSON file.

Step 5 Click **Create** after the configuration is complete.

The new secret is displayed in the secret list.

----End

Secret Resource File Configuration

This section provides a configuration example of a secret resource file.

For example, you can retrieve the username and password for a workload through a secret.

- YAML format

  The content in the secret file **secret.yaml** is as follows. The value 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:
  username: my-username  #Username
  password: ******  #The value must be encoded using Base64.
type: Opaque     #You are advised not to change this parameter value.
```

- JSON format

  The content in the secret file **secret.json** is as follows:

```
{
  "apiVersion": "v1",
  "kind": "Secret",
  "metadata": {
    "name": "mysecret",
    "namespace": "default"
  },
  "data": {
    "username": "my-username",
    "password": "******"
  },
  "type": "Opaque"
}
```

Creating a Secret Using kubectl

Step 1 Configure kubectl. For details, see **Connecting to a Cluster Through kubectl**.

Step 2 Create and edit the cce-secrets.yaml file based on the Base64 encoding method.

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

vi cce-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: admin
  password: ******
```

Step 3 Create a secret.

kubectl create -f cce-secret.yaml

You can query the secret after creation.

kubectl get secret

----End

Related Operations

After a secret is created, you can perform the operations described in **Table 16-4**.

NOTE

The secret list contains system-defined secrets that can only be viewed but cannot be updated or deleted.

Table 16-4 Other operations

Operation	Description
Viewing the YAML file	Click **YAML** in the row where the target secret resides to view its YAML file.
Updating a secret	1. Click **Update** in the row where the target secret resides. 2. Modify the secret information according to **Table 16-3**. 3. Click **Update**.

Operation	Description
Deleting the secret	Click **Delete** in the row where the target secret resides. Delete the secret as prompted.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete** above the secret list. 3. Delete the secret as prompted.

Base64 Encoding

To perform Base64 encoding on a character 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
******
```

16.4 Using a Secret

After secrets are created, they can be mounted as data volumes or be exposed as environment variables to be used by a container in a pod.

NOTICE

The following keys are used by the CCE system. Do not perform any operation on them.

- Do not operate secrets under kube-system.
- Do not operate default-secret and paas.elb in other namespaces. The default-secret is used to pull the private image of SWR, and the paas.elb is used to connect the service in the namespace to the ELB service.

- **Using Secrets as Data Volumes In a Pod**
- **Using Secrets as Environment Variables in a Pod**

This section uses the following secret as an example to describe how to use a secret.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: my-username  #Username
  password: ******  #The value must be encoded using Base64.
```

NOTICE

When a secret is used in a pod, the pod and secret must be in the same cluster and namespace.

Using Secrets as Data Volumes In a Pod

A secret can be used as a file in a pod. In the following example, the username and password in the **mysecret** secret are saved in the **/etc/foo** directory as files.

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```

In addition, you can specify the directory and permission to access a secret. The username is stored in the **/etc/foo/my-group/my-username** directory of the container.

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
  volumes:
  - name: foo
    secret:
      secretName: mysecret
      items:
      - key: username
        path: my-group/my-username
        mode: 511
```

To mount a secret to a data volume, you can also perform operations on the CCE console. When creating a workload, set advanced settings for the container, choose **Data Storage > Local Disk**, click **Add Local Disk**, and select **Secret**. For details, see **Secret**.

Using Secrets as Environment Variables in a Pod

A secret can be used as an environment variable in a pod. In the following example, the username and password of the **mysecret** secret are defined as an environment variable of the pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
  restartPolicy: Never
```

17 Related Services

17.1 Software Repository for Container (SWR)

The image repository feature is provided by the Software Repository for Container (SWR) service for you to easily store, manage, and deploy Docker container images.

Using an Image

When creating a workload on CCE, you can choose pushed images from **My Images**. This section describes how to choose a pushed image when creating a workload.

Step 1 Log in to the CCE console. In the navigation pane, choose **Workloads** > **Deployments**. On the page displayed, click **Create Deployment**.

Step 2 Set the following parameters and retain the default values for other parameters:

- **Workload Name**: Set it to **game**.
- **Cluster Name**: Select the cluster where the workload is to run.
- **Pods**: Set it to **1**.

Step 3 Click **Next** to add a container.

Click **Add Container**. On the **My Images** tab page, select a pushed image and click **OK**.

Step 4 Complete the workload creation process by referring to **Creating a Deployment** or **Creating a StatefulSet**.

----End

17.2 Application Operations Management (AOM)

After creating workloads on CCE, you can operate and maintain the workloads on AOM.

You are advised to configure alarm monitoring on AOM for the workloads you created on CCE. If alarm monitoring is not configured, you cannot receive alarms when workloads are faulty and need to manually inspect the environment.

AOM is a one-stop and multi-dimensional O&M management platform for cloud applications. It monitors applications and related cloud resources in real time, collects and associates resource metrics, logs, and events to analyze application health status, and provides flexible alarm reporting and data visualization. With AOM, you can detect and locate faults in a timely manner and monitor running status of applications, resources, and services.

Common Operations

- Create **threshold** rules for metrics of these resources to monitor changes of certain resources.
- Use the dashboard to learn comprehensive information in real time during routine O&M. You can create and add concerned contents to the dashboard.
- Perform routine inspection on applications.

17.3 CTS

17.3.1 CCE operations that can be recorded by CTS

Cloud Trace Service (CTS) records operations on cloud service resources, allowing you to query, audit, and backtrack the resource operation requests initiated from the IEF console or open APIs as well as responses to the requests.

Table 17-1 CCE operations that can be recorded by CTS

Operation	Description
createCluster	Creating a cluster
updateCluster	Updating a cluster
deleteCluster	Deleting a cluster
createNode	Creating a node
addStaticNode	Adding a static node
updateNode	Updating a node
deleteOneHost	Deleting a host
deleteAllHosts	Deleting all hosts
suspendUserResource	Suspending user resources
createConfigmaps	Creating a ConfigMap
createDaemonsets	Creating a DaemonSet
createDeployments	Creating a Deployment

Operation	Description
createEvents	Creating an event
createIngresses	Creating an Ingress
createJobs	Creating a Job
createNamespaces	Creating a namespace
createNodes	Creating a node
createPersistentvolumeclaims	Creating a PersistentVolumeClaim
createPods	Creating a Pod
createReplicasets	Creating a ReplicaSet
createResourcequotas	Creating a resource quota
createSecrets	Creating a Secret
createServices	Creating a Service
createStatefulsets	Creating a StatefulSet
createVolumes	Creating a Volume
deleteConfigmaps	Deleting a ConfigMap
deleteDaemonsets	Deleting a DaemonSet
deleteDeployments	Deleting a Deployment
deleteEvents	Deleting an Event
deleteIngresses	Deleting an Ingress
deleteJobs	Deleting a Job
deleteNamespaces	Deleting a Namespace
deleteNodes	Deleting a node
deletePods	Deleting a Pod
deleteReplicasets	Deleting a replica set
deleteResourcequotas	Deleting a resource quota
deleteSecrets	Deleting a Secret
deleteServices	Deleting a Service
deleteStatefulsets	Deleting a StatefulSet
deleteVolumes	Deleting volumes
updateConfigmaps	Replacing a Specified ConfigMap
updateDaemonsets	Replacing a Specified DaemonSet

Operation	Description
updateDeployments	Replacing a Specified Deployment
updateEvents	Replacing a specified event
updateIngresses	Replacing a specified ingress
updateJobs	Replacing a Specified Job
updateNamespaces	Replacing a specified namespace
updateNodes	Replacing a specified node
updatePersistentvolumeclaims	Replacing a Specified PersistentVolumeClaim
updatePods	Replacing a Specified Pod
updateReplicasets	Replacing a specified ReplicaSet
updateResourcequotas	Replacing specified resource quotas
updateSecrets	Replacing a Specified Secret
updateServices	Replacing a Specified Service
updateStatefulsets	Replacing a specified StatefulSet
updateStatus	Replacing a specified status
uploadChart	Uploading a component template
updateChart	Updating a component template
deleteChart	Deleting a chart
createRelease	Creating a template application
updateRelease	Updating a template application
deleteRelease	Deleting a template application

17.3.2 Viewing CTS Traces

After you enable CTS, the system starts recording operations on CCE resources. Operation records of the last 7 days can be viewed on the CTS management console.

Scenario

After CTS is enabled, the system starts recording operations on CCE resources. Operation records for the last seven days can be viewed on the CTS console.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region.

Step 3 Click **Service List** and choose **Management & Deployment** > **Cloud Trace Service**.

Step 4 In the navigation pane of the CTS console, choose **Cloud Trace Service** > **Trace List**.

Step 5 Filter conditions to query traces. The following four filters are available:

- **Trace Source**, **Resource Type**, and **Search By**

  Select the desired value from the drop-down lists one by one. Set Trace Source to **CCE**.

  If you select **Trace name** from the **Search By** drop-down list, specify a trace name.

  If you select **Resource ID** for **Search By**, select or enter a specific resource ID.

  If you select **Resource name** for **Search By**, select or enter a specific resource name.

- **Operator**: Select an operator (at user level rather than account level).
- **Trace Status**: Available options include **All trace statuses**, **Normal**, **Warning**, and **Incident**. You can only select one of them.
- **Start Date** and **End Date**: You can specify the time period to query traces.

Step 6 Click the expanding button on the left expand trace details

Step 7 Click **View Trace** in the **Operation** column. In the **View Trace** dialog box , the trace structure is displayed.

----End

18 Using kubectl to Perform Operations on Clusters

18.1 Kubectl Usage Guide

Before running kubectl commands, you should have the kubectl development skills and understand the kubectl operations. For details, see **Kubernetes API** and **kubectl CLI**.

Table 18-1 kubectl usage guide

Category	kubectl Usage
Connecting to a cluster	**Connecting to a Kubernetes cluster using kubectl**
kube-dns HA	**Configuring high availability of kube-dns using kubectl**
Workload Creation	**Creating a Deployment using kubectl**
	Creating a StatefulSet using kubectl
Workload affinity/anti-affinity scheduling	**Example YAML for Running a Workload on a Specified Node**
	Example YAML for Refraining a Workload from Running on a Specified Node
	Example YAML for Co-locating Workloads on the Same Node
	Example YAML for Running Workloads on Different Nodes
	Example YAML for Running a Workload in a Specified AZ
	Example YAML for Refraining a Workload from Running in a Specified AZ

Category	kubectl Usage
Workload Access Mode Settings	**Implementing intra-cluster access using kubectl**
	Implementing Intra-VPC Access Using kubectl
	Implementing Layer 4 load balancing using kubectl
	Implementing EIP-based Access Using kubectl
	Implementing Layer 7 load balancing using kubectl
Advanced Workload Settings	**Example YAML for setting the container lifecycle**
Task management	**Creating a job using kubectl**
	Creating a cron job using kubectl
Configuration Center	**Creating a ConfigMap using kubectl**
	Creating a secret using kubectl
Storage Management	Creating an EVS disk using kubectl
	Attaching an EVS volume using kubectl
	Creating an SFS file system using kubectl
	Attaching an SFS volume using kubectl

18.2 Connecting to a Cluster Through kubectl

If you need to access the Kubernetes cluster from the client, you can use kubectl.

Preparations

CCE allows you to access a cluster through the VPC network or from a public network.

- Intra-VPC access: You need to apply for an ECS on the **ECS** console and ensure that the ECS is in the same VPC as the current cluster.
- Public network access: You need to prepare an ECS that can connect to a public network.

NOTICE

If public network access is used, the kube-apiserver of the cluster will be exposed to the public network and may be attacked. You are advised to configure Advanced Anti-DDoS for the EIP of the node where the kube-apiserver is located.

Procedure

Step 1 Log in to the CCE console, click **Resource Management** > **Clusters**, and choose **Command Line Tool** > **Kubectl** under the cluster to be connected.

Step 2 Connect to the cluster as prompted.

NOTE

- CCE allows a master account and its member accounts to download the config file (kubeconfig.json) separately. The config file downloaded by the member account is valid only for 30 days, whereas the one downloaded by the master account is valid permanently.
- The Kubernetes permissions of the config file downloaded by the member account are the same as those of the member account on the CCE console.

----End

Related Operations

After connecting to the cluster, you can use Kubernetes to manage workloads. For details, see **Kubectl Usage Guide**.

18.3 Configuring High Availability of kube-dns/ CoreDNS Using kubectl

Use the Kubernetes command line tool (kubectl) to configure high-availability of kube-dns/CoreDNS.

kube-dns/CoreDNS provides the Domain Name Service (DNS) for clusters. It is advised to configure multiple kube-dns/CoreDNS instances for a cluster. If there is only one kube-dns/CoreDNS in a cluster, the entire cluster will not run properly once the kube-dns/CoreDNS is down.

NOTE

- By default, CCE clusters of Kubernetes v1.11 and later versions have coredns installed.
- For more information about the DNS, see **coredns (System Resource Add-on, Mandatory)** or **Using Kubernetes In-Cluster DNS**.

Preparations

The cluster is accessible from a public network, or the cluster and the client are in the same VPC.

Procedure

Step 1 Log in to the CCE console, choose **Resource Management** > **Cluster Management**, and click **Kubectl** under the cluster to be connected.

Step 2 Set the API access mode for the cluster.

Step 3 Configure the kubectl.

After kubectl is successfully configured, you can use it to manually configure high availability of kube-dns/CoreDNS.

Step 4 Log in to the client.

Step 5 Edit the Deployment configuration file of kube-dns/CoreDNS.

The following uses the CoreDNS as an example:

kubectl edit deployment coredns -n kube-system

Change the value of **replicas** in the **spec** section in the file to the number of CoreDNS instances required.

Example:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: 2019-02-11T09:36:04Z
  generation: 1
  labels:
    app: coredns
    kubernetes-app: coredns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: CoreDNS
    release: cceaddon-coredns
  name: coredns
  namespace: kube-system
  resourceVersion: "1927"
  selfLink: /apis/extensions/v1beta1/namespaces/kube-system/deployments/coredns
  uid: 737b9296-2de0-11e9-b629-fa163e7fb882
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: coredns
      kubernetes-app: coredns
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      annotations:
        checksum/config: 3095a9b4028195e7e0b8b22c550bf183d0b7a8a7eba20808b36081d0b39f8b81
```

----End

18.4 Common kubectl Commands

Getting Started

get

The **get** command displays one or many resources of a cluster.

This command prints a table of the most important information about all resources, including cluster nodes, running pods, replication controllers, and services.

NOTICE

A cluster can have multiple namespaces. If no namespace is specified, this command will run with the **--namespace=default** flag.

Examples:

To list all pods with detailed information:

```
kubectl get po -o wide
```

To display pods in all namespaces:

```
kubectl get po --all-namespaces
```

To list labels of pods in all namespaces:

```
kubectl get po --show-labels
```

To list all namespaces of the node:

```
kubectl get namespace
```

NOTE

To list information of other nodes, run this command with the **-s** flag. To list a specified type of resources, add the resource type to this command, for example, **kubectl get rc**, **kubectl get svc**, **kubectl get nodes**, and **kubectl get deploy**.

To list a pod with a specified name in YAML output format:

```
kubectl get po <podname> -o yaml
```

To list a pod with a specified name in JSON output format:

```
kubectl get po <podname> -o json
kubectl get po rc-nginx-2-btv4j -o=custom-columns=LABELS:.metadata.labels.app
```

NOTE

LABELS indicates a comma separated list of user-defined column titles. **metadata.labels.app** indicates the data to be listed in either YAML or JSON output format.

create

The **kubectl** command creates a cluster resource from a file or input.

If there is already a resource descriptor (a YAML or JSON file), you can create the resource from the file by running the **kubectl create -f** *filename* command.

```
kubectl create -f filename
```

expose

The **expose** command exposes a resource as a new Kubernetes service. Possible resources include a pod, replication controller, service, and deployment.

```
kubectl expose deployment deployname --port=81 --type=NodePort --target-port=80 --name=service-name
```

NOTE

The example command creates a service of NodePort type for the deployment with the name specified in **deployname**. The service will serve on port 81 specified in **-port** and connect to the containers on port 80 specified in **-target-port**. More specifically, the service is reachable at <cluster-internal IP address>:<port>, and containers are reachable at <node IP address>:<target-port>.

run

Example:

To run a particular image in the cluster:

```
kubectl run deployname --image=nginx:latest
```

To run a particular image using a specified command:

```
kubectl run deployname -image=busybox --command -- ping baidu.com
```

set

The **set** command configures object resources.

Example:

To change the image of a deployment with the name specified in **deployname** to image 1.0:

```
kubectl set image deploy deployname containername=containername:1.0
```

edit

The **edit** command edits a resource from the default editor.

Examples:

To update a pod:

```
kubectl edit po po-nginx-btv4j
```

The example command yields the same effect as the following command:

```
kubectl get po po-nginx-btv4j -o yaml >> /tmp/nginx-tmp.yaml
vim /tmp/nginx-tmp.yaml
/*do some changes here */
kubectl replace -f /tmp/nginx-tmp.yaml
```

explain

The **explain** command gets documentation for a resource.

Example:

To get documentation of pods or services:

```
kubectl explain pods,svc
```

delete

The **delete** command deletes resources by resource name or label.

Example:

To delete a pod with minimal delay:

```
kubectl delete po podname --now
kubectl delete -f nginx.yaml
kubectl delete deployment deployname
```

Deployment Commands

rolling-update*

rolling-update is a very important command. It updates a running service with zero downtime. Pods are incrementally replaced by new ones. One pod is updated at a time. The old pod is deleted only after the new pod is up. New pods must be distinct from old pods by name, version, and label. Otherwise, an error message will be reported.

```
kubectl rolling-update poname -f newfilename
kubectl rolling-update poname -image=image:v2
```

If any problem occurs during the rolling update, run the command with the **-rollback** flag to abort the rolling update and revert to the previous pod.

```
kubectl rolling-update poname -rollback
```

rollout

The **rollout** command manages the rollout of a resource.

Example:

To check the rollout status of a particular deployment:

```
kubectl rollout status deployment/deployname
```

To check the rollout history of a particular deployment:

```
kubectl rollout history deployment/deployname
```

To roll back to the previous deployment (by default, a resource is rolled back to the previous version):

```
kubectl rollout undo deployment/test-nginx
```

scale

The **scale** command sets a new size for a resource by adjusting the quantity of resource replicas.

```
kubectl scale deployment deployname --replicas=newnumber
```

autoscale

The **autoscale** command automatically chooses and sets the quantity of pods. This command specifies the range for the number of pod replicas maintained by a replication controller. If there are too many pods, the replication controller terminates the extra pods. If there is too few, the replication controller starts more pods.

```
kubectl autoscale deployment deployname --min=minnumber --max=maxnumber
```

Cluster Management Commands

cordon, drain, uncordon*

If a node to be upgraded is running many pods or is already down, perform the following steps to prepare the node for maintenance:

Step 1 Run the **cordon** command to mark a node as unschedulable. This means that new pods will not be scheduled onto the node.

```
kubectl cordon nodename
```

Step 2 Run the **drain** command to smoothly migrate the running pods from the node to another node.

```
kubectl drain newnodename
```

Step 3 Perform maintenance operations on the node, such as upgrading the kernel and upgrading Docker.

Step 4 After node maintenance is completed, run the **uncordon** command to mark the node as schedulable.

```
kubectl uncordon nodename
```

----End

cluster-info

To check the add-ons running in the cluster:

```
kubectl cluster-info
```

To view the details:

```
kubectl cluster-info dump
```

top*

The **top** command displays resource (CPU/memory/storage) usage. This command requires Heapster to be correctly configured and working on the server.

taint*

The **taint** command updates the taints on one or more nodes.

certificate*

The **certificate** command modifies the certificate resources.

Fault Diagnosis and Debugging Commands

describe

The **describe** command is similar to the **get** command. The difference is that the **describe** command shows details of a specific resource or group of resources, whereas the **get** command lists one or more resources in a cluster. The **describe** command does not support the **-o** flag. For resources of the same type, resource details are printed out in the same format.

NOTE

If the information about a resource is queried, you can use the **get** command to obtain more detailed information. If you want to check the status of a specific resource, for example, to check if a pod is in the running state, run the **describe** command to show more detailed status information.

```
kubectl describe po <podname>
```

logs

The **logs** command prints logs for a container in a pod or specified resource to stdout. To display logs in the tail -f mode, run this command with the **-f** flag.

```
kubectl logs -f podname
```

exec

The kubectl **exec** command is similar to the Docker **exec** command and executes a command in a container. If there are multiple containers in a pod, use the **-c** flag to choose a container.

```
kubectl exec -it podname bash
kubectl exec -it podname -c containername bash
```

port-forward*

The **port-forward** command forwards one or more local ports to a pod.

Example:

To listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod:

```
kubectl port -forward podname 5000:6000
```

proxy*

The **proxy** command creates a proxy server between local host and the Kubernetes API server.

Example:

To enable the HTTP Rest interface on the master node:

```
kubectl proxy -accept-hosts= '.*' -port=8001 -address=' 0.0.0.0'
```

cp

The **cp** command copies files and directories to and from containers.

```
cp filename newfilename
```

auth*

The **auth** command inspects authorization.

attach*

The **attach** command is similar to the **logs -f** command and attaches to a process that is already running inside an existing container. To exit, run the **ctrl-c** command. If a pod contains multiple containers, to view the output of a specific container, use the **-c** flag preceded by podname to choose a container.

```
kubectl attach podname -c containername
```

Advanced Commands

replace

The **replace** command updates or replaces an existing resource by attributes including the number of replicas, label, image version, and port number. You can directly modify the original YAML file and then run the **replace** command.

```
kubectl replace -f filename
```

NOTICE

Resource names cannot be updated. After a pod label is updated, pods with the original label will fall out of the scope of the replication controller using the new label selector. The replication controller notices that some pods no longer match its new label selector and spun up a specified number of new pod replicas to replace original pods. By default, original pods with the original label are not deleted. In this case, if you run the **get po** command, you will find that the number of pods is doubled. The original pods are no longer controlled by the replication controller using the new label selector.

apply*

The **apply** command provides a more strict control on resource updating than **patch** and **edit** commands. The **apply** command applies a configuration to a resource and maintains a set of configuration files in source control. Whenever there is an update, the configuration file is pushed to the server, and then the kubectl **apply** command applies the latest configuration to the resource. The Kubernetes compares the new configuration file with the original one and updates only the changed configuration instead of the whole file. The configuration that is not contained in the **-f** flag will remain unchanged. Like the **replace** command, the **apply** command creates a resource without deleting the original resource. It updates the original resource. Similar to the git operation, the **apply** command adds an annotation to the resource to mark the current apply.

```
kubectl apply -f
```

patch

If you want to modify attributes of a running container without first deleting the container or using the **replace** command, the **patch** command is to the rescue. The **patch** command updates field(s) of a resource using strategic merge patch, a JSON merge patch, or a JSON patch. For example, you need to change a pod label from **app=nginx1** to **app=nginx2** while the pod is running.

```
kubectl patch pod podname -p '{"metadata":{"lables":{"app":"nginx1"}}}'
```

convent*

The **convert** command converts configuration files between different API versions.

Configuration Commands

label

The **label** command update labels on a resource.

```
kubectl label pods my-pod new-label=newlabel
```

annotate

The **annotate** command update annotations on a resource.

```
kubectl label pods my-pod icon-url=http://......
```

completion

The **completion** command provides autocompletion for shell.

Other Commands

api-versions

The **api-versions** command prints the supported API versions.

```
kubectl api-versions
```

api-resources

The **api-resources** command prints the supported API resources.

```
kubectl api-resources
```

config*

The **config** command modifies kubeconfig files. An example use case of this command is to configure authentication information in API calls.

help

The **help** command gets all command references.

version

The **version** command prints the client and server version information for the current context.

```
kubectl version
```

19 Reference

19.1 Checklist for Migrating Containerized Applications to the Cloud

Overview

Cloud Container Engine (CCE) provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. With CCE, you can easily deploy, manage, and scale containerized applications in the cloud.

This checklist describes the system availability, data reliability, and O&M reliability of migrating containerized applications to the cloud. It contains check items, impact, FAQs, and examples, helping you migrate services to CCE and avoid application exceptions or cluster reconstruction caused by improper use.

Check Items

Table 19-1 System availability

Category	Check Item	Type	Impact
Cluster	When creating a cluster, set **High Availability** to **Yes**.	Reliability	A cluster with **High Availability** set to **No** is a non-HA cluster with only one master. If the master node is faulty, the entire cluster will be unavailable. Therefore, you are advised to create an HA cluster in the production environment.

Category	Check Item	Type	Impact
	Before creating a cluster, determine the container network model that is suitable to the service scenario.	Network Planning	Different container network models apply to different scenarios.
	Before creating a cluster, plan the subnet CIDR block and container network CIDR block properly.	Network Planning	If the range of the subnet and container network CIDR blocks is not properly set, the number of available nodes in the cluster will be less than the number of nodes supported by the cluster. Network planning has different constraints on different container network models.
	Before creating a cluster, properly plan CIDR blocks for the related Direct Connect, peering connection, container network, service network, and subnet to avoid IP address conflicts.	Network Planning	If CIDR blocks are not properly set and IP address conflicts occur, service access will be affected.
Workload Management	When creating a workload, set the upper and lower limits of CPU and memory resources.	Confirming Deployment	If the upper and lower resource limits are not set for an application, a resource leak of this application will make resources unavailable for other applications deployed on the same node. In addition, applications that do not have upper and lower resource limits cannot be accurately monitored.
	When creating an application, set the number of pods to more than two and set the scheduling policy based on service requirements.	Reliability	A single-pod application will be faulty if the node or pod is faulty.

Category	Check Item	Type	Impact
	Properly set affinity and anti-affinity.	Reliability	If affinity and anti-affinity are both configured for an application that provides Services externally, Services may fail to be accessed after the application is upgraded or restarted.
	When creating a workload, set the health check policy, that is, set the workload liveness probe and the readiness probe.	Reliability	If the two probes are not set, pods cannot detect service exceptions or automatically restart the service to restore it. This results in a situation where the pod status is normal but the service in the pod is abnormal.
	When creating a workload, set the pre-stop processing command (**Lifecycle** > **Pre-Stop**) to ensure that the services running in the pods can be completed in advance in the case of application upgrade or pod deletion.	Reliability	If the pre-stop processing command is not configured, the pod will be directly killed and services will be interrupted during application upgrade.
	When creating a Service, select an access mode based on service requirements. Currently, the following types of access modes are supported: intra-cluster access, intra-VPC access, and external access.	Confirming Deployment	If the access mode is not properly set, internal and external access may be in disorder and resources may be wasted.

Table 19-2 Data reliability

Category	Check Item	Type	Impact
Container data persistency	Store application data in the cloud, rather than on a local disk.	Reliability	If a node is faulty and cannot be restored, data on the local disk cannot be restored.
Backup	Back up application data.	Reliability	Data cannot be restored after being lost.

Table 19-3 O&M reliability

Category	Check Item	Type	Impact
Project	The quotas of ECS, VPC, subnet, EIP, and EVS resources must meet customer requirements.	Confirming Deployment	If the quota is insufficient, resources will fail to be created. Specifically, users who have configured automatic capacity expansion must have sufficient resource quotas.
	Do not install private software or modify OS configurations on a cluster node.	Confirming Deployment	If private software is installed on a cluster node or OS configurations are modified, exceptions may occur on Kubernetes components on the node, making it unavailable for application deployment.
	Do not modify information about resources created by CCE, such as security groups and EVS disks. Resources created by CCE are labeled **cce**.	Confirming Deployment	CCE cluster functions may be abnormal.
Proactive O&M	Configure alarm monitoring on AOM for the applications you deployed in CCE clusters.	Monitoring Label Specifications	If alarm monitoring is not configured, you cannot receive alarms when applications are faulty and need to manually locate the faults.

19.2 Creating a Linux LVM Disk Partition for Docker

This section describes how to check whether there are **available raw disks** and **Linux LVM disk partitions** and how to create Linux LVM disk partitions.

Preparations

To improve the system stability, attach a data disk to Docker and use the direct-lvm mode.

Procedure

Step 1 Check whether available raw disks exist on the current node.

1. Log in to the target node as the **root** user.
2. Check the raw disk device.

lsblk -l | grep disk

If the following information is displayed, the raw disks named **xvda** and **xvdb** exist on the node.

```
xvda  202:0    0   40G  0 disk
xvdb  202:16   0  100G  0 disk
```

3. Check whether the raw disk is in use.

   lsblk /dev/*<devicename>*

   devicename indicates the raw disk name, for example, **xvda** and **xvdb** in the previous step.

   Run the **lsblk /dev/xvda** and **lsblk /dev/xvdb** commands. If the following information is displayed, **xvda** has been partitioned and used while **xvdb** is available. If no raw disk is available, bind an EVS disk to the node. It is recommended that the disk space be no less than 80 GB.

```
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   40G  0 disk
├─xvda1 202:1    0  100M  0 part /boot
└─xvda2 202:2    0 39.9G  0 part /
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvdb 202:16   0  100G  0 disk
```

Step 2 Check whether there are partitions available. Currently, only Linux LVM partitions are supported.

1. Log in to the target node as the **root** user.
2. Check the partition whose system type is Linux LVM.

   sfdisk -l 2>>/dev/null| grep "Linux LVM"

   If the following information is displayed, two Linux LVM partitions, **/dev/nvme0n1p1** and **/dev/nvme0n1p2**, exist in the system.

```
/dev/nvme0n1p1          1  204800  204800  209715200   8e  Linux LVM
/dev/nvme0n1p2     204801  409600  204800  209715200   8e  Linux LVM
```

3. Check whether the partition is in use.

   lsblk ***<partdevice>***

   <partdevice> is the Linux LVM partition found in the previous step.

   In this example, run the **lsblk/dev/nvme0n1p1** and **lsblk/dev/nvme0n1p2** commands. If the following information is displayed, partition **nvme0n1p** is in use while **nvme0n1p2** is available.

```
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
nvme0n1p1                   259:3    0  200G  0 part
└─vgpaas-thinpool_tdata   251:8    0  360G  0 lvm
  └─vgpaas-thinpool       251:10   0  360G  0 lvm
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
nvme0n1p2 259:1    0  100G  0 part
```

   If no AZ is available, perform **Step 3** to create a partition for the docker.

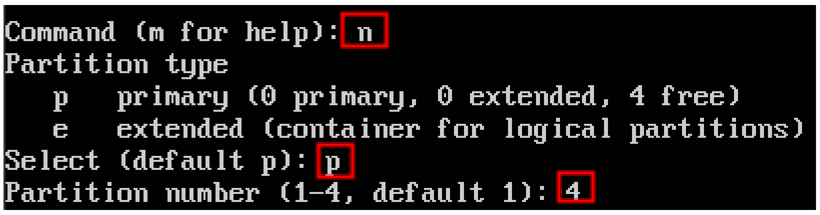
Step 3 Create a Linux LVM disk partition for the docker.

1. Run the following command to create a disk partition. **devicename** indicates the available raw disk name, for example, **xvdb** in **Step 1**.

   fdisk /dev/*devicename*

2. Enter **n** to create a new partition. Enter **p** to display the primary partition number. Enter **4** to indicate the fourth primary partition.

Figure 19-1 Creating a partition



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 4
```

3. Configure the start and last sectors as follows for example:

```
Start sector (1048578048-4294967295, 1048578048 by default):
1048578048
Last sector, +sector or size {K, M, or G} (1048578048-4294967294, 4294967294 by default): +100G
```

   This configuration indicates that partition 4 has been set to the Linux type and the size is 100 GiB.

4. Enter **t** to change the partition system type. Enter the hex code **8e** when prompted to change the system type to Linux LVM.

```
Command (enter m to obtain help): t
Partition ID (ranging from 1 to 4, 4 by default): 4
Hex code (enter L to list all codes): 8e
This configuration changes the type of the partition Linux to Linux LVM.
```

5. Enter **w** to save the modification.

```
Command (enter m to obtain help): w
The partition table has been altered!
```

6. Run the **partprobe** command to refresh the disk partition.

----End

19.3 Using Kubernetes In-Cluster DNS

Overview to Workload's DNS Configuration

Every Kubernetes cluster has a built-in DNS add-on (kube-dns/CoreDNS) to provide domain name resolution for workloads in the cluster. When handling a high concurrency of DNS queries, kube-dns/CoreDNS may encounter a performance bottleneck, that is, it may fail occasionally to fulfill DNS queries. There are cases when Kubernetes workloads initiate unnecessary DNS queries. This makes DNS overloaded if there are many concurrent DNS queries. Tuning DNS configuration for workloads will reduce the risks of DNS query failures to some extent.

For more information about the DNS, see **coredns (System Resource Add-on, Mandatory)** or **Configuring High Availability of kube-dns/CoreDNS Using kubectl**.

Configuration Options in the DNS Resolver Configuration File for Linux Operating Systems

Run the **cat /etc/resolv.conf** command on a Linux node or container to view the DNS resolver configuration file. The following is an example DNS resolver configuration of a container in a Kubernetes cluster:

```
nameserver 10.247.x.x
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

Configuration Options

- **nameserver**: an IP address list of a name server that the resolver will query. If this parameter is set to 10.247.x.x, the resolver will query the kube-dns/CoreDNS. If this parameter is set to another IP address, the resolver will query a cloud or on-premises DNS server.
- **search**: a search list for host-name lookup. When a domain name cannot be resolved, DNS queries will be attempted combining the domain name with each domain in the search list in turn until a match is found or all domains in the search list are tried. For CCE clusters, the search list is currently limited to three domains per container. When resolving a nonexistent domain name, eight DNS queries will be initiated because each domain in the search list will be queried, one for IPv4 and the other for IPv6.
- **options**: options that allow certain internal resolver variables to be modified. Common options include timeout, attempts, and ndots. The value **ndots:5** means that if a domain name has fewer than 5 dots (.), DNS queries will be attempted by combining the domain name with each domain in the search list in turn. If no match is found after all the domains in the search list are tried, the domain name is then used for DNS query. If the domain name has 5 or more than 5 dots, it will be tried first for DNS query. In case that the domain name cannot be resolved, DNS queries will be attempted by combining the domain name with each domain in the search list in turn. For example, the domain name www.console.prod-cloud-ocb.orange-business.com has only 2 dots (smaller than the value of ndots), and therefore the sequence of DNS queries is as follows: www.console.prod-cloud-ocb.orange-business.default.svc.cluster.local, www.console.prod-cloud-ocb.orange-business.com.svc.cluster.local, www.console.prod-cloud-ocb.orange-business.com.cluster.local, and www.console.prod-cloud-ocb.orange-business.com. This means that at least seven DNS queries will be initiated before the domain name is resolved into an IP address. It is clear that when many unnecessary DNS queries will be initiated to access an external domain name. There is room for improvement in workload's DNS configuration.

NOTE

For more information about configuration options in the resolver configuration file used by Linux operating systems, visit **https://man7.org/linux/man-pages/man5/resolv.conf.5.html**.

- **Configuration Options in the DNS Configuration File for Applications in Kubernetes Clusters**

  As explained earlier, applications may initiate unnecessary DNS queries in some scenarios. Kubernetes provides DNS-related configuration options for applications. The use of application's DNS configuration can effectively reduce unnecessary DNS queries in certain scenarios and improve service concurrency. In application configuration, there are two DNS-related fields: **dnsPolicy** and **dnsConfig**.

  dnsPolicy

  The **dnsPolicy** field is used to configure a DNS policy for an application. The default value is **ClusterFirst**. The DNS parameters in **dnsConfig** will be merged to the DNS file generated according to **dnsPolicy**. The merge rules are later explained in **dnsConfig** description. Currently, **dnsPolicy** supports the following four values:

  – ClusterFirst: CCE cluster's kube-dns/CoreDNS, which is cascaded with a cloud DNS by default, is used for workloads. Containers can resolve both the cluster-internal domain names registered by a Service and the external domain names exposed to public networks. The search list (**search** option) and **ndots: 5** are present in the DNS configuration file. Therefore, when accessing an external domain name and a long cluster-internal domain name (for example, kubernetes.default.svc.cluster.local), the search list will usually be traversed first, resulting in at least six invalid DNS queries. The issue of invalid DNS queries disappears only when a short cluster-internal domain name (for example, kubernetes) is being accessed.

– **ClusterFirstWithHostNet**: By default, the DNS configuration file that the kubelet's **--resolv-conf** flag points to is configured for workloads running with hostNetwork, that is, a cloud DNS is used for CCE clusters. If **dnsPolicy** is set to **ClusterFirstWithHostNet**, kube-dns/CoreDNS is used for workloads, and container's DNS configuration file is the same as **ClusterFirst**, eliminating the problem of invalid DNS queries.
– **Default**: Container's DNS configuration file is the DNS configuration file that the kubelet's **--resolv-conf** flag points to. In this case, a cloud DNS is used for CCE clusters. Both **search** and **options** fields are left unspecified. This configuration can only resolve the external domain names registered with the Internet, and not cluster-internal domain names. This configuration is free from the issue of invalid DNS queries.
– **None**: This value is introduced since Kubernetes v1.9 (Beta in v1.10). If **dnsPolicy** is set to **None**, the **dnsConfig** field must be specified because all DNS settings are supposed to be provided using the **dnsConfig** field.

dnsConfig

The **dnsConfig** field is used to configure DNS parameters for workloads. The configured parameters are merged to the DNS configuration file generated according to **dnsPolicy**. If **dnsPolicy** is set to **None**, the workload's DNS configuration file is specified by the **dnsConfig** field. If **dnsPolicy** is not set to **None**, the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated according to **dnsPolicy**.

– **nameservers**: a list of IP addresses that will be used as DNS servers. If workload's **dnsPolicy** is set to **None**, the list must contain at least one IP address, otherwise this property is optional. The servers listed will be combined to the nameservers generated from the specified DNS policy in **dnsPolicy** with duplicate addresses removed.
– **searches**: a list of DNS search domains for hostname lookup in the Pod. This property is optional. When specified, the provided list will be merged into the search domain names generated from the chosen DNS policy in **dnsPolicy**. Duplicate domain names are removed. Kubernetes allows for at most 6 search domains.
– **options**: an optional list of objects where each object may have a name property (required) and a value property (optional). The contents in this property will be merged to the options generated from the specified DNS policy in **dnsPolicy**. Duplicate entries are removed.

Configuration Examples

The following example describes how to configure DNS for workloads.

- **Use Case 1: Using kube-dns/CoreDNS Built in Kubernetes Clusters**

Scenario

Kubernetes in-cluster Kube-DNS/CoreDNS is applicable to resolving only cluster-internal domain names or cluster-internal domain names + external domain names. This is the default DNS for workloads.

Example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx
  dnsPolicy: ClusterFirst
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 2: Using a cloud DNS**

  Scenario

  A cloud DNS cannot resolve cluster-internal domain names and therefore is applicable to the scenario where workloads access only external domain names registered with the Internet.

  Example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx
  dnsPolicy: Default//The DNS configuration file that the kubelet's --resolv-conf flag points to is
used. In this case, a cloud DNS is used for CCE clusters.
```

  Container's DNS configuration file:

```
nameserver 10.125.x.x
```

- **Use Case 3: Using kube-dns/CoreDNS for Workloads Running with hostNetwork**

  Scenario

  By default, a cloud DNS is used for workloads running with hostNetwork. If workloads need to use kube-dns/CoreDNS, set **dnsPolicy** to **ClusterFirstWithHostNet**.

  Example:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

- **Use Case 4: Customizing Application's DNS Configuration**

  Scenario

  You can flexibly customize the DNS configuration file for applications. Using **dnsPolicy** and **dnsConfig** together can address almost all scenarios, including the scenarios in which an on-premises DNS will be used, multiple DNSs will be cascaded, and DNS configuration options will be modified.

  Example 1: Using Your On-Premises DNS

  *Set **dnsPolicy** to **None** so application's DNS configuration file is generated based on **dnsConfig**.*

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
    - 10.2.3.4 //IP address of your on-premises DNS
    searches:
    - ns1.svc.cluster.local
    - my.dns.search.suffix
    options:
    - name: ndots
      value: "2"
    - name: timeout
      value: "3"
```

  Container's DNS configuration file:

```
nameserver 10.2.3.4
search ns1.svc.cluster.local my.dns.search.suffix
options timeout:3 ndots:2
```

  Example 2: Modifying the ndots Option in the DNS Configuration File to Reduce Invalid DNS Queries

  Set **dnsPolicy** to a value other than **None** so the DNS parameters configured in **dnsConfig** are added to the DNS configuration file generated based on **dnsPolicy**.

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
  - name: test
    image: nginx
  dnsPolicy: "ClusterFirst"
  dnsConfig:
    options:
```

```
  - name: ndots
    value: "2" //Changes the ndots:5 option in the DNS configuration file generated based on
the ClusterFirst policy to ndots:2.
```

Container's DNS configuration file:

```
nameserver 10.247.3.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:2
```

19.4 How Do I Enable ICMP Security Group Rules?

If a workload uses UDP for both load balancing and health checking, enable ICMP security group rules for the nodes of the workload.

Procedure

Step 1 Log in to the ECS console, find the ECS corresponding to any node where the workload runs, and click the ECS name. The ECS details page is displayed.

Step 2 On the **Security Groups** tab page, click **Modify Security Group Rule**.

Step 3 Click **Add Rule** to add an inbound rule for the ECS.

- **Protocol/Application**: Select **ICMP**.
- **Port**: set to **All**.
- **Source**: Enter an IP address, for example, 192.168.0.22/32.

Step 4 When the configuration is complete, click **OK**.

----End

19.5 How Do I Troubleshoot Insufficient EIPs When a Node Is Added?

Bug Fixed

When a node is added, **EIP** is set to **Automatically assign**. The node cannot be created, and a message indicating that EIPs are insufficient is displayed.

Solutions

Two methods are available to solve the problem.

- Method 1. Unbind the VMs bound to EIPs and add a node again.
  a. Log in to the management console.
  b. Click **Service List** and choose **Computing** > **Elastic Cloud Server**.
  c. In the ECS list, locate the target ECS and click its name.
  d. On the ECS details page, click the **EIPs** tab. In the EIP list, click **Unbind** at the row of the target ECS and click **Yes**.
  e. Return to the **Create Node** page on the CCE console and click **Use existing** to add an EIP.

- Method 2: Increase the EIP quota.

  Quotas are used to limit the number of resources available to users. If the existing resource quota cannot meet your service requirements, you can apply for a higher quota.

19.6 How Do I Format a Data Disk Using Command Line Injection?

Before using command line injection, write a script that can format data disks and save it to your OBS bucket. Then, inject a command line that will automatically execute the disk formatting script when the node is up. Use input parameters to specify the size of each docker data disk (for example, the default docker disk of 100 GB and the additional disk of 110 GB) and the mount path (**/data/code**) of the additional disk. In this example, the script is named **formatdisk.sh**.

Example command line:

```
cd /tmp;curl -k -X GET OBS bucket address/formatdisk.sh -1 -O;fdisk -l;sleep 30;bash -x formatdisk.sh 100 /
data/code;fdisk -l
```

Example script (**formatdisk.sh**):

```
dockerdisksize=$1
mountdir=$2
systemdisksize=40
i=0
while [ 20 -gt $i ]; do
    echo $i;
    if [ $(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}' |wc -l) -
ge 3 ]; then
        break
    else
        sleep 5
    fi;
    i=$[i+1]
done
all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')
for device in ${all_devices[@]}; do
    isRawDisk=$(sudo lsblk -n $device 2>/dev/null | grep disk | wc -l)
    if [[ ${isRawDisk} > 0 ]]; then
        # is it partitioned ?
        match=$(sudo lsblk -n $device 2>/dev/null | grep -v disk | wc -l)
        if [[ ${match} > 0 ]]; then
            # already partited
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Raw disk ${device} has been partition, will skip
this device"
            continue
        fi
    else
        isPart=$(sudo lsblk -n $device 2>/dev/null | grep part | wc -l)
        if [[ ${isPart} -ne 1 ]]; then
            # not parted
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has not been partition, will skip this
device"
            continue
        fi
        # is used ?
        match=$(sudo lsblk -n $device 2>/dev/null | grep -v part | wc -l)
        if [[ ${match} > 0 ]]; then
            # already used
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
            continue
        fi
        isMount=$(sudo lsblk -n -o MOUNTPOINT $device 2>/dev/null)
        if [[ -n ${isMount} ]]; then
            # already used
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
            continue
        fi
        isLvm=$(sudo sfdisk -lqL 2>>/dev/null | grep $device | grep "8e.*Linux LVM")
        if [[ ! -n ${isLvm} ]]; then
            # part system type is not Linux LVM
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} system type is not Linux LVM, will
skip this device"
            continue
        fi
    fi
    block_devices_size=$(sudo lsblk -n -o SIZE $device 2>/dev/null | awk '{ print $1}')
    if [[ ${block_devices_size}"x" != "${dockerdisksize}Gx" ]] && [[ ${block_devices_size}"x" != "$
{systemdisksize}Gx" ]]; then
echo "n
p
1


w
" | fdisk $device
        mkfs -t ext4 ${device}1
        mkdir -p $mountdir
        echo "${device}1  $mountdir ext4  noatime  0 0" | sudo tee -a /etc/fstab >/dev/null
        mount $mountdir
    fi
done
```

19.7 How Do I Use heapster in Clusters of v1.13.10?

After a cluster of v1.13.10 is created, you can use heapster only after rbac is enabled.

Procedure

Step 1 Connect to the cluster on which you need to use heapster by following the instruction provided in **Connecting to a Cluster Through kubectl**.

Step 2 Delete the existing heapster cluster role.

kubectl delete clusterrole system:heapster

Step 3 Create a heapster cluster role.

Copy the following file to a server on which kubectl is supported, and name the file to **heapster-cluster-role.yaml**.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:heapster
rules:
- apiGroups:
  - ""
  resources:
  - events
  - namespaces
  - nodes
  - pods
  - nodes/stats
  verbs:
  - create
  - get
  - list
  - watch
- apiGroups:
  - extensions
  resources:
  - deployments
  verbs:
  - get
  - list
  - update
  - watch
```

Run the following command to create a heapster cluster role.

kubectl create -f heapster-cluster-role.yaml

Step 4 Create a heapster service account.

Copy the following file to a server on which kubectl is supported, and name the file to **heapster-serviceaccount.yaml**.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: heapster
  namespace: kube-system
```

Run the following command to create a heapster serviceaccount.

kubectl create -f heapster-serviceaccount.yaml

Step 5 Create a heapster cluster role binding.

Copy the following file to a server on which kubectl is supported, and name the file to **heapster-cluster-rolebinding.yaml**.

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: heapster
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:heapster
subjects:
- kind: ServiceAccount
  name: heapster
  namespace: kube-system
```

Run the following command to create a heapster cluster role binding.

kubectl create -f heapster-cluster-rolebinding.yaml

Step 6 Re-create the heapster deployment.

Copy the following file to a server on which kubectl is supported, and name the file to **heapster-apiserver.yaml**.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  generation: 1
  labels:
    k8s-app: heapster
    module: apiserver
    version: v6
  name: heapster-apiserver
  namespace: kube-system
spec:
  progressDeadlineSeconds: 2147483647
  replicas: 1
  revisionHistoryLimit: 2147483647
  selector:
    matchLabels:
      k8s-app: heapster
      module: apiserver
      version: v6
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: heapster
        module: apiserver
        version: v6
      name: heapster
    spec:
      containers:
      - command:
        - /heapster
        - --source=kubernetes.summary_api:''?
useServiceAccount=true&kubeletPort=10250&kubeletHttps=true&insecure=true&auth=/srv/config
        - --api-server
        - --secure-port=6443
        image: k8s.gcr.io/heapster-amd64:v1.5.3
        imagePullPolicy: IfNotPresent
        name: heapster
        ports:
        - containerPort: 6443
          name: https
          protocol: TCP
        - containerPort: 8080
          name: http
          protocol: TCP
        resources: {}
        securityContext:
          runAsUser: 0
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /root/.kube
          name: config
        - mountPath: /srv/config
          name: heapster
          subPath: config
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: heapster
      serviceAccountName: heapster
      terminationGracePeriodSeconds: 30
      volumes:
      - hostPath:
          path: /root/.kube
          type: ""
        name: config
      - configMap:
          defaultMode: 420
          items:
          - key: config
            path: config
          name: heapster
        name: heapster
```

Run the following command to create a heapster deployment.

kubectl delete -f heapster-apiserver.yaml

kubectl create -f heapster-apiserver.yaml

Step 7 Run the following command to check whether heapster is enabled.

kubectl top nodes

heapster is enabled when statistics are displayed in the command output.

----End

19.8 How Do I Change the Mode of the Docker Device Mapper?

Currently, private CCE clusters use Device Mapper as the Docker storage driver.

Device Mapper is developed based on the kernel framework and supports many advanced volume management technologies on Linux.

Docker Device Mapper storage driver leverages the thin provisioning and snapshot capabilities of this framework to manage images and containers.

For CCE clusters of v1.7.3-r6 or earlier, the Docker Device Mapper is set to the loop-lvm mode by default. By default, Docker generates data and metadata files in the **/var/lib/docker/devicemapper/devicemapper** directory. The two files are attached to loop devices and used as block devices. After multiple containers are attached to the files, the performance deteriorates dramatically.

The loop-lvm mode enables you to use Docker out of the box, without additional configuration. This mode is not recommended in the production environment. The Docker Device Mapper also supports the direct-lvm mode. This mode enables you to use raw partitions (no file systems). In the medium-load and high-density environments, this mode provides better performance.

To ensure system stability, you need to set the Docker Device Mapper to the direct-lvm mode.

CCE allows you to change the mode of the Device Mapper on VM nodes running on EulerOS and CentOS.

NOTICE

- Changing the Docker Device Mapper mode on a node requires a data disk. Therefore, in the change process, the system automatically creates a 100 GB SATA disk and binds it to the node.
- When the Docker Device Mapper mode on a node is changed to **direct-lvm**, the container and image data on the node will be deleted. Therefore, you must back up the container and image data of the node to a private image repository or Docker Hub repository before changing the mode.

Procedure

Step 1 Check whether the Docker Device Mapper mode on a node is **direct-lvm**.

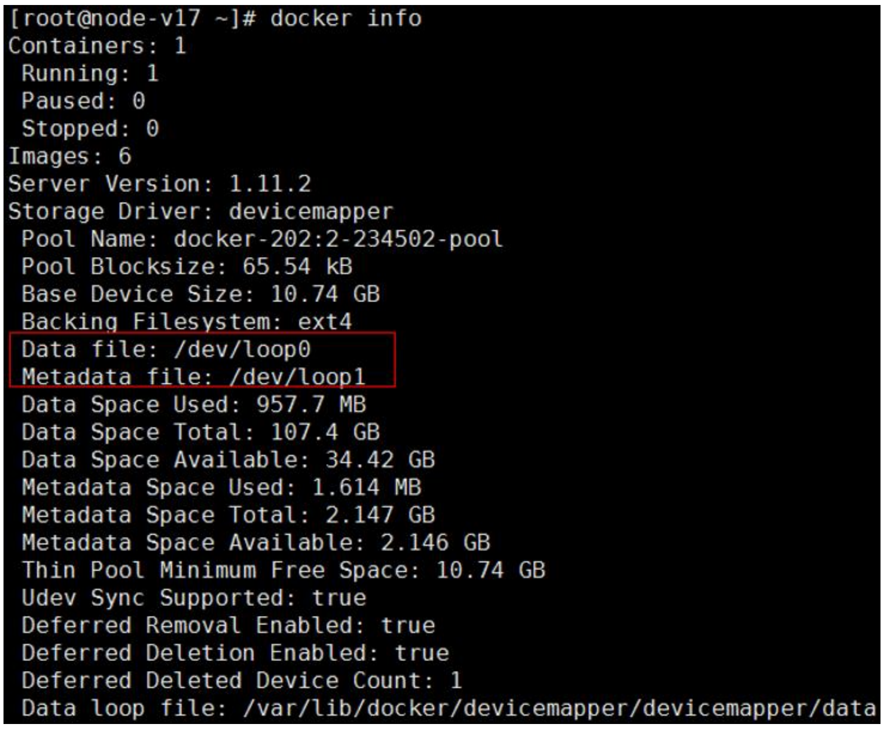
Method 1:

1. Log in to a node on which you want to view the Docker Device Mapper mode.
2. Enter the following command to view the configuration information under **Storage Driver**.

   docker info

   – If the values of the **Data file** and **Metadata file** parameters under **Storage Driver** are **/dev/loopx**, the Docker Device Mapper mode of the current node is **loop-lvm**. Change the mode by following **Step 2**.

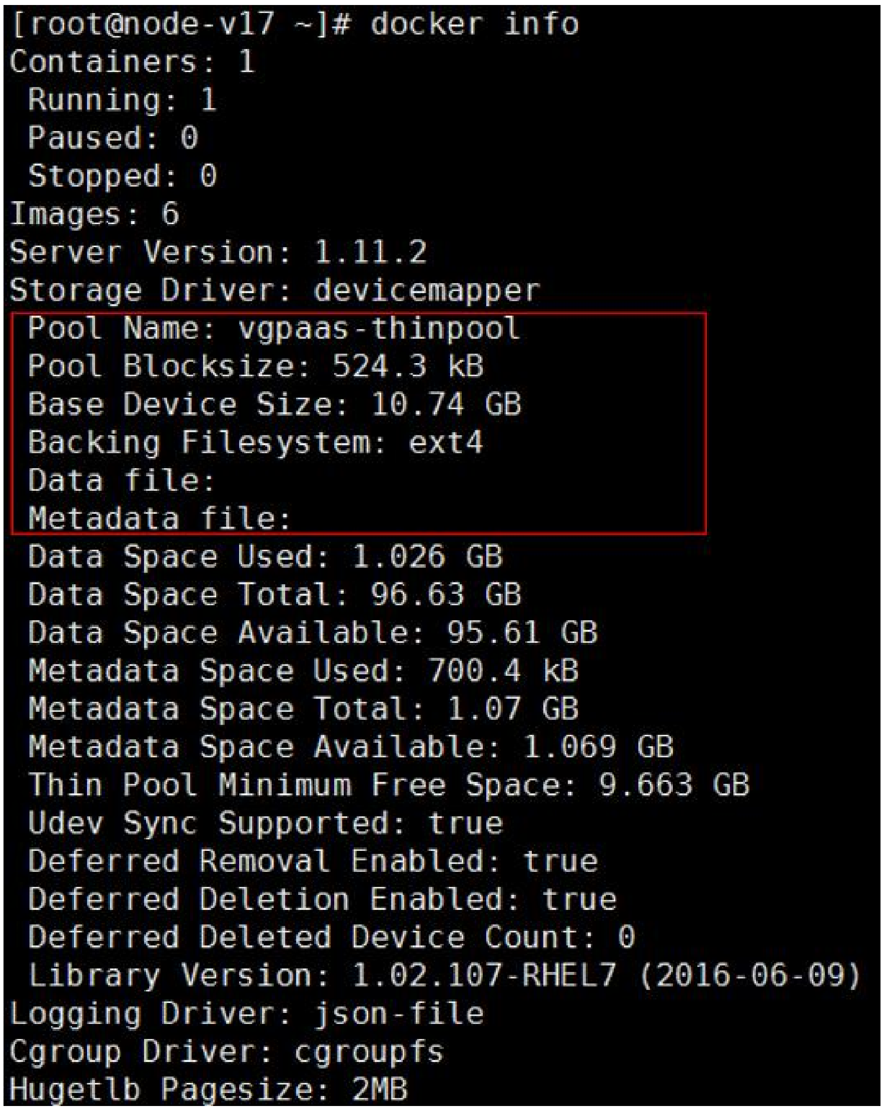
   Example:



```
[root@node-v17 ~]# docker info
Containers: 1
 Running: 1
 Paused: 0
 Stopped: 0
Images: 6
Server Version: 1.11.2
Storage Driver: devicemapper
 Pool Name: docker-202:2-234502-pool
 Pool Blocksize: 65.54 kB
 Base Device Size: 10.74 GB
 Backing Filesystem: ext4
 Data file: /dev/loop0
 Metadata file: /dev/loop1
 Data Space Used: 957.7 MB
 Data Space Total: 107.4 GB
 Data Space Available: 34.42 GB
 Metadata Space Used: 1.614 MB
 Metadata Space Total: 2.147 GB
 Metadata Space Available: 2.146 GB
 Thin Pool Minimum Free Space: 10.74 GB
 Udev Sync Supported: true
 Deferred Removal Enabled: true
 Deferred Deletion Enabled: true
 Deferred Deleted Device Count: 1
 Data loop file: /var/lib/docker/devicemapper/devicemapper/data
```

   – If the values of the **Data file** and **Metadata file** parameters under **Storage Driver** are left blank and the value of **Pool Name** is **vgpaas-thinpool**, the Docker Device Mapper mode of the current node is **direct-lvm**. You do not need to change the mode.

   Example:



```
[root@node-v17 ~]# docker info
Containers: 1
 Running: 1
 Paused: 0
 Stopped: 0
Images: 6
Server Version: 1.11.2
Storage Driver: devicemapper
 Pool Name: vgpaas-thinpool
 Pool Blocksize: 524.3 kB
 Base Device Size: 10.74 GB
 Backing Filesystem: ext4
 Data file:
 Metadata file:
 Data Space Used: 1.026 GB
 Data Space Total: 96.63 GB
 Data Space Available: 95.61 GB
 Metadata Space Used: 700.4 kB
 Metadata Space Total: 1.07 GB
 Metadata Space Available: 1.069 GB
 Thin Pool Minimum Free Space: 9.663 GB
 Udev Sync Supported: true
 Deferred Removal Enabled: true
 Deferred Deletion Enabled: true
 Deferred Deleted Device Count: 0
 Library Version: 1.02.107-RHEL7 (2016-06-09)
Logging Driver: json-file
Cgroup Driver: cgroupfs
Hugetlb Pagesize: 2MB
```

Method 2:

1. Log in to a node on which you want to view the Docker Device Mapper mode.
2. Enter the following command and check whether the command output contains the information listed below:

   cat /etc/docker/daemon.json

   ```
   "dm.thinpooldev=/dev/mapper/vgpaas-thinpool"
   ```

   – If the command output contains the preceding information, the Docker Device Mapper mode of the current node is **direct-lvm**. You do not need to change the mode.

   – If the command output does not contain the preceding information or a message indicating that a file such as **daemon.json** is unavailable is displayed, the Docker Device Mapper mode of the current node is not **direct-lvm**. Change the mode by following **Step 2**.

Step 2 (Optional) If no elastic IP address is bound to the node for which the Docker Device Mapper mode needs to be changed, bind an elastic IP address.

Step 3 Log in to the node with an elastic IP address as the root user.

Step 4 Create a configuration file.

touch config.yaml

Step 5 Copy the following content to the configuration file:

```
user:
  domainName:
  username:
  password:
  projectName:
apiGatewayIp:
iamHostname:
ecsHostname:
evsHostname:
swrAddr:
defaultPassword:
defaultPrivateKey:
hosts:
  - host: <node_ip_01>
    user: root
    password:
    privateKey:
    serverId:
  - host: <node_ip_02>
    user: root
    password:
    privateKey:
    serverId:
```

Table 19-4 Parameter description

Parameter	Description
domainName	Tenant name.
username	User name.
password	User password, which is enclosed in double quotation marks.
projectName	Name of the project to which the node to be configured belongs.
apiGatewayIp	IP address of an API gateway
iamHostname	Endpoint of the IAM service
ecsHostname	Endpoint of the ECS service.
evsHostname	EVS service endpoint.
swrAddr	Address of a software repository.
defaultPassword	(Optional) Default login password of a node. The value must be enclosed in quotation marks (" ").
defaultPrivateKey	(Optional) Absolute path to the default key file for logging in to a node. The value must be enclosed in quotation marks (" ").

Parameter	Description
hosts	Host array structure [1]. You can set multiple nodes for which you want to change the Device Mapper mode. The following parameters must be included: **user**, **password/privateKey**, and **serverId**. For details about the host array structure, see **Table 19-5**.

Table 19-5 Parameter description about the host array structure

Parameter	Description
host	IP address of the node for which you want to change the Device Mapper mode. This node must be in the same subnet as the current logged-in node.
user	User name. Set this parameter to **root**.
password	Password for the **root** user on the node for which you want to change the Device Mapper mode. The value must be enclosed in quotation marks (" "). **NOTE** Set either **password** or **privateKey**.
privateKey	Absolute path to the key file of the **root** user on the node for which you want to change the Device Mapper mode. The value must be enclosed in quotation marks (" "). **NOTE** Set either **password** or **privateKey**.
serverId	ID of the ECS corresponding to the node for which you want to change the Device Mapper mode.

Step 6 Modify the configuration of the nodes in the cluster.

It takes about 3 to 5 minutes to configure a node.

curl -k https://<swr-address>:20202/swr/v2/domains/op_svc_servicestage/namespaces/op_svc_servicestage/repositories/default/packages/cluster-versions/versions/base/file_paths/cceadm -1 -O;chmod u+x cceadm; ./cceadm batch-config-docker --conf=./config.yaml

Replace *<swr-address>* with the address of a software repository, which is the same as the value of **swrAddr** in **Table 19-4**.

----End

19.9 Can I Do If My Cluster Status Is Available but the Node Status Is Unavailable?

If the cluster status is available but some nodes in the cluster are unavailable, perform the following operations to rectify the fault:

Fault Locating

- **Check Item 1: Whether the Node Is Overloaded**
- **Check Item 2: Whether the ECS Is Deleted or Faulty**
- **Check Item 3: Whether You Can Log In to the ECS**
- **Check Item 4: Whether the Security Group Is Modified**
- **Check Item 5: Whether the Disk Is Abnormal**
- **Check Item 6: Whether Internal Components Are Normal**
- **Check Item 7: Whether Related Components Fail to Be Installed on the Node**

Check Item 1: Whether the Node Is Overloaded

Fault locating:

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**.

Step 2 Click the name of an unavailable node to go to the node details page.

Step 3 On the **Monitoring** tab page, click **View Monitoring Details** to go to the AOM console and view historical monitoring records.

A too high CPU or memory usage of the node will result in an excessive network latency or trigger system OOM (see **What Should I Do If the OOM Killer Is Triggered When a Container Uses Memory Resources More Than Limited?**). Therefore, the node is displayed as unavailable.

----End

Solution

1. You are advised to migrate services to reduce the workloads on the node and set the resource upper limit for the workloads.
2. You can also restart the node on the ECS console.

After the node becomes available, the workload is restored.

Check Item 2: Whether the ECS Is Deleted or Faulty

Step 1 Check whether the cluster is available.

Log in to the CCE console, and choose **Resource Management** > **Clusters** in the navigation pane. On the page displayed, check whether the cluster is available.

- If the cluster is unavailable, contact technical support to rectify the fault.
- If the cluster is available but some nodes in the cluster are unavailable, go to **Step 2**.

Step 2 Log in to the ECS console. In the navigation pane, choose **Elastic Cloud Server** to view the ECS status.

- If the ECS status is **Deleted**, go back to the CCE console, choose **Resource Management > Nodes** in the navigation pane, delete the corresponding node, and then create another one.
- If the ECS status is **Stopped** or **Frozen**, restore the ECS first. It takes about 3 minutes to restore the ECS.
- If the ECS status is **Faulty**, restart the ECS. If the ECS is still faulty, contact technical support to rectify the fault.
- If the ECS status is **Running**, log in to the ECS to locate the fault according to **Check Item 6: Whether Internal Components Are Normal**.

----End

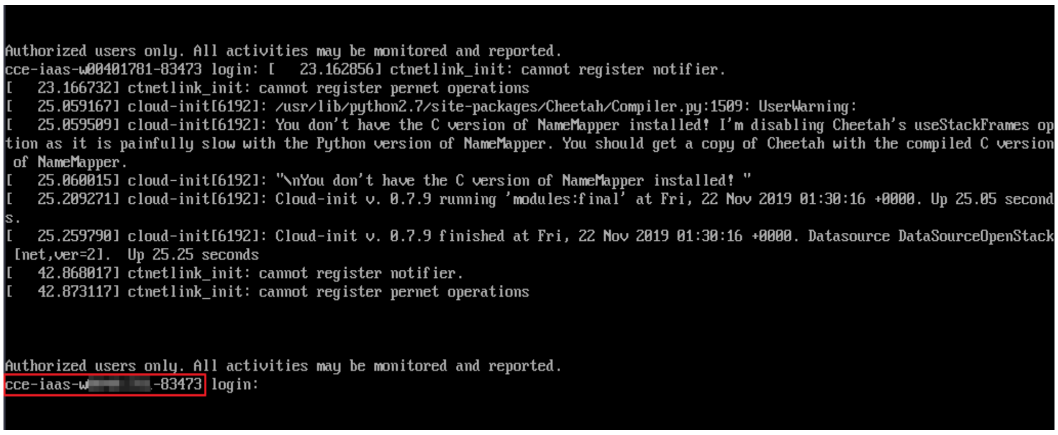
Check Item 3: Whether You Can Log In to the ECS

Step 1 Log in to the management console. Choose **Service List** > **Computing** > **Elastic Cloud Server**.

Step 2 In the ECS list, locate the newly created node (generally named in the format of *cluster name-random number*) in the cluster and click **Remote Login** in the **Operation** column.

Step 3 Check whether the node name displayed on the page is the same as that on the VM and whether the password or key can be used to log in to the node.

Figure 19-2 Checking the node name on the VM and whether the node can be logged in to



```
Authorized users only. All activities may be monitored and reported.
cce-iaas-w00401781-83473 login: [   23.162856] ctnetlink_init: cannot register notifier.
[   23.166732] ctnetlink_init: cannot register pernet operations
[   25.059167] cloud-init[6192]: /usr/lib/python2.7/site-packages/Cheetah/Compiler.py:1509: UserWarning:
[   25.059509] cloud-init[6192]: You don't have the C version of NameMapper installed! I'm disabling Cheetah's useStackFrames op
tion as it is painfully slow with the Python version of NameMapper. You should get a copy of Cheetah with the compiled C version
 of NameMapper.
[   25.060015] cloud-init[6192]: "\nYou don't have the C version of NameMapper installed! "
[   25.209271] cloud-init[6192]: Cloud-init v. 0.7.9 running 'modules:final' at Fri, 22 Nov 2019 01:30:16 +0000. Up 25.05 second
s.
[   25.259790] cloud-init[6192]: Cloud-init v. 0.7.9 finished at Fri, 22 Nov 2019 01:30:16 +0000. Datasource DataSourceOpenStack
 [net,ver=2].  Up 25.25 seconds
[   42.868017] ctnetlink_init: cannot register notifier.
[   42.873117] ctnetlink_init: cannot register pernet operations



Authorized users only. All activities may be monitored and reported.
cce-iaas-w[redacted]-83473 login:
```

If the node names are inconsistent and the password and key cannot be used to log in to the node, Cloud-Init problems occurred when an ECS was created. In this case, restart the node and submit a service ticket to the ECS personnel to locate the root cause.

----End

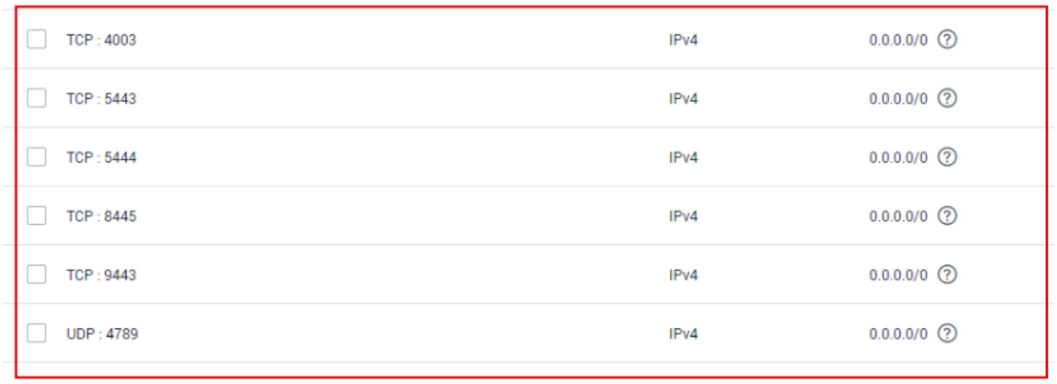
Check Item 4: Whether the Security Group Is Modified

Step 1 Log in to the management console, and choose **Service List** > **Network** > **Virtual Private Cloud**. In the navigation pane, choose **Access Control** > **Security Groups**, and locate the security group of the master node.

The name of this security group is in the format of cluster name-cce-**control**-ID,

Step 2 Click the security group. On the details page displayed, ensure that the security group rules of the master node are the same as those in the following figure.

Figure 19-3 Viewing inbound rules of the security group

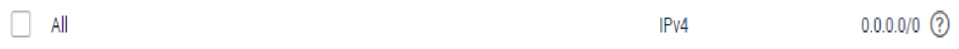


TCP : 4003	IPv4	0.0.0.0/0
TCP : 5443	IPv4	0.0.0.0/0
TCP : 5444	IPv4	0.0.0.0/0
TCP : 8445	IPv4	0.0.0.0/0
TCP : 9443	IPv4	0.0.0.0/0
UDP : 4789	IPv4	0.0.0.0/0

Inbound rule parameter description:

- **4789**: used for network access between containers.
- **5443-5444**: used by kubelet of the node to listen to kube-api of the master node.
- **9443**: used by canal of the node to listen to canal-api of the master node.
- **8445**: used by storage_driver of the node to access csms-storagemgr of the master node.

Figure 19-4 Viewing outbound rules of the security group



All	IPv4	0.0.0.0/0

----End

Check Item 5: Whether the Disk Is Abnormal

After a node is created in a cluster of v1.7.3-r7 or a later version, a 100 GB data disk dedicated for Docker is bound to the node. If the data disk is uninstalled or damaged, the Docker service becomes abnormal and the node becomes unavailable.

Click the node name to check whether the data disk mounted to the node is uninstalled. If the disk is uninstalled, mount a data disk to the node again and restart the node. Then the node can be recovered.

Check Item 6: Whether Internal Components Are Normal

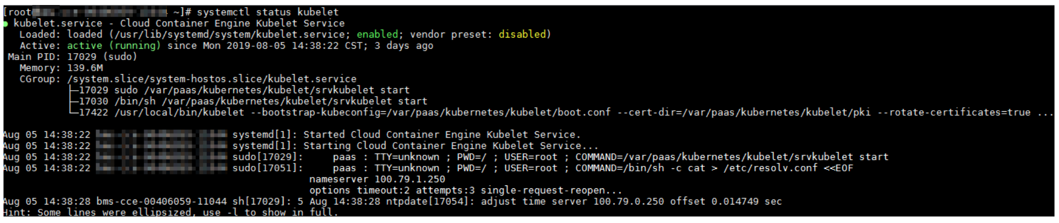
Step 1 Log in to the ECS where the unavailable node is located.

Step 2 Run the following command to check whether the PaaS components are normal:

For version 1.13, run the following command:

systemctl status kubelet

If this command fails to be run, contact technical support. If this command is successfully executed, the status of each component is displayed as **active**, as shown in the following figure.



```
[root[redacted] ~]# systemctl status kubelet
● kubelet.service - Cloud Container Engine Kubelet Service
   Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-08-05 14:38:22 CST; 3 days ago
 Main PID: 17029 (sudo)
   Memory: 139.6M
   CGroup: /system.slice/system-hostos.slice/kubelet.service
           ├─17029 sudo /var/paas/kubernetes/kubelet/srvkubelet start
           ├─17030 /bin/sh /var/paas/kubernetes/kubelet/srvkubelet start
           └─17422 /usr/local/bin/kubelet --bootstrap-kubeconfig=/var/paas/kubernetes/kubelet/boot.conf --cert-dir=/var/paas/kubernetes/kubelet/pki --rotate-certificates=true ...

Aug 05 14:38:22 [redacted] systemd[1]: Started Cloud Container Engine Kubelet Service.
Aug 05 14:38:22 [redacted] systemd[1]: Starting Cloud Container Engine Kubelet Service...
Aug 05 14:38:22 [redacted] sudo[17029]:     paas : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/var/paas/kubernetes/kubelet/srvkubelet start
Aug 05 14:38:22 [redacted] sudo[17051]:     paas : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/bin/sh -c cat > /etc/resolv.conf <<EOF
                                                                nameserver 100.79.1.250
                                                                options timeout:2 attempts:3 single-request-reopen...
Aug 05 14:38:28 bms-cce-00406059-11044 sh[17029]: 5 Aug 14:38:28 ntpdate[17054]: adjust time server 100.79.0.250 offset 0.014749 sec
Hint: Some lines were ellipsized, use -l to show in full.
```

If the component status is not **active**, run the following commands (using the faulty component **canal** as an example):

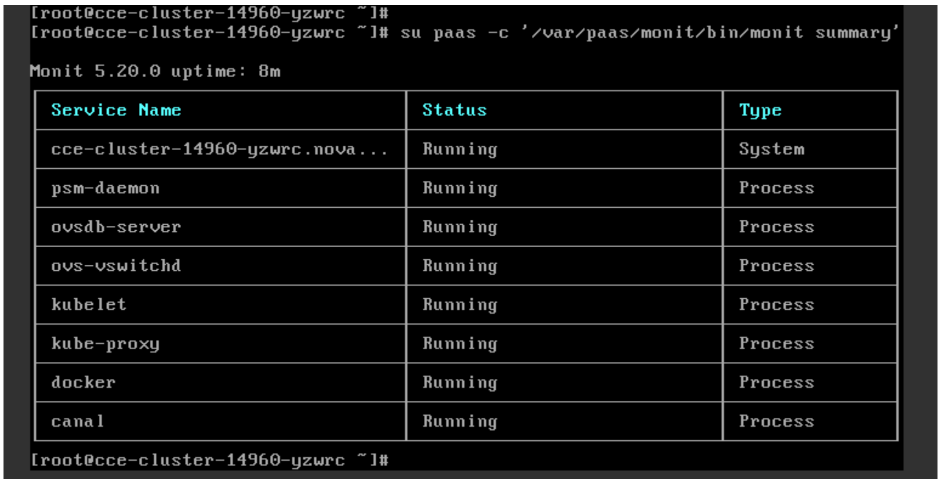
Run **systemctl restart canal** to restart the component.

After restarting the component, run **systemctl status canal** to check the status.

For versions earlier than v1.13, run the following command:

su paas -c '/var/paas/monit/bin/monit summary'

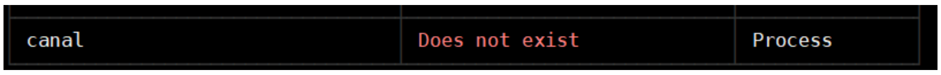
If this command fails to be run, contact technical support. If this command is successfully executed, the status of each component is displayed, as shown in the following figure.



```
[root@cce-cluster-14960-yzwrc ~]#
[root@cce-cluster-14960-yzwrc ~]# su paas -c '/var/paas/monit/bin/monit summary'

Monit 5.20.0 uptime: 8m
┌────────────────────────────────┬───────────────────────────┬──────────────┐
│ Service Name                   │ Status                    │ Type         │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ cce-cluster-14960-yzwrc.nova...│ Running                   │ System       │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ psm-daemon                     │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ ovsdb-server                   │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ ovs-vswitchd                   │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ kubelet                        │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ kube-proxy                     │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ docker                         │ Running                   │ Process      │
├────────────────────────────────┼───────────────────────────┼──────────────┤
│ canal                          │ Running                   │ Process      │
└────────────────────────────────┴───────────────────────────┴──────────────┘
[root@cce-cluster-14960-yzwrc ~]#
```

If any service component is not in the **Running** state, restart the corresponding service. For example, the canal component is abnormal, as shown in the following figure.

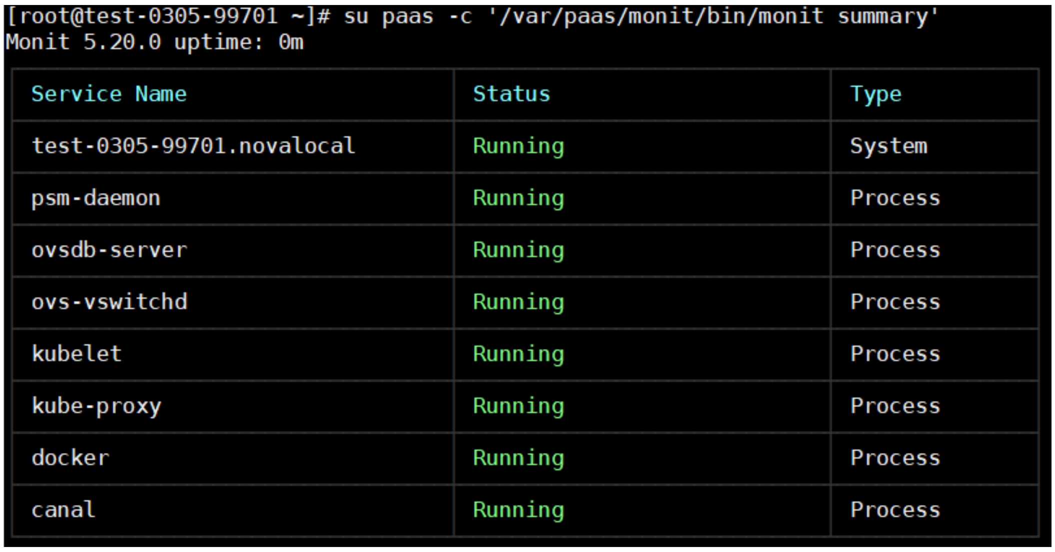


```
│ canal                          │ Does not exist            │ Process      │
```

Run **su paas -c '/var/paas/monit/bin/monit restart canal'** to restart the canal component.

After the restart, run **su paas -c '/var/paas/monit/bin/monit summary'** to query the status of the canal component.

In that case, the status of each component is **Running**, as shown in the following figure.



```
[root@test-0305-99701 ~]# su paas -c '/var/paas/monit/bin/monit summary'
Monit 5.20.0 uptime: 0m
```

Service Name	Status	Type
test-0305-99701.novalocal	Running	System
psm-daemon	Running	Process
ovsdb-server	Running	Process
ovs-vswitchd	Running	Process
kubelet	Running	Process
kube-proxy	Running	Process
docker	Running	Process
canal	Running	Process

Step 3 If the restart command fails to be run, run the following command to check the running status of the monitrc process:

ps -ef | grep monitrc

- If the monitrc process exists, run the following command to kill this process. The monitrc process will be automatically restarted after it is killed.

  kill -s 9 `ps -ef | grep monitrc | grep -v grep | awk '{print $2}'`

- If the monitrc process does not exist or is not restarted after being killed, contact technical support.

Step 4 If the fault persists, collect logs in the **/var/log/message** and **/var/paas/sys/log** directories, and contact technical support.

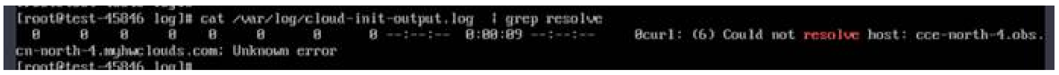
----End

Check Item 7: Whether Related Components Fail to Be Installed on the Node

Step 1 Log in to the management console. Choose **Service List** > **Computing** > **Elastic Cloud Server**.

Step 2 In the ECS list, locate the newly created node (generally named in the format of *cluster name-random number*) in the cluster and click **Remote Login** in the **Operation** column.

Step 3 After logging in to the node, check the **/var/log/cloud-init-output.log** file. As shown in the following figure, domain name resolution failure is recorded.



```
[root@test-45846 log]# cat /var/log/cloud-init-output.log  | grep resolve
  0     0    0     0    0     0     0      0 --:--:--  0:00:09 --:--:--     0curl: (6) Could not resolve host: cce-north-4.obs.
cn-north-4.myhwclouds.com: Unknown error
[root@test-45846 log]#
```

Step 4 Check whether the domain name can be resolved on the node.

----End

Solution

- If the domain name cannot be resolved, check whether the DNS address in the **/etc/resolv.conf** file is the same as that configured on the VPC subnet. In most cases, the DNS address in the file is incorrectly configured. As a result, the domain name cannot be resolved.

  Change the DNS address on the VPC console and reset the node to rectify the fault.
- If the domain name can be resolved, run **bash /home/paas/node/install_node.sh** to manually trigger the installation. Generally, the installation is complete within 3 minutes.

19.10 How Do I Rectify the Fault When the Cluster Status Is Unavailable?

If the cluster is **Unavailable**, perform the following operations to rectify the fault:

Fault Locating

- Check Item 1: Whether the Security Group Is Modified
- Check Item 2: Whether the DHCP Function of the Subnet Is Disabled

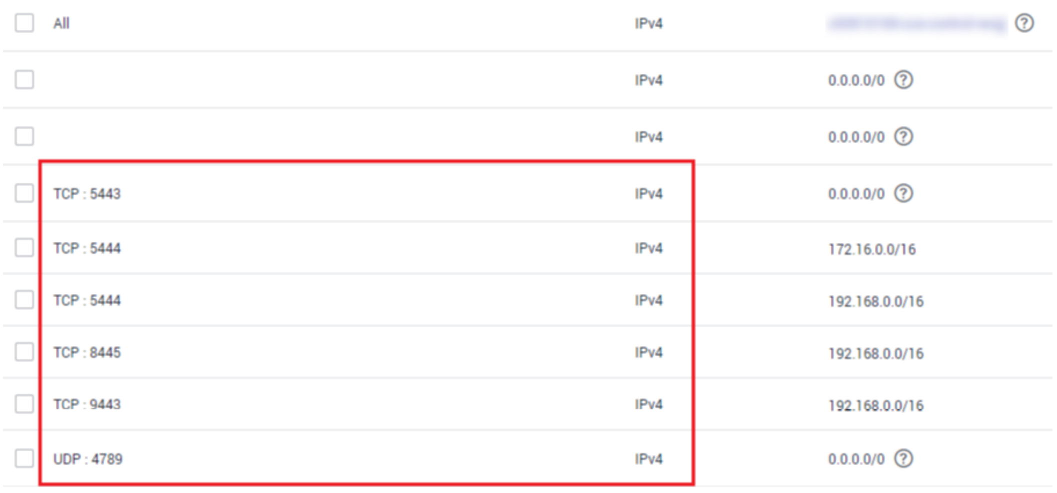
Check Item 1: Whether the Security Group Is Modified

Step 1 Log in to the management console, and choose **Service List** > **Network** > **Virtual Private Cloud**. In the navigation pane on the left, choose **Access Control** > **Security Groups** to find the security group of the master node in the cluster.

The name of this security group is in the format of cluster name -cce-**control**-ID.

Step 2 Click the security group. On the details page displayed, ensure that the security group rules of the master node are the same as those in the following figure.

Figure 19-5 Viewing inbound rules of the security group



All	IPv4	[redacted]
	IPv4	0.0.0.0/0
	IPv4	0.0.0.0/0
TCP : 5443	IPv4	0.0.0.0/0
TCP : 5444	IPv4	172.16.0.0/16
TCP : 5444	IPv4	192.168.0.0/16
TCP : 8445	IPv4	192.168.0.0/16
TCP : 9443	IPv4	192.168.0.0/16
UDP : 4789	IPv4	0.0.0.0/0

Inbound rule parameter description:

- **4789**: used for network access between containers.
- **5443-5444**: used by kubelet of the node to listen to kube-api of the master node.
- **9443**: used by canal of the node to listen to canal-api of the master node.
- **8445**: used by storage_driver of the node to access csms-storagemgr of the master node.

Figure 19-6 Viewing outbound rules of the security group

All	IPv4	0.0.0.0/0

----End

Check Item 2: Whether the DHCP Function of the Subnet Is Disabled

Step 1 On the CCE console, click the unavailable cluster to access the details page. View the VPC and subnet where the cluster is located. Assume that the VPC and subnet names are **vpc-test** and **subnet-test** respectively.

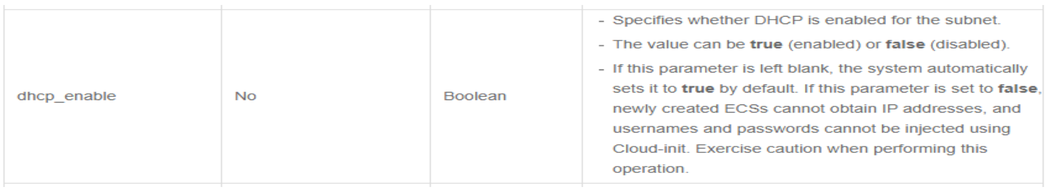
Step 2 On the management console, choose **Service List** > **Network** > **Virtual Private Cloud**. On the VPC details page displayed, check whether the DHCP function of the corresponding subnet is enabled.

If the DHCP function is disabled, the IP address allocated by the subnet will be reclaimed. As a result, the network in the cluster is disconnected and the cluster is abnormal.

In this case, enable the DHCP function again. After this function is enabled, the subnet allocates the original IP address to the original node and the cluster can automatically recover.

Currently, the VPC console no longer supports enabling or disabling DHCP for a VPC. You can change the configuration using APIs. For details, see the description of the **dhcp_enable** field in **Updating Subnet Information**.

Figure 19-7 DHCP description in the VPC API Reference



dhcp_enable	No	Boolean	- Specifies whether DHCP is enabled for the subnet. - The value can be **true** (enabled) or **false** (disabled). - If this parameter is left blank, the system automatically sets it to **true** by default. If this parameter is set to **false**, newly created ECSs cannot obtain IP addresses, and usernames and passwords cannot be injected using Cloud-init. Exercise caution when performing this operation.

----End

19.11 How Do I Use kubectl to Set the Workload Access Type to LoadBalancer (ELB)?

This section uses the Nginx workload as an example to describe how to set the workload access type to **LoadBalancer (ELB)**.

Prerequisite

- An ELB has been created.
- You have connected an Elastic Cloud Server (ECS) to the cluster by running the kubectl command. For details, see **Connecting to a CCE Cluster Using kubectl**.

Procedure

Step 1 Log in to the ECS on which kubectl has been configured.

Step 2 Create and edit the **nginx-deployment.yaml** and **nginx-elb-svc.yaml** files.

The file names are user-defined. **nginx-deployment.yaml** and **nginx-elb-svc.yaml** are merely example file names.

vi nginx-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

vi nginx-elb-svc.yaml

NOTE

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

- The workload protocol is TCP.
- Anti-affinity has been configured between pods of the workload. That is, all pods of the workload are deployed on different nodes.

- Automatically creating a load balancer

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/session-affinity-mode: SOURCE_IP
    kubernetes.io/elb.subnet-id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0
    kubernetes.io/elb.enterpriseID: debb7ae2-6d2f-4e6c-a0aa-1ccafd92b8eb
    kubernetes.io/elb.autocreate: '{"type":"public","bandwidth_name":"cce-
bandwidth-1551163379627","bandwidth_chargemode":"traffic","bandwidth_size":
5,"bandwidth_sharetype":"PER","eip_type":"5_bgp","name":"james"}'
  labels:
    app: nginx
  name: nginx
```

```
spec:
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Using an existing load balancer

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/session-affinity-mode: SOURCE_IP
    kubernetes.io/elb.id: 3c7caa5a-a641-4bff-801a-feace27424b6
    kubernetes.io/elb.subnet-id: 5083f225-9bf8-48fa-9c8b-67bd9693c4c0
  labels:
    app: nginx
  name: nginx
spec:
  loadBalancerIP: 10.78.42.242
  externalTrafficPolicy: Local
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Table 19-6 Key parameters

Parameter	Category	Description
kubernetes.io/elb.class	String	Mandatory and must be set to **union** if an ELB is in use.
kubernetes.io/session-affinity-mode	String	Optional. If sticky session is enabled, add this parameter. The value **SOURCE_IP** indicates that listeners ensure sticky session based on source IP addresses.
kubernetes.io/elb.id	String	Optional. This parameter is mandatory if an existing load balancer is used. ID of an ELB.
kubernetes.io/elb.subnet-id	String	Optional. This parameter is mandatory only if a load balancer will be automatically created. For clusters of v1.11.7-r0 or later, this parameter can be left blank.

Parameter	Category	Description
kubernetes.io/elb.enterpriseID	String	Optional. This parameter is mandatory if a public/private network load balancer will be automatically created. This parameter indicates the name of the ELB enterprise project in which the load balancer will be created.
kubernetes.io/elb.autocreate	**elb.autocreate** object	Optional. This parameter is mandatory if a public network load balancer will be automatically created. The system will create a load balancer and an EIP. This parameter is also mandatory if a private network load balancer will be automatically created. The system will create a load balancer. **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":"traffic","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"}
loadBalancerIP	String	Private IP address of a private network load balancer or public IP address of a public network load balancer.
externalTrafficPolicy	String	Optional. If sticky session is enabled, add this parameter so requests are transferred to a fixed node. If a LoadBalancer Service with this parameter set to **Local** is created, a client can access the target backend only if the client is installed on the same node as the backend.
port	Integer	Access port that is registered on the load balancer and mapped to the cluster-internal IP address.
targetPort	String	Container port on the CCE console.

Table 19-7 Data structure of the elb.autocreate field

Parameter	Category	Description
name	String	Name of the automatically created load balancer. The value is a string of 1 to 64 characters that consist of letters, digits, underscores (_), and hyphens (-).
type	String	Network type of the load balancer. ● **public**: public network load balancer ● **inner**: private network load balancer
bandwidth_name	String	Bandwidth name. The default value is **cce-bandwidth-******. The value is a string of 1 to 64 characters that consist of letters, digits, underscores (_), hyphens (-), and periods (.).
bandwidth_chargemode	String	Bandwidth billing mode. The value is **traffic**, indicating that the billing is based on traffic.
bandwidth_size	Integer	Bandwidth size. Set this parameter based on the bandwidth range supported by the region.
bandwidth_sharetype	String	Bandwidth sharing mode. ● **PER**: dedicated bandwidth ● **WHOLE**: shared bandwidth
eip_type	String	EIP type. For details, see the EIP types supported by ELB.

Step 3 Create a workload.

kubectl create -f nginx-deployment.yaml

If information similar to the following is displayed, the workload is being created:

```
deployment "nginx" created
```

kubectl get po

If information similar to the following is displayed, the workload is running.

```
NAME                    READY     STATUS             RESTARTS   AGE
etcd-0                  0/1       ImagePullBackOff   0          1h
icagent-m9dkt           0/0       Running            0          3d
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 is being created:

```
service "nginx" created
```

kubectl get svc

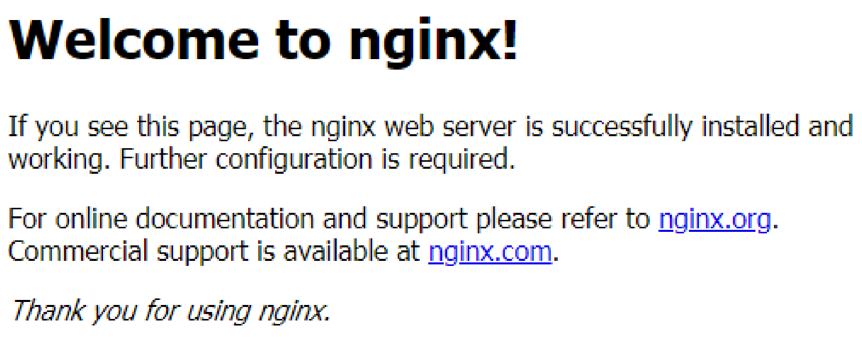
If information similar to the following is displayed, the access type has been set successfully, and the workload is accessible.

```
NAME        TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
etcd-svc    ClusterIP    None            <none>        3120/TCP       1h
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 In the address box of your browser, enter **10.78.42.242:31540** and press **Enter**. **10.78.42.242** indicates the IP address of the load balancer, and **31540** indicates the access port displayed on the CCE console.

Nginx is accessible.

Figure 19-8 Accessing Nginx through a load balancer



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

19.12 How Do I Select a Network Model When Creating a CCE Cluster? What Are the Differences Between These Models?

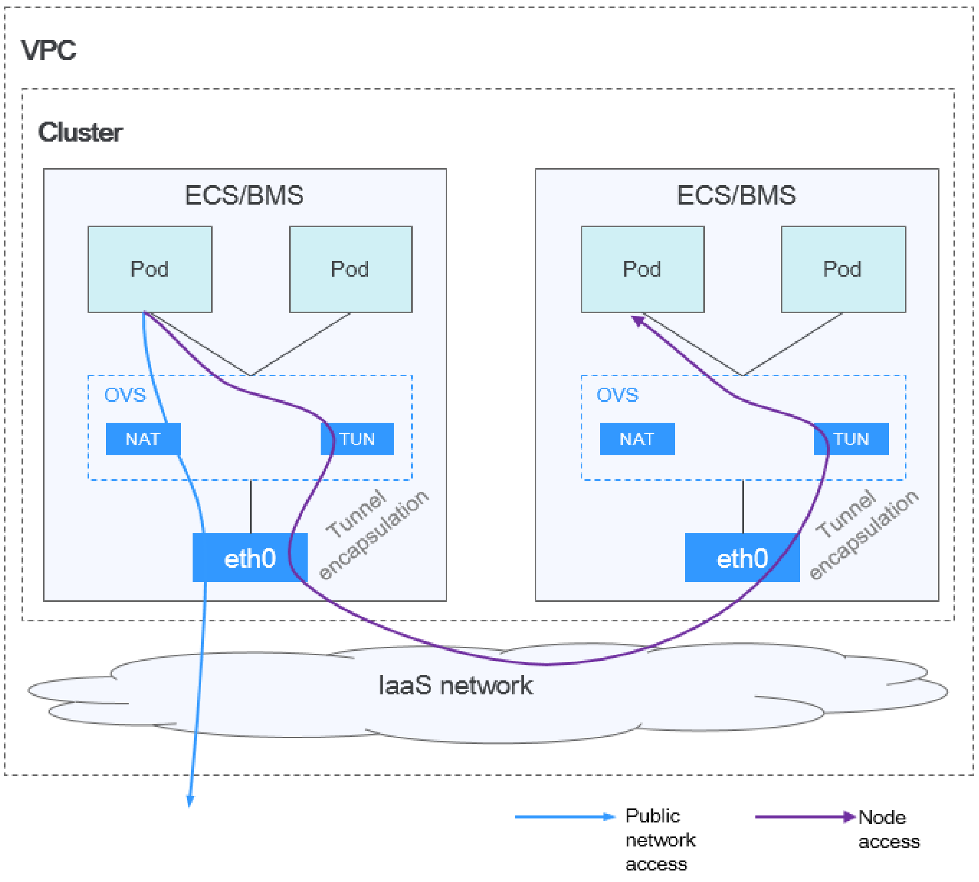
CCE uses high-performance container network add-ons, which support the tunnel network and the VPC network models.

CAUTION

After a cluster is created, the network model cannot be changed. Exercise caution when selecting a network model.

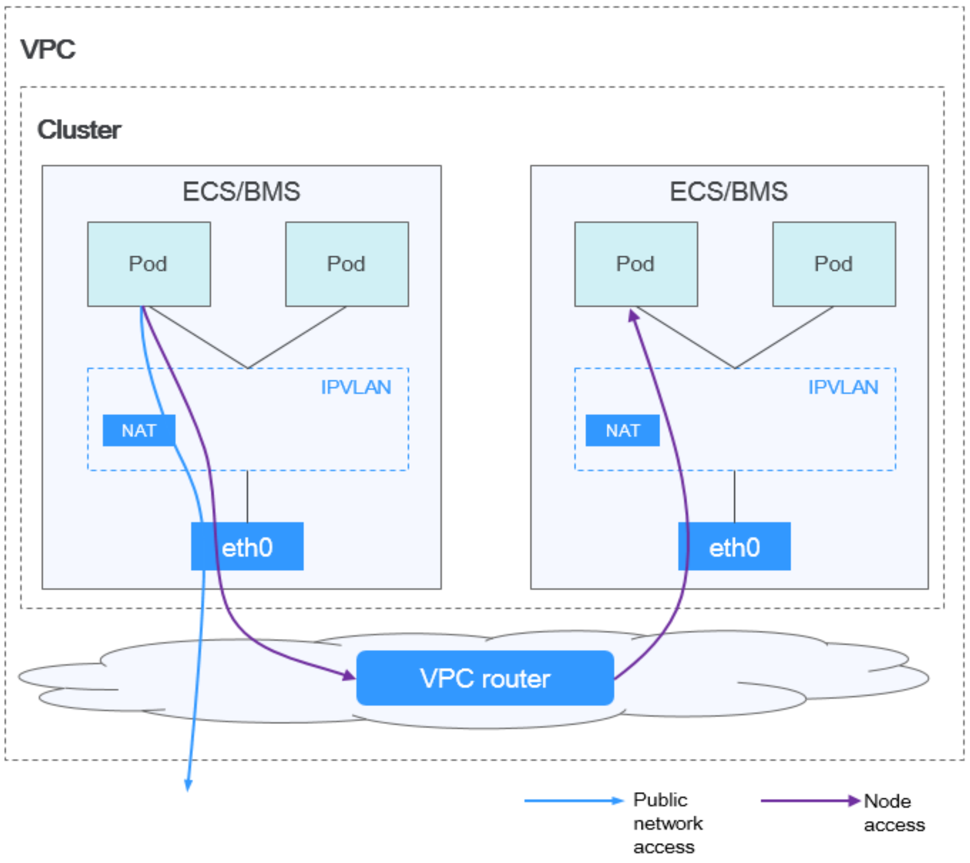
- **Tunnel network**: The container network is an overlay tunnel network on top of a VPC network and uses the VXLAN technology. This network model is applicable when there is no high requirements on performance. VXLAN encapsulates Ethernet packets as UDP packets for tunnel transmission. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.

Figure 19-9 Container tunnel network



- **VPC network**: The container network uses VPC routing to integrate with the underlying network. This network model is applicable to performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. VPC networks are free from tunnel encapsulation overhead and outperform container tunnel networks. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in the cluster can be directly accessed from outside the cluster.

Figure 19-10 VPC network



The following table lists the differences between the network models.

Table 19-8 Network comparison

Dimension	Tunnel Network	VPC Network
Core component	OVS	IPVlan
Applicable clusters	Hybrid cluster VM cluster	Hybrid cluster VM cluster
Support for network policies (networkpolicy)	Yes	No
Support for ENI	No	Yes. The container network is deeply integrated with the VPC network, and ENI is used for pods to communicate.

Dimension	Tunnel Network	VPC Network
IP address management	IP addresses can be migrated.	● Each node is allocated with a small subnet. ● A static route is added on the VPC router with the next hop set to the node IP address.
Network performance	Performance loss due to VXLAN tunnel encapsulation	● No performance loss as no tunnel encapsulation is required; performance comparable to bare metal networks ● Data forwarded across nodes through the VPC router
Networking scale	1,000 nodes	Limited by the VPC route table.
External Dependency	None	Static route table of the VPC router
Application Scenarios	● Common container service scenarios ● Scenarios that do not have high requirements on network latency and bandwidth	● Scenarios that have high requirements on network latency and bandwidth ● Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.

NOTICE

1. The actual cluster scale is limited by the quota of custom routes of the VPC. Therefore, estimate the number of required nodes before creating a VPC.
2. By default, the VPC network model supports direct communication between containers and hosts in the same VPC. If a peering connection policy is configured between the VPC and another VPC, the containers can directly communicate with hosts on the peer VPC. In addition, in hybrid networking scenarios such as cloud private line and VPN, communication between containers and hosts on the peer end can also be achieved with proper planning.

19.13 Which CIDR Blocks Does CCE Support?

Before creating a cluster on CCE, determine the number of VPCs, number of subnets, container CIDR blocks, and Services for access based on service requirements.

This topic describes the functions of various addresses in the CCE cluster in the VPC environment and how to plan CIDR blocks.

Basic Concepts

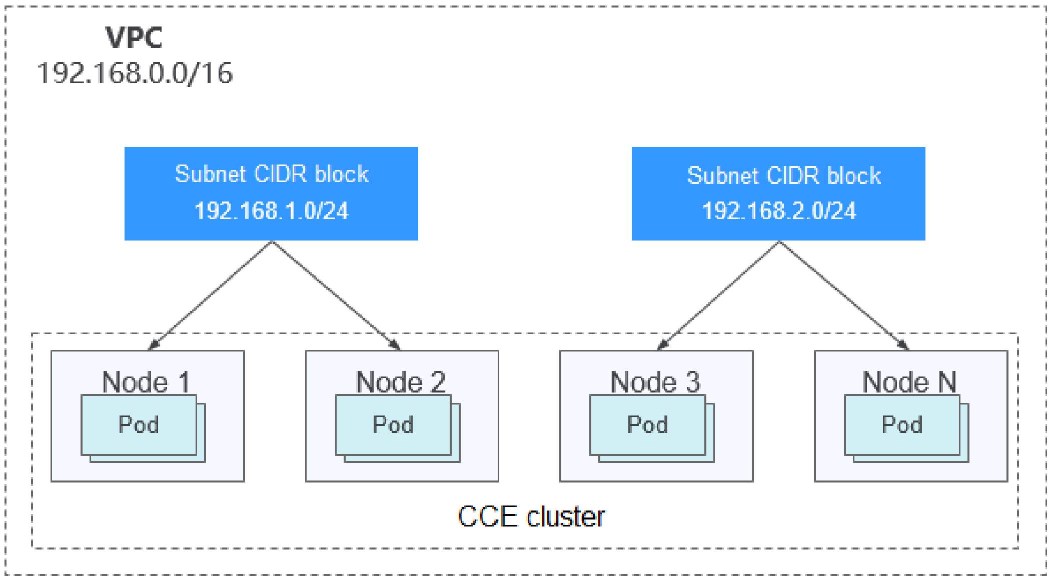
VPC CIDR Block

Virtual Private Cloud (VPC) enables you to provision logically isolated, configurable, and manageable virtual networks for cloud servers, cloud containers, and cloud databases. You have complete control over your virtual network, including selecting your own CIDR block, creating subnets, and configuring security groups. You can also assign EIPs and allocate bandwidth in your VPC for secure and easy access to your business system.

Subnet CIDR Block

A subnet is a network that manages ECS network planes. It supports IP address management and DNS. The IP addresses of all ECSs in a subnet belong to the subnet.

Figure 19-11 VPC CIDR block architecture



By default, ECSs in all subnets of the same VPC can communicate with one another, while ECSs in different VPCs cannot communicate with one another.

You can create a VPC peering connection to enable ECSs in different VPCs to communicate with each other.

Container (Pod) CIDR Block

Pod is a Kubernetes concept. Each pod has an IP address.

When creating a cluster on CCE, you can specify the pod (container) CIDR block, which cannot overlap with the subnet CIDR block. For example, if the subnet CIDR block is 192.168.0.0/16, the container CIDR block cannot be 192.168.0.0/18 or 192.168.1.0/18, because these addresses are included in 192.168.0.0/16.

Service CIDR Block

Service is also a Kubernetes concept. Each Service has an address. When creating a cluster on CCE, you can specify the Service CIDR block. Similarly, the Service CIDR block cannot overlap with the subnet CIDR block or the container CIDR block. The Service CIDR block can be used only within a cluster.

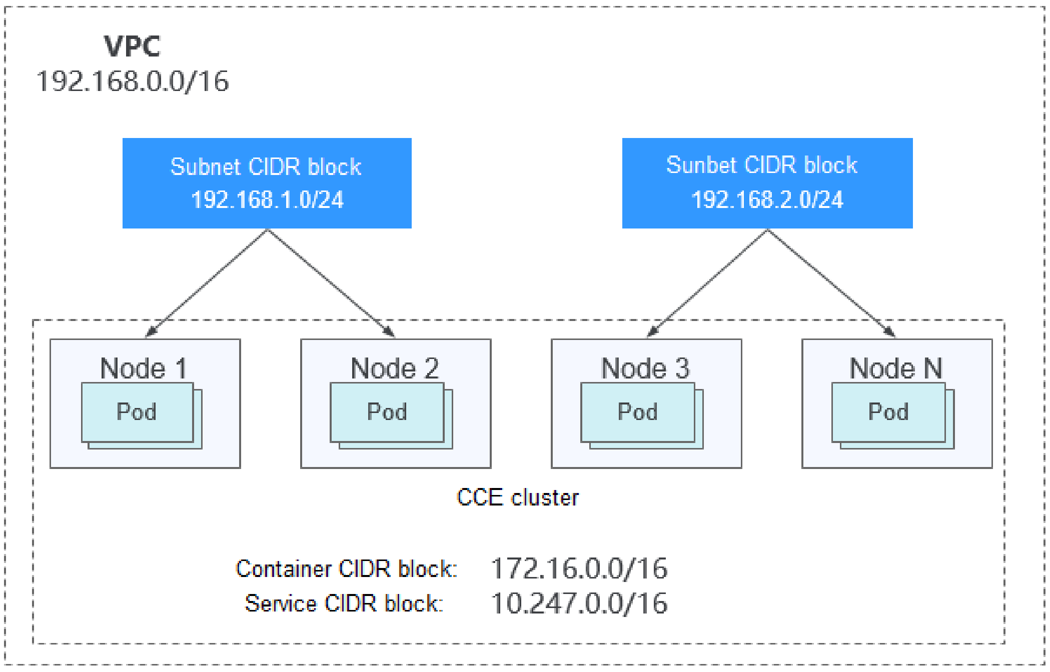
For details about the relationship between these CIDR blocks, see **Figure 19-12**.

How Do I Select a CIDR Block?

Single-VPC Single-Cluster Scenarios

These are the simplest scenarios. The VPC CIDR block is determined when the VPC is created. When creating a CCE cluster, select a CIDR block different from that of the current VPC.

Figure 19-12 CIDR block in the single-VPC single-cluster scenario



Single-VPC Multi-Cluster Scenarios

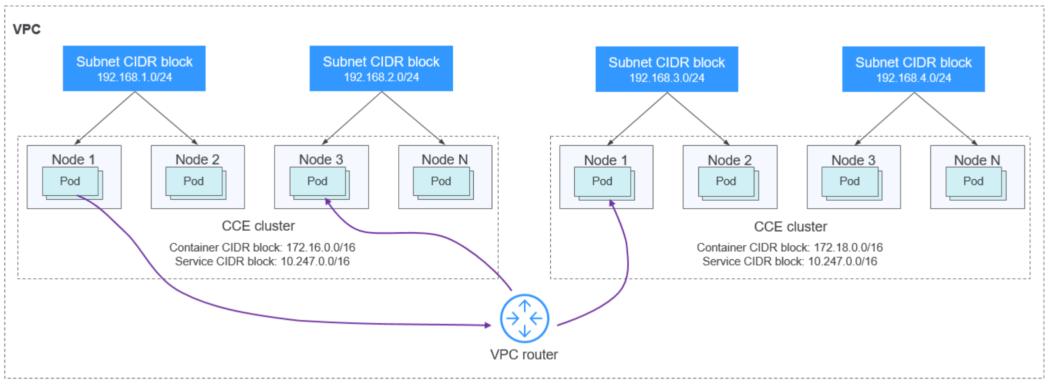
Multiple CCE clusters are created in a VPC.

In the **VPC network** mode, pod packets are forwarded through VPC routes. CCE automatically configures a routing table on the VPC routes to each container CIDR block.

Pay attention to the following:

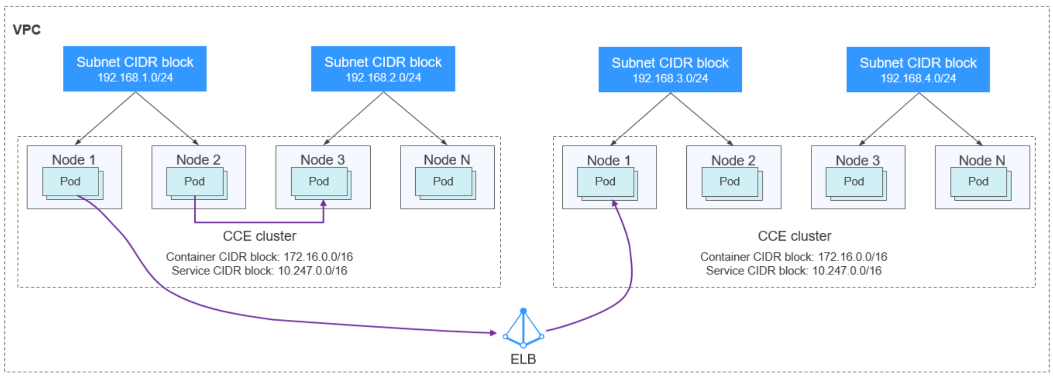
- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can. In this case, CCE clusters are partially interconnected. A pod of a cluster can directly access the pods of another cluster, but cannot access the Services of the cluster.
- The network scale is limited by the VPC route table.

Figure 19-13 VPC network - multi-cluster scenario


VPC
Subnet CIDR block 192.168.1.0/24
Subnet CIDR block 192.168.2.0/24
Subnet CIDR block 192.168.3.0/24
Subnet CIDR block 192.168.4.0/24
Node 1
Node 2
Node 3
Node N
Pod
CCE cluster
Container CIDR block: 172.16.0.0/16
Service CIDR block: 10.247.0.0/16
CCE cluster
Container CIDR block: 172.18.0.0/16
Service CIDR block: 10.247.0.0/16
VPC router

In the tunnel network model, the container network is an overlay network plane deployed over the VPC network. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.

Figure 19-14 Tunnel network - multi-cluster scenario


VPC
Subnet CIDR block 192.168.1.0/24
Subnet CIDR block 192.168.2.0/24
Subnet CIDR block 192.168.3.0/24
Subnet CIDR block 192.168.4.0/24
Node 1
Node 2
Node 3
Node N
Pod
CCE cluster
Container CIDR block: 172.16.0.0/16
Service CIDR block: 10.247.0.0/16
CCE cluster
Container CIDR block: 172.16.0.0/16
Service CIDR block: 10.247.0.0/16
ELB

Pay attention to the following:

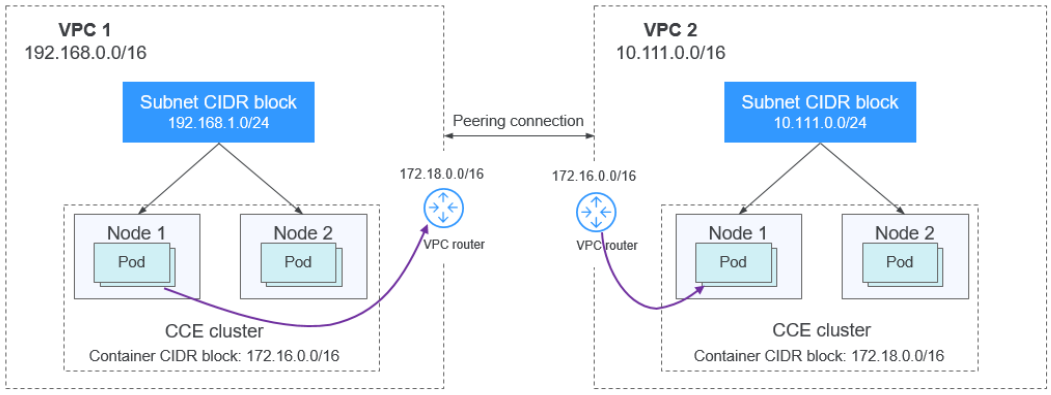
- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters can overlap, so do the Service CIDR blocks.
- It is recommended that ELB be used for the cross-cluster access between containers.

VPC Interconnection Scenarios

When two VPC networks are interconnected, you can configure the packets to be sent to the peer VPC in the route table.

In the VPC network model, after creating a peering connection, you need to add routes for the peering connection to enable communication between the two VPCs.

Figure 19-15 VPC Network - VPC interconnection scenario



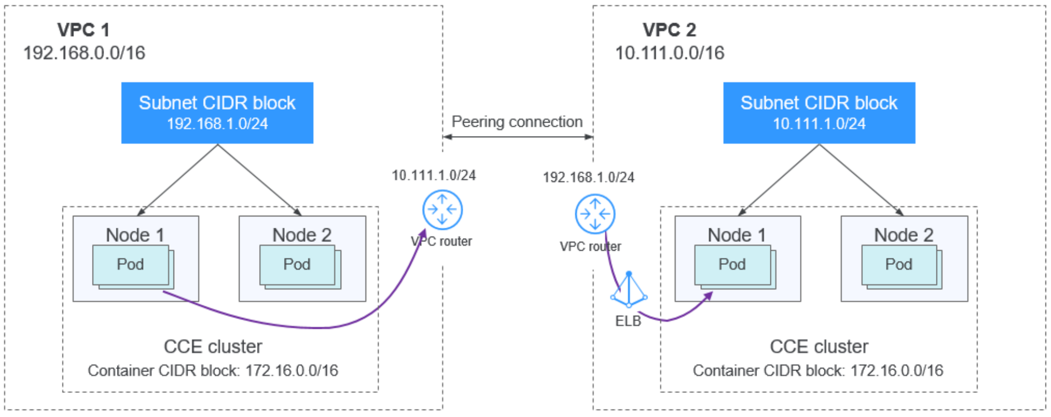
To interconnect cluster containers across VPCs, you need to create VPC peering connections.

Pay attention to the following:

- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can.
- Add the peer container CIDR block to the route table of the VPC peering connection.

In the tunnel network model, after creating a peering connection, you need to add routes for the peering connection to enable communication between the two VPCs.

Figure 19-16 Tunnel network - VPC interconnection scenario



Pay attention to the following:

- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can.
- Add the peer subnet CIDR block to the route table of the VPC peering connection.

VPC-IDC Scenarios

Similar to the VPC interconnection scenario, some CIDR blocks in the VPC are routed to the IDC. The pod IP addresses of CCE clusters cannot overlap with the addresses within these CIDR blocks. To access the pod IP addresses in the cluster in the IDC, you need to configure the route table to the private line VBR on the IDC.

19.14 How Do I Add a Second Data Disk to a Node in a CCE Cluster?

You can use the **pre-installation script** feature to configure CCE cluster nodes (ECSs).

Before using this feature, write a script that can format data disks and save it to your OBS bucket. Then, inject a command line that will automatically execute the disk formatting script when the node is up. The script specifies the size of each docker data disk (for example, the default docker disk is 100 GB and the additional disk is 110 GB) and the mount path (**/data/code**) of the additional disk. In this example, the script is named **formatdisk.sh**. **Note that the script must be run by the root user.**

Example command line:

```
cd /tmp;curl -k -X GET OBS bucket address /formatdisk.sh -1 -O;fdisk -l;sleep 30;bash -x formatdisk.sh
100 /data/code;fdisk -l
```

Example script (**formatdisk.sh**):

```
dockerdisksize=$1
mountdir=$2
systemdisksize=40
i=0
while [ 20 -gt $i ]; do
    echo $i;
    if [ $(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}' |wc -l) -
ge 3 ]; then
        break
    else
        sleep 5
    fi;
    i=$[i+1]
done
all_devices=$(lsblk -o KNAME,TYPE | grep disk | grep -v nvme | awk '{print $1}' | awk '{ print "/dev/"$1}')
for device in ${all_devices[@]}; do
    isRawDisk=$(lsblk -n $device 2>/dev/null | grep disk | wc -l)
    if [[ ${isRawDisk} > 0 ]]; then
        # is it partitioned ?
        match=$(lsblk -n $device 2>/dev/null | grep -v disk | wc -l)
        if [[ ${match} > 0 ]]; then
            # already partited
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Raw disk ${device} has been partition, will skip
this device"
            continue
        fi
    else
        isPart=$(lsblk -n $device 2>/dev/null | grep part | wc -l)
        if [[ ${isPart} -ne 1 ]]; then
            # not parted
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has not been partition, will skip this
device"
            continue
        fi
        # is used ?
        match=$(lsblk -n $device 2>/dev/null | grep -v part | wc -l)
        if [[ ${match} > 0 ]]; then
            # already used
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
            continue
        fi
        isMount=$(lsblk -n -o MOUNTPOINT $device 2>/dev/null)
        if [[ -n ${isMount} ]]; then
            # already used
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} has been used, will skip this device"
            continue
        fi
        isLvm=$(sfdisk -lqL 2>>/dev/null | grep $device | grep "8e.*Linux LVM")
        if [[ ! -n ${isLvm} ]]; then
            # part system type is not Linux LVM
            [[ -n "${DOCKER_BLOCK_DEVICES}" ]] && echo "Disk ${device} system type is not Linux LVM, will
skip this device"
            continue
        fi
    fi
    block_devices_size=$(lsblk -n -o SIZE $device 2>/dev/null | awk '{ print $1}')
    if [[ ${block_devices_size}"x" != "${dockerdisksize}Gx" ]] && [[ ${block_devices_size}"x" != "$
{systemdisksize}Gx" ]]; then
echo "n
p
1


w
" | fdisk $device
        mkfs -t ext4 ${device}1
        mkdir -p $mountdir
    uuid=$(blkid ${device}1 |awk '{print $2}')
    echo "${uuid}  $mountdir ext4  noatime  0 0" | tee -a /etc/fstab >/dev/null
        mount $mountdir
    fi
done
```

19.15 Workload Abnormalities

19.15.1 Fault Locating and Troubleshooting for Abnormal Workloads

If a workload is running improperly, you can view events to determine the cause and rectify the fault by referring to **Common Issues and Solutions**.

On the CCE console, choose **Workloads** > **Deployments** or **StatefulSets** in the navigation pane and click the name of the faulty workload. On the workload details page, view the workload events.

Viewing events

Step 1 Log in to the CCE console. In the navigation pane on the left, choose **Workloads** > **Deployments** or **Workloads** > **StatefulSets**.

Step 2 In the workload list, click the name of the abnormal workload.

Step 3 On the **Pods** tab page, view the latest workload events.

----End

Common Issues and Solutions

- Failed to Schedule an Instance
- What Should I Do If Image Re-pull Fails?
- What Should I Do If Container Restart Fails?
- What Should I Do If An Evicted Pod Exists?
- What Should I Do If a Pod Fails to be Evicted?
- What Should I Do If Pods in the Terminating State Cannot Be Deleted?
- What Should I Do If a Workload Is Stopped Caused by Pod Deletion?
- What Should I Do If Sandbox-Related Errors Are Reported When the Pod Remains in the Creating State?
- What Should I Do If a Pod Is in the Evicted State?
- What Should I Do If a Pod Is in the Evicted State?

19.15.2 Failed to Schedule an Instance

Fault Locating

- Viewing K8s Event Information
- Check Item 1: Whether a Node Is Available in the Cluster
- Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient
- Check Item 3: Affinity and Anti-Affinity Configuration of the Workload
- Check Item 4: Whether the Workload's Volume and Node Reside in the Same Zone

Viewing K8s Event Information

If the workload is in the Unready state and reports the "**InstanceSchedulingFailed**" event, check the workload's Kubernetes events to identify the cause.

In this example, the K8s event is **0/163 nodes are available: 133 Insufficient memory**, indicating that the memory is insufficient.

Complex scheduling failure information:

- **no nodes available to schedule pods**: No node resource is available for scheduling workload pods.
- **0/163 nodes are available: 133 Insufficient memory**: The node is available but the memory is insufficient.

- **163 Insufficient cpu**: The CPU is insufficient.
- **49 Insufficient nvidia.com/gpu**: The nvidia.com/gpu resources are insufficient.
- **49 InsufficientResourceOnSingleGPU**: GPU resources are insufficient.

Information interpretation:

- **0/163 nodes are available**: There are 163 nodes in the cluster, and no node meets the scheduling rules.
- **133 Insufficient memory**: The memory of 133 nodes is insufficient.
- **163 Insufficient cpu**: The CPUs of 163 nodes are insufficient.
- **49 Insufficient nvidia.com/gpu**: The GPUs of 49 nodes are insufficient.

The following is the fault locating procedure:

Check Item 1: Whether a Node Is Available in the Cluster

Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes** to check whether the node where the workload runs is in the available state.

For example, the event "0/1 nodes are available: 1 node(s) were not ready, 1 node(s) were out of disk space" indicates that the pod fails to be scheduled due to no available node.

Solution

- Add a node and migrate the pods to the new available node to ensure that services are running properly. Then, rectify the fault on the unavailable node. For details about the troubleshooting process, see the methods in the node FAQs.
- Create a new node or repair the faulty one.

Check Item 2: Whether Node Resources (CPU and Memory) Are Sufficient

If the requested workload resources exceed the available resources of the node where the workload runs, the node cannot provide the resources required to run new pods and pod scheduling onto the node will definitely fail.

Step 1 On the CCE console, choose **Workloads** > **Deployments** or **StatefulSets** in the navigation pane, click the workload name, and click **Pods** and then **Events** tabs to view pod events.

The event "0/1 nodes are available: 1 Insufficient cpu" indicates that the pod fails to be scheduled due to insufficient node resources.

Step 2 In the navigation pane, choose **Resource Management** > **Nodes** to view available CPUs and memory of the node where the workload runs.

In this example, 0.88 vCPUs and 0.8 GiB memory are available for the node.

Step 3 In the navigation pane, choose **Workloads** and click the workload name to view the workload's CPU request and memory request.

In this example, the CPU request is 2 vCPUs and the memory request is 0.5 GiB. The CPU request exceeds the available CPU resources, which causes pod scheduling to fail.

----End

Solution

On the ECS console, modify node specifications to expand node resources.

Check Item 3: Affinity and Anti-Affinity Configuration of the Workload

Inappropriate affinity policies will cause pod scheduling to fail.

```
0/1 nodes are available: 1 node(s) didn't match pod affinity/anti-affintity, 1 node(s) didn't match pod anti-affinity rules.
```

Solution

- When adding **workload-workload affinity** and **workload-node affinity** policies, ensure that the two types of policies do not conflict each other. Otherwise, workload deployment will fail. For example, workload deployment will fail if the following conditions are met:

  An anti-affinity relationship is established between workload 1 and workload 2. Workload 1 is deployed on node 1 while workload 2 is deployed on node 2.

  When you try to deploy workload 3 on node 3 and establish an affinity relationship with workload 2, a conflict occurs, resulting in a workload deployment failure.
- If the workload has a node affinity policy, make sure that **supportContainer** in the label of the affinity node is set to **true**. Otherwise, pods cannot be scheduled onto the affinity node and the following event is generated:
  ```
  No nodes are available that match all of the following predicates: MatchNode Selector, NodeNotSupportsContainer
  ```
  If **supportContainer** is set to **false**, the scheduling fails. The following figure shows the error information.
  ```
  0/1 nodes are available: 1
  ```

Check Item 4: Whether the Workload's Volume and Node Reside in the Same Zone

Pod scheduling fails if the workload's volume and node reside in different AZs.

```
0/1 nodes are available: 1 NoVolumeZoneConflict.
```

Solution

In the AZ where the workload's node resides, create a new volume. Alternatively, create an identical workload and select an automatically assigned cloud storage volume.

19.15.3 What Should I Do If Image Re-pull Fails?

If the workload details page displays an event indicating that image pulling fails, perform the following operations to locate the fault:

Fault Locating

- Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload
- Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used
- Check Item 3: Whether an Incorrect Secret Is Used or the Secret Expires When a Third-Party Image Is Used
- Check Item 4: Whether the Node Disk Space Is Insufficient

Check Item 1: Whether imagePullSecret Is Specified When You Use kubectl to Create a Workload

For example, when creating a deployment named **nginx**, check whether the YAML file contains the **imagePullSecrets** parameter, which indicates the secret name during image pulling.

- To pull an image from the Software Repository for Container, set this parameter to **default-secret**.
- To pull an image from a third-party image repository, set the **imagePullSecrets** parameter to the created secret name.

The example shows the situation when you fail to pull an image when **imagePullSecret** is not configured in your YAML file.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

Solution

When pulling an image from a third-party image repository, set **imagePullSecrets** to the created secret name.

Check Item 2: Whether the Image Address Is Correct When a Third-Party Image Is Used

CCE can create workloads using images pulled from a third-party image repository.

Enter the third-party image address according to requirements. The format must be **ip:port/path/name:version** or **name:version**. If no tag is specified, **latest** is used by default.

- For a private repository, enter an image address in the format of **ip:port/path/name:version**.
- For an open-source Docker repository, enter an image address in the format of **name:version**, for example, **nginx:latest**.

The following information is displayed when you fail to pull an image due to incorrect image address provided.

```
Failed to pull image "nginx:v1.1": rpc error:code = Unknown desc=Error response from daemon: manifest for nginx:v1.1 not found
```

Solution

You can either edit your YAML file to modify the image address or log in to the CCE console to replace the image on the **Upgrade** tag page of the workload details page.

Check Item 3: Whether an Incorrect Secret Is Used or the Secret Expires When a Third-Party Image Is Used

CCE can create workloads using images pulled from a third-party image repository.

Solution

Generally, a third-party image repository can be accessed only after authentication (using your account and password). CCE uses the secret authentication mode to pull images. Therefore, you need to create a secret for an image repository before pulling images from the repository.

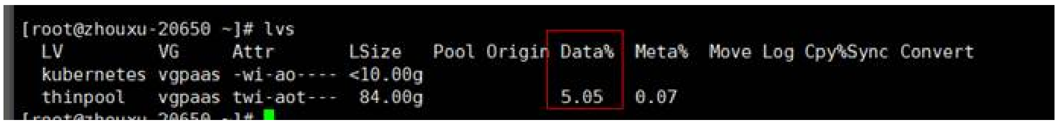
If your secret is incorrect or expires, images will fail to be pulled. In this case, create a new secret.

Check Item 4: Whether the Node Disk Space Is Insufficient

After a node is created in a cluster of v1.7.3-r7 or a later version, a 100 GB data disk dedicated for Docker is bound to the node. If the data disk space is insufficient, image fails to be pulled.

If the Kubernetes event contains the following information, the node has no disk space left for storing images. You need to clean up images or expand the disk capacity.

Run the **lvs** command to check the disk space for storing images on the node.



```
[root@zhouxu-20650 ~]# lvs
  LV         VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  kubernetes vgpaas -wi-ao---- <10.00g
  thinpool   vgpaas twi-aot---  84.00g             5.05   0.07
```

Run the following command to clean up images:

```
docker rmi -f {Image ID}
```

To expand the disk capacity, perform the following steps:

Step 1 Expand the capacity of the Docker disk on the EVS console.

Step 2 Log in to the target node.

Step 3 Run the following commands on the node to add the new disk capacity to the Docker disk:

```
pvresize /dev/vdb
lvextend -l+100%FREE -n vgpaas/thinpool
```

----End

19.15.4 What Should I Do If Container Restart Fails?

On the details page of a workload, if an event is displayed indicating that the container fails to be restarted, perform the following operations to locate the fault:

Step 1 Log in to the node where the abnormal workload is located.

Step 2 Check the ID of the container where the workload pod exits abnormally.

```
docker ps -a | grep $podName
```

Step 3 View the logs of the container.

```
docker logs $containerID
```

Rectify the fault of the workload based on logs.

Step 4 Check the error logs.

```
cat /var/log/messages | grep $containerID  | grep oom
```

Check whether the system OOM is triggered based on the logs.

----End

Fault Locating

- **Check Item 1: Whether There Are Processes that Keep Running in the Container**
- **Check Item 2: Whether Health Check Fails to Be Performed**
- **Check Item 3: Whether the User Service Has a Bug**
- **Check Item 4: Whether the Upper Limit of Container Resources Has Been Reached**
- **Check Item 5: Whether the Container Disk Space Is Insufficient**
- **Check Item 6: Whether the Resource Limits Are Improperly Set for the Container**
- **Check Item 7: Whether the Container Ports in the Same Pod Conflict with Each Other**
- **Check Item 8: Whether the Container Startup Command Is Correctly Configured**

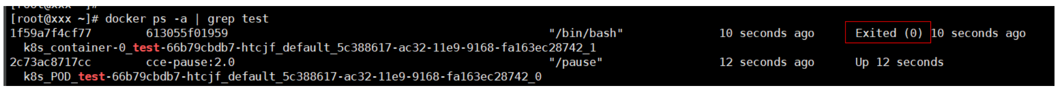
Check Item 1: Whether There Are Processes that Keep Running in the Container

Step 1 Log in to the node where the abnormal workload is located.

Step 2 View the container status.

```
docker ps -a | grep $podName
```

Example:



```
[root@xxx ~]# docker ps -a | grep test
1f59a7f4cf77        613055f01959                                              "/bin/bash"            10 seconds ago      Exited (0) 10 seconds ago
  k8s_container-0_test-66b79cbdb7-htcjf_default_5c388617-ac32-11e9-9168-fa163ec28742_1
2c73ac8717cc        cce-pause:2.0                                             "/pause"               12 seconds ago      Up 12 seconds
  k8s_POD_test-66b79cbdb7-htcjf_default_5c388617-ac32-11e9-9168-fa163ec28742_0
```

If no running process in the container, the status code **Exited (0)** is displayed.

----End

Check Item 2: Whether Health Check Fails to Be Performed

The health check configured for a workload is performed on services periodically. If an exception occurs, the pod reports an event and the pod fails to be restarted.

If the liveness-type (workload liveness probe) health check is configured for the workload and the number of health check failures exceeds the threshold, the containers in the pod will be restarted. On the workload details page, if Kubernetes events contain **Liveness probe failed: Get http...**, the health check fails.

Solution

On the workload details page, choose **Upgrade** > **Advanced Settings** > **Health Check** to check whether the health check policy is properly set and whether services are normal.

Check Item 3: Whether the User Service Has a Bug

Check whether the workload startup command is correctly executed or whether the workload has a bug.

Step 1 Log in to the node where the abnormal workload is located.

Step 2 Check the ID of the container where the workload pod exits abnormally.

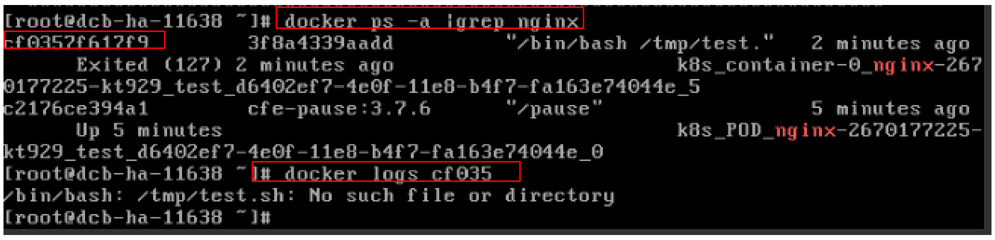
```
docker ps -a | grep $podName
```

Step 3 View the logs of the container.

```
docker logs $containerID
```

Note: In the preceding command, **containerID** indicates the ID of the container that has exited.

Figure 19-17 Incorrect startup command configuration of the container



```
[root@dcb-ha-11638 ~]# docker ps -a |grep nginx
cf0357f617f9        3f8a4339aadd        "/bin/bash /tmp/test."   2 minutes ago
      Exited (127) 2 minutes ago                       k8s_container-0_nginx-267
0177225-kt929_test_d6402ef7-4e0f-11e8-b4f7-fa163e74044e_5
c2176ce394a1        cfe-pause:3.7.6     "/pause"                 5 minutes ago
      Up 5 minutes                                     k8s_POD_nginx-2670177225-
kt929_test_d6402ef7-4e0f-11e8-b4f7-fa163e74044e_0
[root@dcb-ha-11638 ~]# docker logs cf035
/bin/bash: /tmp/test.sh: No such file or directory
[root@dcb-ha-11638 ~]#
```

As shown above, the container fails to be started due to an incorrect startup command. For other errors, rectify the bugs based on the logs.

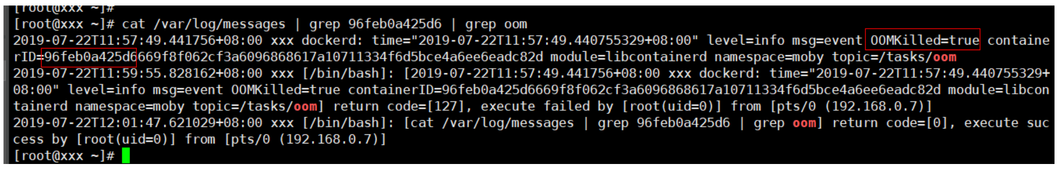
Solution: Re-create a workload and configure a correct startup command.

----End

Check Item 4: Whether the Upper Limit of Container Resources Has Been Reached

If the upper limit of container resources has been reached, OOM will be displayed in the event details as well as in the log:

```
cat /var/log/messages | grep 96feb0a425d6 | grep oom
```



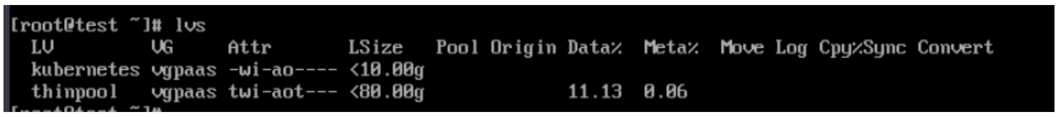
```
[root@xxx ~]#
[root@xxx ~]# cat /var/log/messages | grep 96feb0a425d6 | grep oom
2019-07-22T11:57:49.441756+08:00 xxx dockerd: time="2019-07-22T11:57:49.440755329+08:00" level=info msg=event OOMKilled=true containe
rID=96feb0a425d6669f8f062cf3a6096868617a10711334f6d5bce4a6ee6eadc82d module=libcontainerd namespace=moby topic=/tasks/oom
2019-07-22T11:59:55.828162+08:00 xxx [/bin/bash]: [2019-07-22T11:57:49.441756+08:00 xxx dockerd: time="2019-07-22T11:57:49.440755329+
08:00" level=info msg=event OOMKilled=true containerID=96feb0a425d6669f8f062cf3a6096868617a10711334f6d5bce4a6ee6eadc82d module=libcon
tainerd namespace=moby topic=/tasks/oom] return code=[127], execute failed by [root(uid=0)] from [pts/0 (192.168.0.7)]
2019-07-22T12:01:47.621029+08:00 xxx [/bin/bash]: [cat /var/log/messages | grep 96feb0a425d6 | grep oom] return code=[0], execute suc
cess by [root(uid=0)] from [pts/0 (192.168.0.7)]
[root@xxx ~]#
```

When a workload is created, if the requested resources exceed the configured upper limit, the system OOM is triggered and the container exits unexpectedly.

Check Item 5: Whether the Container Disk Space Is Insufficient

The following message refers to the Thin Pool disk that is allocated from the Docker disk selected during node creation. You can run the **lvs** command as user **root** to view the current disk usage.

```
Thin Pool has 15991 free data blocks which are less than minimum required 16383 free data blocks. Create more free space in thin pool or use dm.min_free_space option to change behavior
```



```
[root@test ~]# lvs
  LV         VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  kubernetes vgpaas -wi-ao---- <10.00g
  thinpool   vgpaas twi-aot--- <80.00g             11.13  0.06
```

Solution

1. Release used disk space.
   ```
   docker rmi -f `docker images | grep 20202 | awk '{print $3}'`
   ```
2. Expand the disk capacity. For details, see the method of expanding the data disk capacity of a node.

Check Item 6: Whether the Resource Limits Are Improperly Set for the Container

If the resource limits set for the container during workload creation are less than required, the container fails to be restarted.

```
Back-off restarting failed container
```

Solution

Modify the container specifications.

Check Item 7: Whether the Container Ports in the Same Pod Conflict with Each Other

Step 1 Log in to the node where the abnormal workload is located.

Step 2 Check the ID of the container where the workload pod exits abnormally.

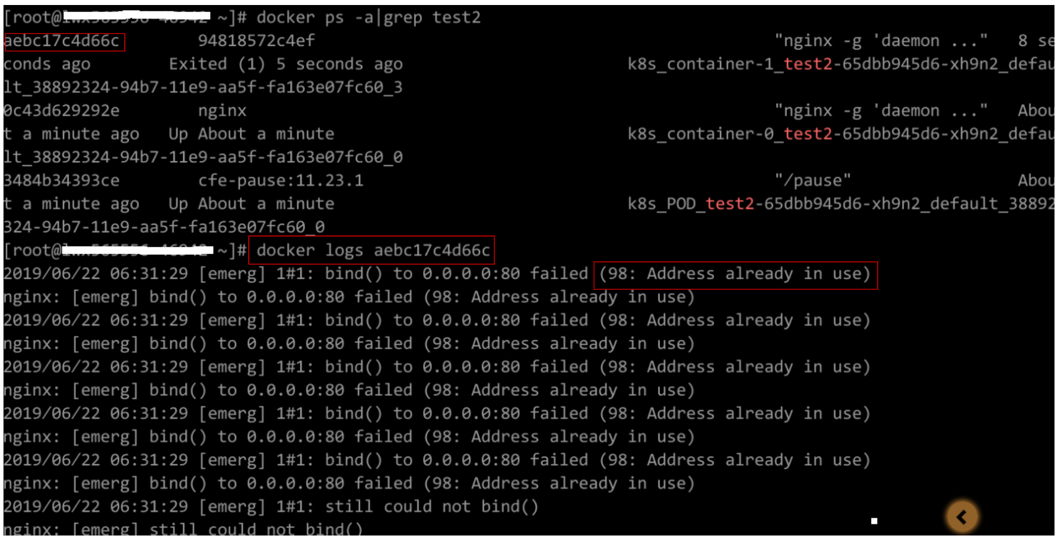
docker ps -a | grep *$podName*

Step 3 View the logs of the container.

docker logs *$containerID*

Rectify the fault of the workload based on logs. As shown in the following figure, container ports in the same pod conflict. As a result, the container fails to be started.

Figure 19-18 Container restart failure due to a container port conflict



```
[root@[redacted] ~]# docker ps -a|grep test2
aebc17c4d66c        94818572c4ef                                    "nginx -g 'daemon ..."   8 se
conds ago       Exited (1) 5 seconds ago                    k8s_container-1_test2-65dbb945d6-xh9n2_defau
lt_38892324-94b7-11e9-aa5f-fa163e07fc60_3
0c43d629292e        nginx                                           "nginx -g 'daemon ..."   Abou
t a minute ago   Up About a minute                          k8s_container-0_test2-65dbb945d6-xh9n2_defau
lt_38892324-94b7-11e9-aa5f-fa163e07fc60_0
3484b34393ce        cfe-pause:11.23.1                               "/pause"                 Abou
t a minute ago   Up About a minute                          k8s_POD_test2-65dbb945d6-xh9n2_default_38892
324-94b7-11e9-aa5f-fa163e07fc60_0
[root@[redacted] ~]# docker logs aebc17c4d66c
2019/06/22 06:31:29 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2019/06/22 06:31:29 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2019/06/22 06:31:29 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2019/06/22 06:31:29 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2019/06/22 06:31:29 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2019/06/22 06:31:29 [emerg] 1#1: still could not bind()
nginx: [emerg] still could not bind()
```

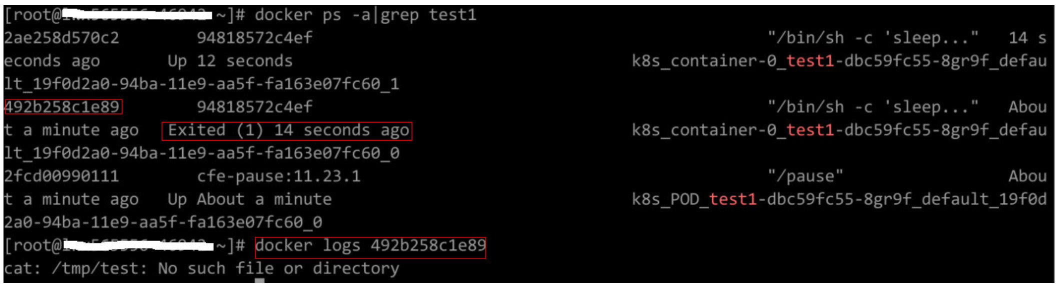
----End

Solution

Re-create the workload and set a port number that is not used by any other pod.

Check Item 8: Whether the Container Startup Command Is Correctly Configured

The error messages are as follows:



```
[root@[redacted] ~]# docker ps -a|grep test1
2ae258d570c2        94818572c4ef                                   "/bin/sh -c 'sleep..."   14 s
econds ago       Up 12 seconds                              k8s_container-0_test1-dbc59fc55-8gr9f_defau
lt_19f0d2a0-94ba-11e9-aa5f-fa163e07fc60_1
492b258c1e89        94818572c4ef                                   "/bin/sh -c 'sleep..."   Abou
t a minute ago   Exited (1) 14 seconds ago                  k8s_container-0_test1-dbc59fc55-8gr9f_defau
lt_19f0d2a0-94ba-11e9-aa5f-fa163e07fc60_0
2fcd00990111        cfe-pause:11.23.1                              "/pause"                 Abou
t a minute ago   Up About a minute                          k8s_POD_test1-dbc59fc55-8gr9f_default_19f0d
2a0-94ba-11e9-aa5f-fa163e07fc60_0
[root@[redacted] ~]# docker logs 492b258c1e89
cat: /tmp/test: No such file or directory
```

Solution

Log in to the CCE console. On the workload details page, click **Upgrade** > **Advanced Settings** > **Lifecycle** > **Startup Command** to see whether the startup command is correctly configured.

19.15.5 What Should I Do If An Evicted Pod Exists?

Pod actions are classified into the following two types:

- kube-controller-manager periodically checks the status of all nodes. If a node is in the NotReady state for a period of time, all pods on the node are evicted.
- kubelet periodically checks the memory and disk resources of the node. If the resources are insufficient, pods are evicted based on the priority. Check results will be recorded in kubelet logs of the node. You can run the following command to search for the information:
  ```
  cat /var/paas/sys/log/kubernetes/kubelet.log | grep -i Evicted -C3
  ```

After a pod is evicted and scheduled to a new node, if pods in that node are also being evicted, the pod will be evicted again. Pods may be evicted repeatedly.

If the eviction is triggered by kube-controller-manager, a pod in the Terminating state is left. It is automatically deleted only after the node where the container is located is restored. If the node has been deleted or cannot be restored due to other reasons, you can forcibly delete the pod.

If the eviction is triggered by kubelet, a pod in the Evicted state is left. It is only used for subsequent fault locating and can be directly deleted.

Common issues:

Why is an evicted container frequently scheduled to the original node?

Answer: A node evicts a container based on the node resource usage. The evicted container is scheduled based on the allocated node resources. Eviction and scheduling are based on different rules. Therefore, an evicted container may be scheduled to the original node again.

This problem can be solved by properly allocating resources to each container.

19.15.6 What Should I Do If a Pod Fails to be Evicted?

When a node is faulty, pods on the node are evicted to ensure workload availability. If the pods are not evicted when the node is faulty, perform the following steps:

Check Item 1: Whether Tolerations Have Been Configured on the Pod

Use kubectl or choose **More** > **Edit YAML** next to the corresponding workload to check whether tolerations are installed on the workload. For details, see **https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/**.

Check Item 2: Whether the Conditions for Stopping Pod Eviction Are Met

If the number of nodes in a cluster is smaller than 50 and the number of faulty nodes accounts for over 55% of the total nodes, the pod eviction will be suspended. In this case, K8s will attempt to evict the workload on the faulty node. For details, see **https://kubernetes.io/docs/concepts/architecture/nodes/**.

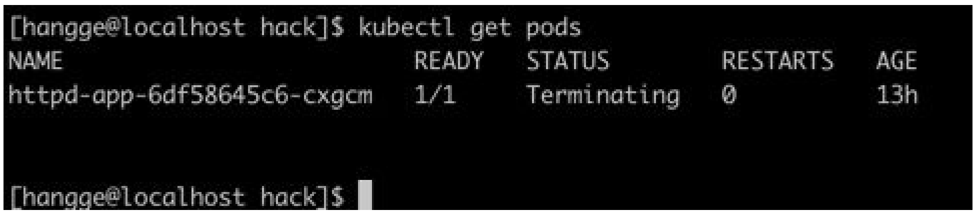
19.15.7 What Should I Do If Pods in the Terminating State Cannot Be Deleted?

Bug Fixed

When a node is in the Unavailable state, CCE migrates container pods on the node and sets the pods running on the node to the **Terminating** state.

After the node is restored, the pods in the **Terminating** state are automatically deleted.

However, some pods remain in the **Terminating** state.



```
[hangge@localhost hack]$ kubectl get pods
NAME                         READY   STATUS        RESTARTS   AGE
httpd-app-6df58645c6-cxgcm   1/1     Terminating   0          13h


[hangge@localhost hack]$ 
```

These pods cannot be deleted by running the **kubectl delete pods** command.

```
kubectl delete pods httpd-app-6df58645c6-cxgcm
```

Solution

You can run the following command to forcibly delete the pods created in any ways:

```
kubectl delete pods <pod> --grace-period=0 --force
```

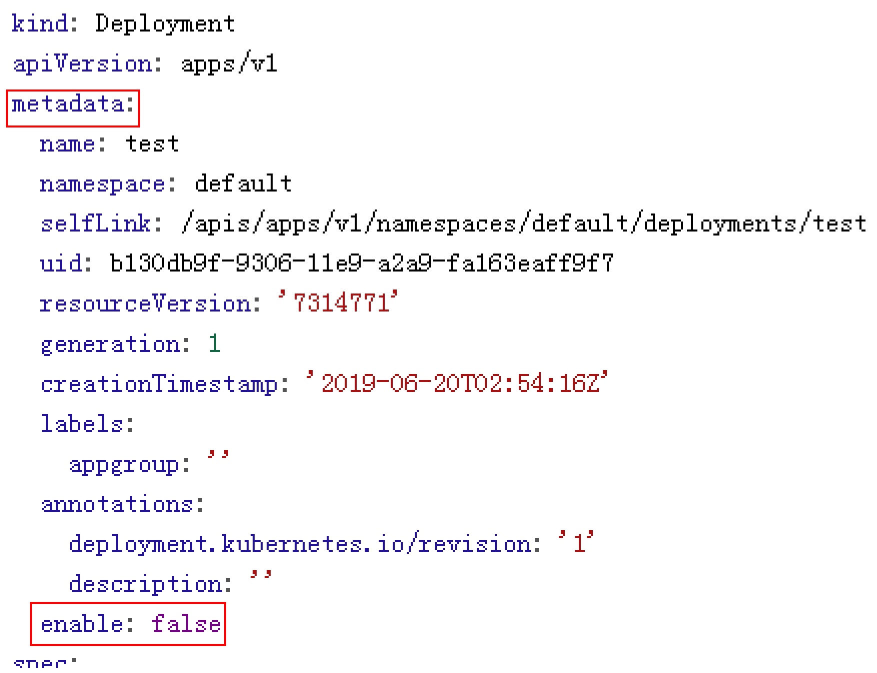
Therefore, you can run the following command to delete the pod **httpd-app-6df58645c6-cxgcm**:

```
kubectl delete pods httpd-app-6df58645c6-cxgcm --grace-period=0 --force
```

19.15.8 What Should I Do If a Workload Is Stopped Caused by Pod Deletion?

Cause:

The **metadata.enable** field in the YAML file of the workload is **false**. As a result, the pod of the workload is deleted and the workload is in the stopped status.



```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: test
  namespace: default
  selfLink: /apis/apps/v1/namespaces/default/deployments/test
  uid: b130db9f-9306-11e9-a2a9-fa163eaff9f7
  resourceVersion: '7314771'
  generation: 1
  creationTimestamp: '2019-06-20T02:54:16Z'
  labels:
    appgroup: ''
  annotations:
    deployment.kubernetes.io/revision: '1'
    description: ''
  enable: false
spec:
```

Symptom

The workload status is **Stopped**.

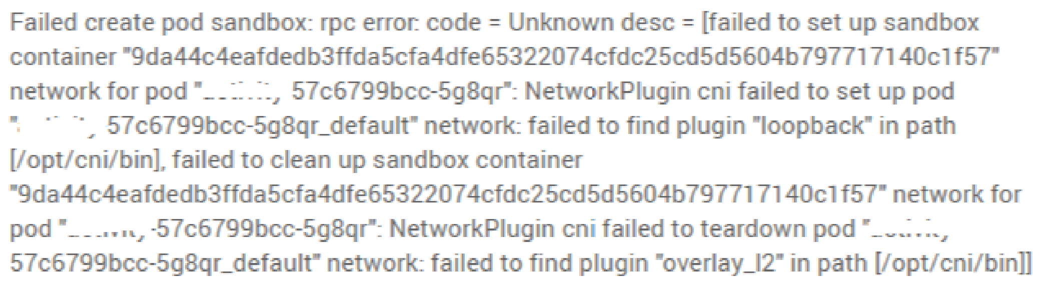
Solution

Delete the **enable** field or set it to **true**.

19.15.9 What Should I Do If Sandbox-Related Errors Are Reported When the Pod Remains in the Creating State?

Symptom

The pod remains in the creating state for a long time, and the sandbox-related errors are reported.



```
Failed create pod sandbox: rpc error: code = Unknown desc = [failed to set up sandbox
container "9da44c4eafdedb3ffda5cfa4dfe65322074cfdc25cd5d5604b797717140c1f57"
network for pod "[redacted]57c6799bcc-5g8qr": NetworkPlugin cni failed to set up pod
"[redacted]57c6799bcc-5g8qr_default" network: failed to find plugin "loopback" in path
[/opt/cni/bin], failed to clean up sandbox container
"9da44c4eafdedb3ffda5cfa4dfe65322074cfdc25cd5d5604b797717140c1f57" network for
pod "[redacted]-57c6799bcc-5g8qr": NetworkPlugin cni failed to teardown pod "[redacted]
57c6799bcc-5g8qr_default" network: failed to find plugin "overlay_l2" in path [/opt/cni/bin]]
```

Solution

Select a troubleshooting method for your cluster:

Clusters of V1.13

1. Sandbox errors are generally caused by the abnormal startup of the container component on the node. You can run the **systemctl status canal** command to check the container component and run the **systemctl restart canal** command to restart the component.
2. All container components on the node are normal, but the **loopback** file of the CNI is missing. The error is as follows: **network: failed to find plugin "loopback" in path [/opt/cni/bin]**. You can copy a complete version of the **loopback** file from the current region or other regions which share the same cluster versions (minor releases can be ignored), and put the **loopback** file in the path **/opt/cni/bin/**. Then, restart the canal component.

Clusters earlier than V1.13

1. Sandbox errors are generally caused by the abnormal startup of the container components on the node. You can run the **su paas -c '/var/paas/monit/bin/monit summary'** command to check the container component and run the **su paas -c '/var/paas/monit/bin/monit restart canal'** command to restart the component.
2. All container components on the node are normal, but the **loopback** file of the CNI is missing. The error is as follows: **network: failed to find plugin "loopback" in path [/opt/cni/bin]**. You can copy a complete version of the **loopback** file from the current region or other regions which share the same cluster versions (minor releases can be ignored), and put the **loopback** file in the path **/opt/cni/bin/**. Then, restart the canal component.

19.15.10 What Should I Do If a Pod Is in the Evicted State?

Symptom

Workload pods in the cluster fail and are being redeployed constantly.

After the following command is run, the command output shows that many pods are in the evicted state.

```
kubectl get pods
```

Possible Cause

When a node is abnormal, Kubernetes evicts pods on the node. This problem is often caused by insufficient resources.

Solution

Check resource usage and locate the fault.

Run the following command to delete the evicted pods:

```
kubectl get pods | grep Evicted | awk '{print $1}' | xargs kubectl delete pod
```

Reference

Kubelet does not delete evicted pods

19.15.11 What Should I Do If the OOM Killer Is Triggered When a Container Uses Memory Resources More Than Limited?

If a node has sufficient memory resources, a container on this node can use more memory resources than requested, but no more than limited. If the memory allocated to a container exceeds the upper limit, the container is stopped first. If the container continuously uses memory resources more than limited, the container is terminated. If a stopped container is allowed to be restarted, kubelet will restart it, but other types of run errors will occur.

Scenario 1

The node's memory has reached the memory limit reserved for the node. As a result, OOM killer is triggered.

Solution

You can either scale up the node or migrate the pods on the node to other nodes.

Scenario 2

The upper limit of resources configured for the pod is too small. When the actual usage exceeds the limit, OOM killer is triggered.

Solution

Set a higher upper limit for the workload.

Example

A pod will be created and allocated memory that exceeds the limit. As shown in the following configuration file of the pod, the pod requests 50 MB memory and the memory limit is set to 100 MB.

Example YAML file (memory-request-limit-2.yaml):

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-2
spec:
  containers:
  - name: memory-demo-2-ctr
    image: vish/stress
    resources:
      requests:
        memory: 50Mi
      limits:
        memory: "100Mi"
    args:
    - -mem-total
    - 250Mi
    - -mem-alloc-size
    - 10Mi
    - -mem-alloc-sleep
    - 1s
```

The **args** parameters indicate that the container attempts to request 250 MB memory, which exceeds the pod's upper limit (100 MB).

Creating a pod:

```
kubectl create -f https://k8s.io/docs/tasks/configure-pod-container/memory-request-limit-2.yaml --
namespace=mem-example
```

Viewing the details about the pod:

```
kubectl get pod memory-demo-2 --namespace=mem-example
```

In this stage, the container may be running or be killed. If the container is not killed, repeat the previous command until the container is killed.

```
NAME            READY     STATUS      RESTARTS   AGE
memory-demo-2   0/1       OOMKilled   1          24s
```

Viewing detailed information about the container:

```
kubectl get pod memory-demo-2 --output=yaml --namespace=mem-example
```

This output indicates that the container is killed because the memory limit is exceeded.

```
lastState:
   terminated:
     containerID: docker://7aae52677a4542917c23b10fb56fcb2434c2e8427bc956065183c1879cc0dbd2
     exitCode: 137
     finishedAt: 2020-02-20T17:35:12Z
     reason: OOMKilled
     startedAt: null
```

In this example, the container can be automatically restarted. Therefore, kubelet will start it again. You can run the following command several times to see how the container is killed and started:

```
kubectl get pod memory-demo-2 --namespace=mem-example
```

The preceding command output indicates how the container is killed and started back and forth:

```
stevepe@sperry-1:~/steveperry-53.github.io$ kubectl get pod memory-demo-2 --namespace=mem-example
NAME            READY     STATUS      RESTARTS   AGE
memory-demo-2   0/1       OOMKilled   1          37s
stevepe@sperry-1:~/steveperry-53.github.io$ kubectl get pod memory-demo-2 --namespace=mem-example
NAME            READY     STATUS    RESTARTS   AGE
memory-demo-2   1/1       Running   2          40s
```

Viewing the historical information of the pod:

```
kubectl describe pod memory-demo-2 --namespace=mem-example
```

The following command output indicates that the pod is repeatedly killed and started.

```
... Normal  Created   Created container with id
66a3a20aa7980e61be4922780bf9d24d1a1d8b7395c09861225b0eba1b1f8511
... Warning BackOff   Back-off restarting failed container
```

19.16 What Should I Do If a Service Released in a Workload Cannot Be Accessed from Public Networks?

A workload can be accessed from public networks through a load balancer. LoadBalancer provides higher reliability than EIP-based NodePort because an EIP is no longer bound to a single node. The LoadBalancer access type is applicable to the scenario in which a Service exposed to public networks is required.

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**.

Fault Locating

- Check Item 1: Container and Container Port
- Check Item 2: Node IP Address and Node Port
- Check Item 3: ELB IP Address and Port

Check Item 1: Container and Container Port

Log in to the CCE console or use kubectl to query the IP address of the pod. Then, log in to the node or container in the cluster and run the curl command to manually call the API. Check whether the expected result is returned.

If <container IP address>:<port> cannot be accessed, you are advised to log in to the application container and access <127.0.0.1>:<port> to locate the fault.

Common issues:

1. The container port is incorrectly configured (the container does not listen to the access port).
2. The URL does not exist (no related path exists in the container).
3. A Service exception (a Service bug in the container) occurs.

Check Item 2: Node IP Address and Node Port

Only **NodePort** or **LoadBalancer** Services can be accessed using the node IP address and node port.

- **NodePort Services:**

  The access port of a node is the port exposed externally by the node.
- **LoadBalancer Service:**

  You can view the node port of a LoadBalancer Service by editing the YAML file.

Example:

nodePort: 30637 is the port exposed externally by the node, **targetPort: 80** is the port exposed by the container, and **port: 123** is the port exposed by the ELB.

```
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 123
      targetPort: 80
      nodePort: 30637
```

After finding the node port, access <IP address>:<port> of the node where the container is located and check whether the expected result is returned.

Common issues:

1. The service port is not allowed in the inbound rules of the node.
2. A custom route is incorrectly configured for the node.
3. The label of the pod does not match that of the Service (created using kubectl or API).

Check Item 3: ELB IP Address and Port

There are several possible causes if <IP address>:<port> of the ELB cannot be accessed, but <IP address>:<port> of the node can be accessed.

Possible causes:

- The backend server group of the port or URL does not meet the expectation.
- The security group on the node has not exposed the related protocol or port to the ELB.
- The health check of the layer-4 load balancing is not enabled. (If so, enable it.)
- The certificate used for Services of layer-7 load balancing has expired.

Common issues:

1. The health check function is disabled (enabled by default) when a layer-4 load balancing Service is released. As a result, a new backend server group fails to be added when you update the backend server groups.
2. For UDP access, the ICMP port of the node has not been allowed in the inbound rules.
3. The label of the pod does not match that of the Service (created using kubectl or API).

19.17 Selecting a Network Model When Creating a Cluster on CCE

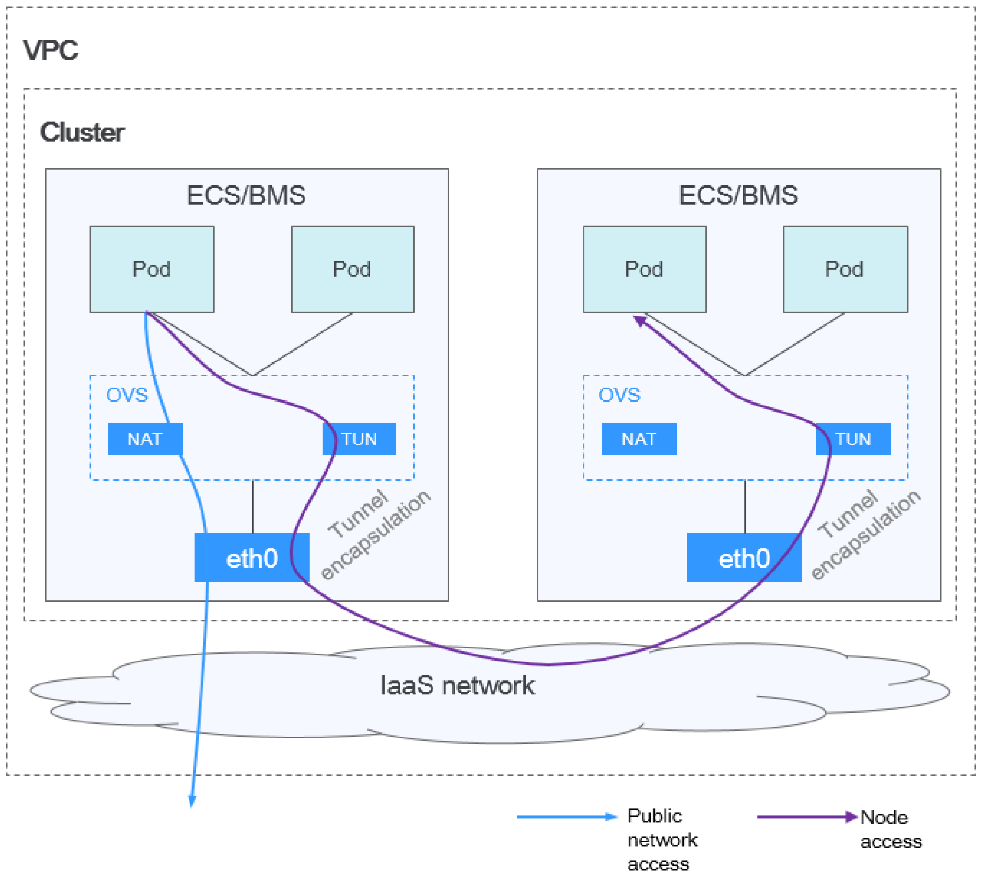
CCE uses high-performance container networking add-ons, which support the tunnel network, VPC network, and Yangtse network models.

CAUTION

After a cluster is created, the network model cannot be changed. Exercise caution when selecting a network model.

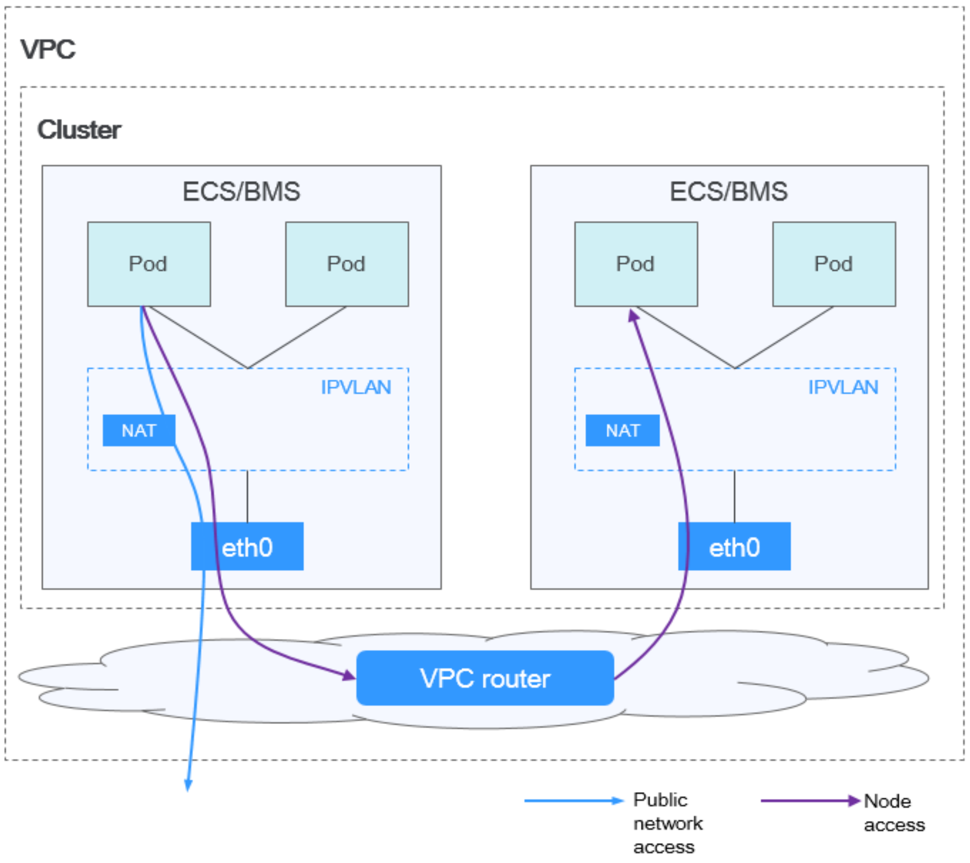
- **Tunnel network**: The container network is an overlay tunnel network on top of a VPC network and uses the VXLAN technology. This network model is applicable when there is no high requirements on performance. VXLAN encapsulates Ethernet packets as UDP packets for tunnel transmission. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.

Figure 19-19 Container tunnel network



- **VPC network**: The container network uses VPC routing to integrate with the underlying network. This network model is applicable to performance-intensive scenarios. The maximum number of nodes allowed in a cluster depends on the route quota in a VPC network. Each node is assigned a CIDR block of a fixed size. VPC networks are free from tunnel encapsulation overhead and outperform container tunnel networks. In addition, as VPC routing includes routes to node IP addresses and the container CIDR block, container pods in the cluster can be directly accessed from outside the cluster.

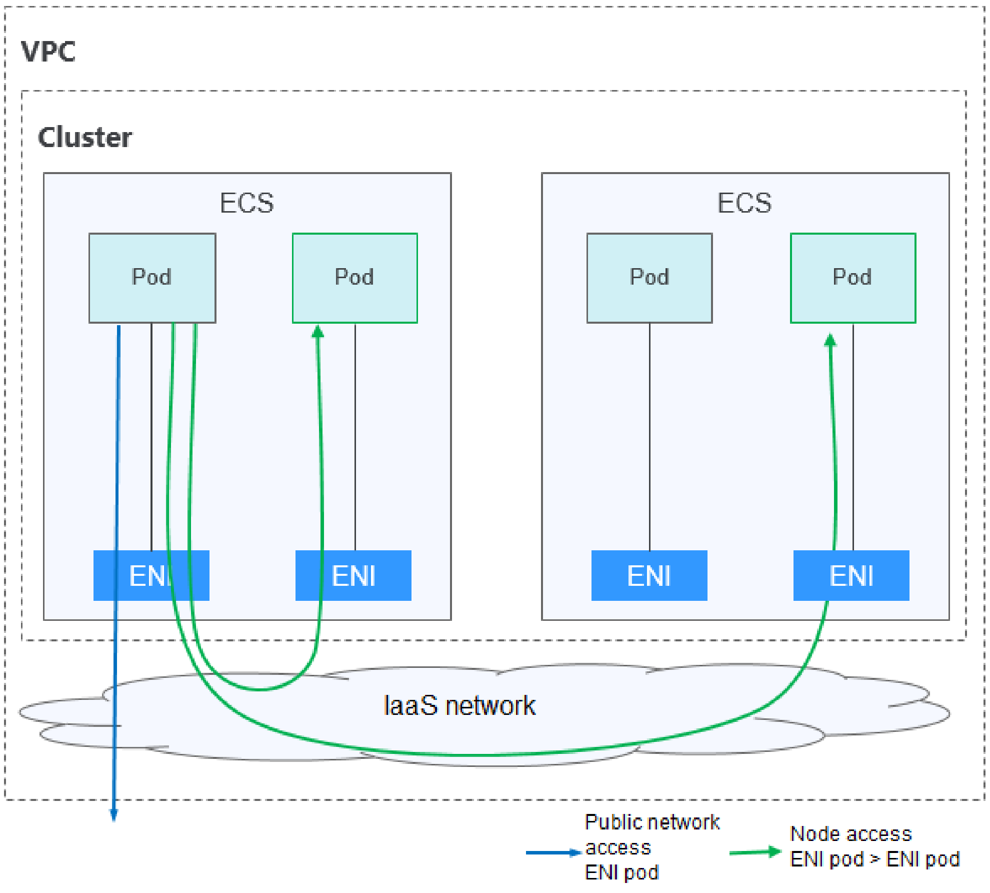
Figure 19-20 VPC network



- **Yangtse**: The container network deeply integrates the native elastic network interface (ENI) capability of VPC, uses the VPC CIDR block to allocate container addresses, and supports direct traffic distribution to containers through a load balancer to deliver high performance.

  The Yangtse network model is now in the open beta test (OBT) phase. After a cluster is created, the network model cannot be changed. Therefore, exercise caution when selecting the network model.

Figure 19-21 Yangtse network



The following table lists the differences between the network models.

Table 19-9 Network comparison

Dimension	Tunnel Network	VPC Network	Yangtse
Core component	OVS	IPVlan	ENI
Applicable clusters	Hybrid cluster VM cluster	Hybrid cluster VM cluster	CCE Turbo cluster
Support for network policies (networkpolicy)	Yes	No	You can use the security group of the VPC network to implement this function.

Dimension	Tunnel Network	VPC Network	Yangtse
Support for ENI	No	Yes. The container network is deeply integrated with the VPC network, and ENI is used for pods to communicate.	Yes. ENI is used for pods to communicate.
IP address management	IP addresses can be migrated.	● Each node is allocated with a small subnet. ● A static route is added on the VPC router with the next hop set to the node IP address.	A VPC subnet can be customized and used to allocate IP addresses to pods.
Network performance	Performance loss due to VXLAN tunnel encapsulation	● No performance loss as no tunnel encapsulation is required; performance comparable to bare metal networks ● Data forwarded across nodes through the VPC router	● The bottom-layer VPC network capability is fully used; no dependency on the VPC router. ● High-performance direct connection between ELB and containers
Networking scale	A maximum of 2,000 nodes are supported.	Limited by the VPC route table.	● Cluster scale: unlimited ● The number of pods on a single node is limited by the number of ENIs supported by the ECS instance. The maximum number of NICs must be greater than the number of add-ons you install.
External dependency	None	Static route table of the VPC router	ENI capability

Dimension	Tunnel Network	VPC Network	Yangtse
Application scenarios	• Common container service scenarios • Scenarios that do not have high requirements on network latency and bandwidth	• Scenarios that have high requirements on network latency and bandwidth • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.	• Scenarios that have high requirements on network latency, bandwidth, and performance • Containers can communicate with VMs using a microservice registration framework, such as Dubbo and CSE.

NOTICE

1. The actual cluster scale is limited by the quota of custom routes of the VPC. Therefore, estimate the number of required nodes before creating a VPC.
2. By default, the VPC network model supports direct communication between containers and hosts in the same VPC. If a peering connection policy is configured between the VPC and another VPC, the containers can directly communicate with hosts on the peer VPC. In addition, in hybrid networking scenarios such as Direct Connect and VPN, communication between containers and hosts on the peer end can also be achieved with proper planning.

19.18 Planning CIDR Blocks for a CCE Cluster

Before creating a cluster on CCE, determine the number of VPCs, number of subnets, container CIDR blocks, and Services for access based on service requirements.

This section describes the functions of various addresses in a CCE cluster in a VPC and how to plan CIDR blocks.

Basic Concepts

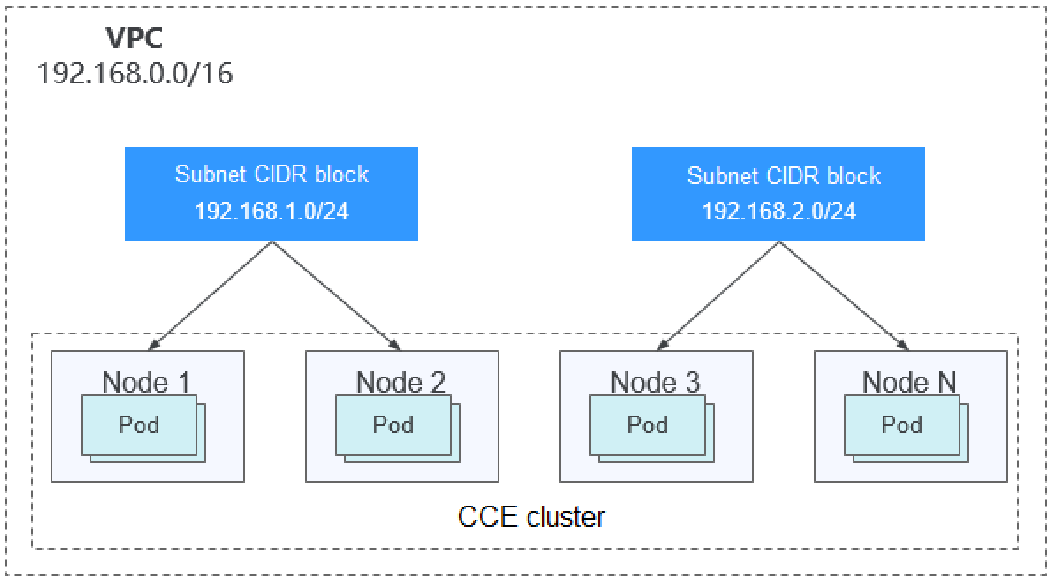
VPC CIDR Block

Virtual Private Cloud (VPC) enables you to provision logically isolated, configurable, and manageable virtual networks for cloud servers, cloud containers, and cloud databases. You have complete control over your virtual network, including selecting your own CIDR block, creating subnets, and configuring security groups. You can also assign EIPs and allocate bandwidth in your VPC for secure and easy access to your business system.

Subnet CIDR Block

A subnet is a network that manages ECS network planes. It supports IP address management and DNS. The IP addresses of all ECSs in a subnet belong to the subnet.

Figure 19-22 VPC CIDR block architecture



By default, ECSs in all subnets of the same VPC can communicate with one another, while ECSs in different VPCs cannot communicate with each other.

You can create VPC peering connections to enable ECSs in different VPCs to communicate with one another.

Container (Pod) CIDR Block

Pod is a Kubernetes object. Each pod has an IP address.

When creating a cluster on CCE, you can specify the pod (container) CIDR block, which cannot overlap with the subnet CIDR block. For example, if the subnet CIDR block is 192.168.0.0/16, the container CIDR block cannot be 192.168.0.0/18 or 192.168.1.0/18, because these addresses are included in 192.168.0.0/16.

Service CIDR Block

Service is also a Kubernetes object. Each Service has an address. When creating a cluster on CCE, you can specify the Service CIDR block. Similarly, the Service CIDR block cannot overlap with the subnet CIDR block or the container CIDR block. The Service CIDR block can be used only within a cluster.

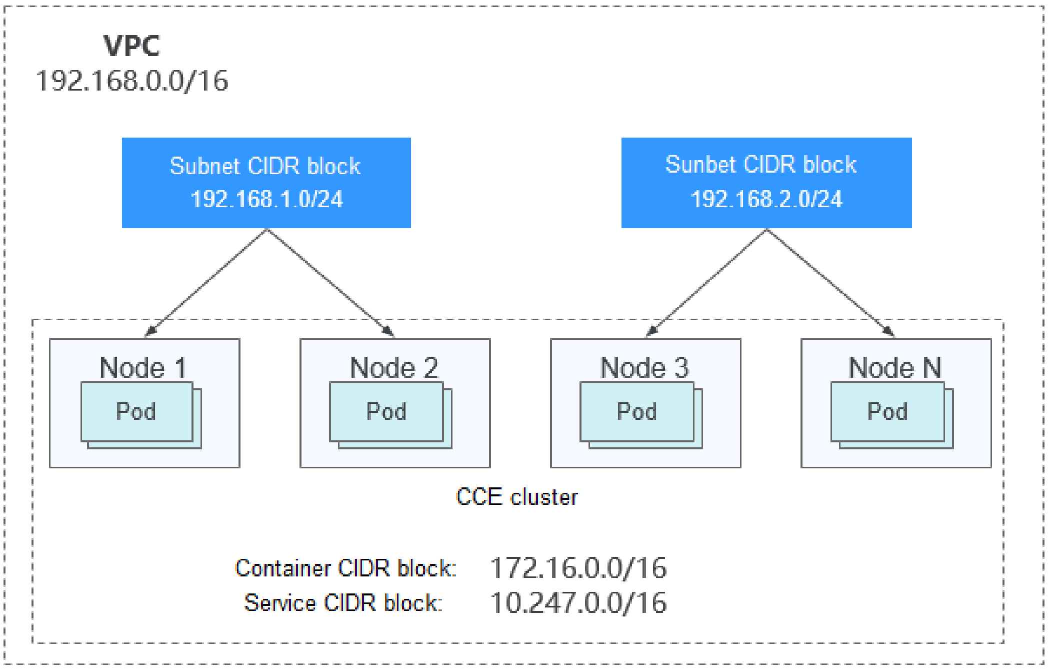
For details about the relationship between these CIDR blocks, see **Figure 19-23**.

How Do I Select a CIDR Block?

Single-VPC Single-Cluster Scenarios

These are the simplest scenarios. The VPC CIDR block is determined when the VPC is created. When creating a CCE cluster, select a CIDR block different from that of the current VPC.

Figure 19-23 CIDR block in the single-VPC single-cluster scenario



Single-VPC Multi-Cluster Scenarios

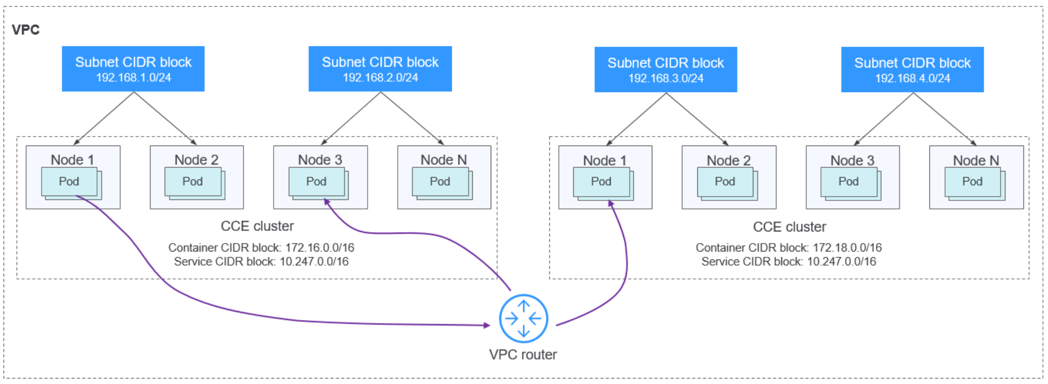
Multiple CCE clusters are created in a VPC.

In the **VPC network** mode, pod packets are forwarded through VPC routes. CCE automatically configures a routing table on the VPC routes to each container CIDR block.

Pay attention to the following:

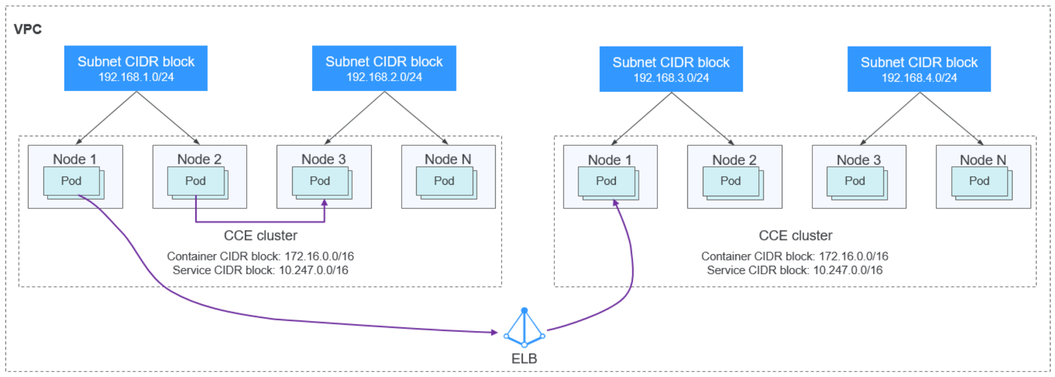
- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can. In this case, CCE clusters are partially interconnected. A pod of a cluster can directly access the pods of another cluster, but cannot access the Services of the cluster.
- The network scale is limited by the VPC route table.

Figure 19-24 VPC network - multi-cluster scenario



In the tunnel network model, the container network is an overlay network plane deployed over the VPC network. Though at some cost of performance, the tunnel encapsulation enables higher interoperability and compatibility with advanced features (such as network policy-based isolation), meeting the requirements of most applications.

Figure 19-25 Tunnel network - multi-cluster scenario


VPC
Subnet CIDR block 192.168.1.0/24
Subnet CIDR block 192.168.2.0/24
Subnet CIDR block 192.168.3.0/24
Subnet CIDR block 192.168.4.0/24
Node 1
Pod
Node 2
Pod
Node 3
Pod
Node N
Pod
CCE cluster
Container CIDR block: 172.16.0.0/16
Service CIDR block: 10.247.0.0/16
Node 1
Pod
Node 2
Pod
Node 3
Pod
Node N
Pod
CCE cluster
Container CIDR block: 172.16.0.0/16
Service CIDR block: 10.247.0.0/16
ELB

Pay attention to the following:

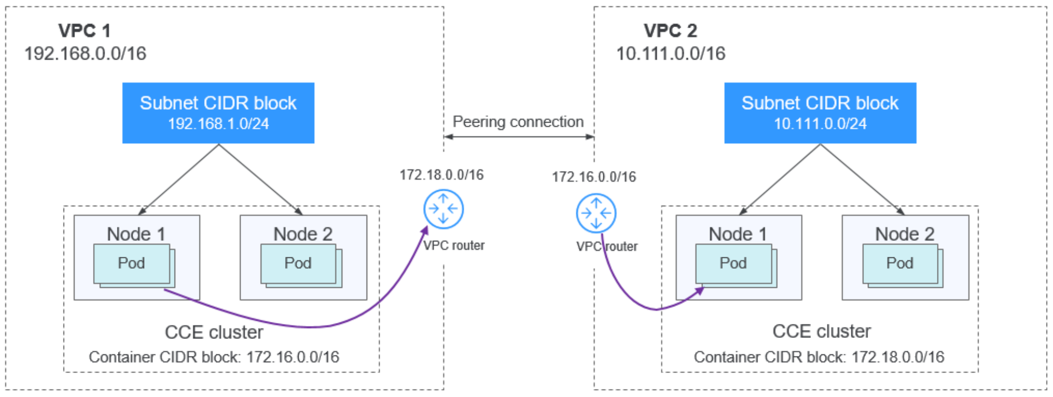
- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters can overlap, so do the Service CIDR blocks.
- It is recommended that ELB be used for the cross-cluster access between containers.

VPC Interconnection Scenarios

When two VPC networks are interconnected, you can configure the packets to be sent to the peer VPC in the route table.

In the VPC network model, after creating a peering connection, you need to add routes for the peering connection to enable communication between the two VPCs.

Figure 19-26 VPC Network - VPC interconnection scenario

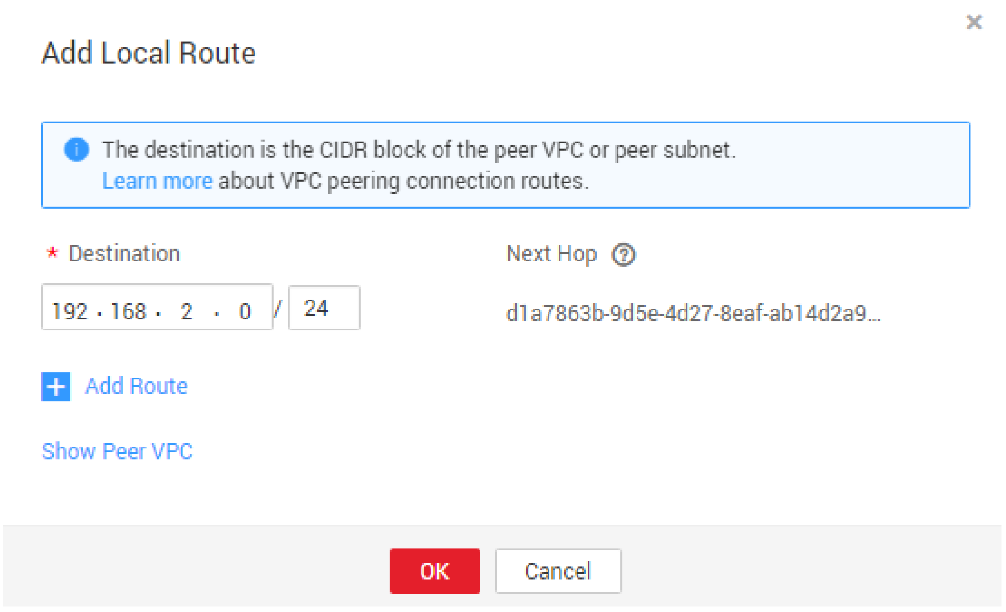


To interconnect cluster containers across VPCs, you need to create VPC peering connections.

Pay attention to the following:

- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can.
- Add the peer container CIDR block to the route table of the VPC peering connection. The following is an example:

Figure 19-27 Adding the peer container CIDR block to the local route on the VPC console



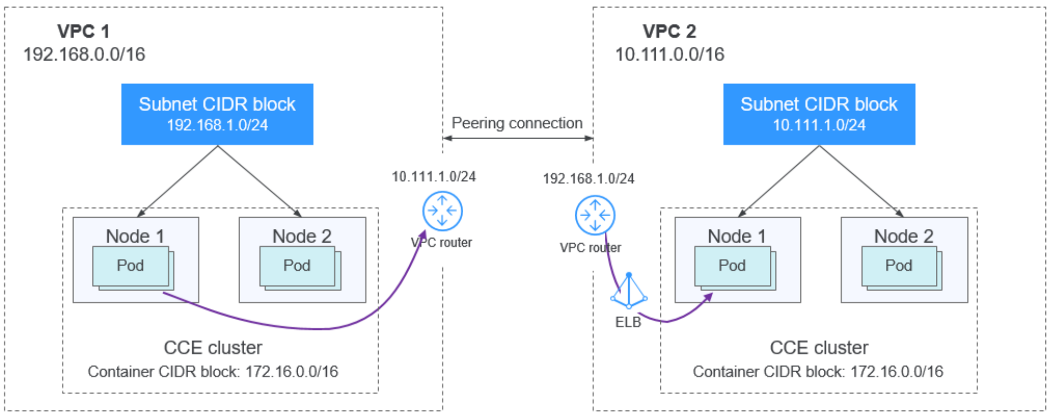
Add Local Route

The destination is the CIDR block of the peer VPC or peer subnet.
Learn more about VPC peering connection routes.

* Destination	Next Hop
192 . 168 . 2 . 0 / 24	d1a7863b-9d5e-4d27-8eaf-ab14d2a9...

In the tunnel network model, after creating a peering connection, you need to add routes for the peering connection to enable communication between the two VPCs.

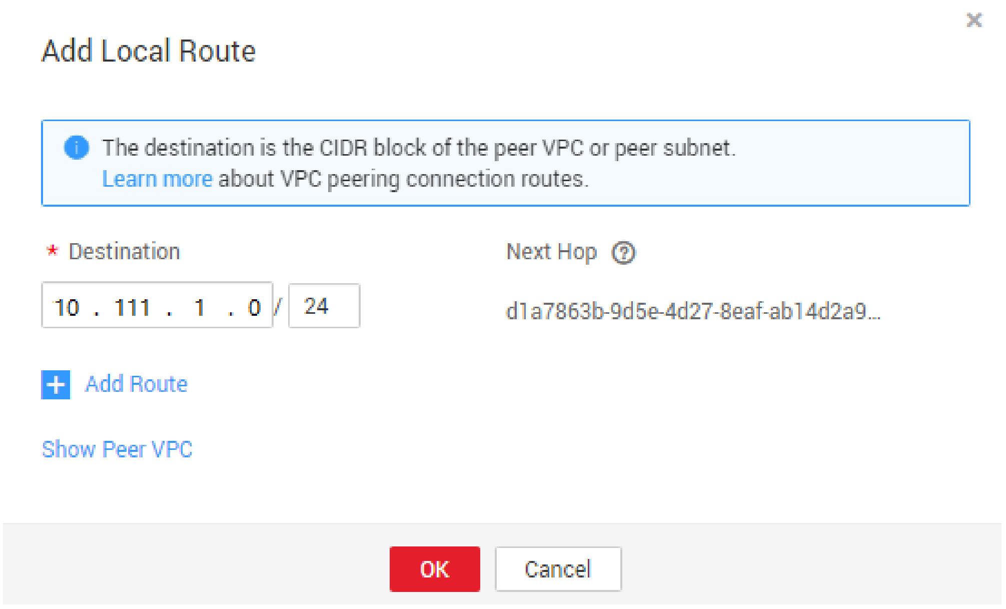
Figure 19-28 Tunnel network - VPC interconnection scenario



Pay attention to the following:

- The VPC address is determined during VPC creation. When creating a cluster, select a CIDR block for each cluster that does not overlap with the VPC CIDR block or other container CIDR blocks.
- The container CIDR blocks of all clusters cannot overlap, but the Service CIDR blocks can.
- Add the peer subnet CIDR block to the route table of the VPC peering connection. The following is an example:

Figure 19-29 Adding the subnet CIDR block of the peer cluster node to the local route on the VPC console



Add Local Route

The destination is the CIDR block of the peer VPC or peer subnet.
Learn more about VPC peering connection routes.

* Destination	Next Hop
10 . 111 . 1 . 0 / 24	d1a7863b-9d5e-4d27-8eaf-ab14d2a9...

VPC-IDC Scenarios

Similar to the VPC interconnection scenario, some CIDR blocks in the VPC are routed to the IDC. The pod IP addresses of CCE clusters cannot overlap with the addresses within these CIDR blocks. To access the pod IP addresses in the cluster in the IDC, you need to configure the route table to the private line VBR on the IDC.

19.19 What Is the Relationship Between Clusters, VPCs, and Subnets?

A VPC is similar to a private local area network (LAN) managed by a home gateway whose IP address is 192.168.0.0/16. A VPC is a private network built on the cloud and provides basic network environment for running elastic cloud servers (ECSs), elastic load balancers (ELBs), and middleware. Networks of different scales can be configured based on service requirements. Generally, you can set the CIDR block to 10.0.0.0/8–24, 172.16.0.0/12–24, or 192.168.0.0/16–24. The largest CIDR block is 10.0.0.0/8, which corresponds to a class A network.

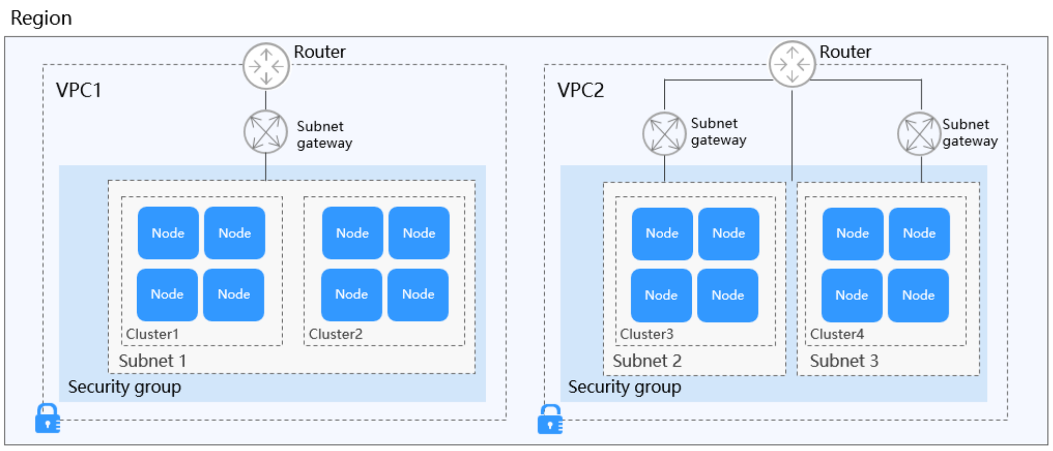
A VPC can be divided into multiple subnets. Security groups are configured to determine whether these subnets can communicate with each other. This ensures that subnets can be isolated from each other, so that you can deploy different services on different subnets.

A cluster consists of one or more ECSs (also known as nodes) in the same VPC. It provides a computing resource pool for running containers.

As shown in **Figure 19-30**, a region may comprise multiple VPCs. A VPC consists of one or more subnets. The subnets communicate with each other through a subnet gateway. A cluster is created in a subnet. There are three scenarios:

- Different clusters are created in different VPCs.
- Different clusters are created in the same subnet.
- Different clusters are created in different subnets.

Figure 19-30 Relationship between clusters, VPCs, and subnets



20 Migrating Data from CCE 1.0 to CCE 2.0

Perform the following operations to migrate data from CCE 1.0 to CCE 2.0:

1. **Migrating Images**: Pull images from the image repository of CCE 1.0, and push them to the image repository of CCE 2.0.
2. **Migrating clusters**: Create VM clusters in CCE 2.0 for application deployment.
3. **Migrating applications**: Move applications deployed in CCE 1.0 to the new clusters of CCE 2.0.

20.1 Differences Between CCE 1.0 and CCE 2.0

CCE 2.0 inherits and modifies the features of CCE 1.0, and release new features.

- Modified features:
  - Clusters in CCE 1.0 are equivalent to VM clusters in CCE 2.0.
  - CCE 2.0 does not have the concept of component template. In CCE 1.0, you can set parameters in the component template when creating an application.
  - Applications in CCE 1.0 are equivalent to Deployments in CCE 2.0. In addition, the concept of StatefulSet is added in CCE 2.0.
- New Features:
  - EVS disks and file storage can be created and directly mounted during workload creation on the CCE console.
  - Auto scaling for cluster nodes.
  - Docker upgraded to the 1706 version.
  - Customized helm charts for orchestrating workloads.
- In CCE 2.0, before uploading an image, you must create an image repository.

The following table compares the two versions.

Table 20-1 Comparison between CCE 1.0 and CCE 2.0

CCE 1.0	CCE 2.0
Container registry	Image repository
Cluster	Resource Management > VM Cluster
Component template	-
App Designer	-
App Manager	Deployment

20.2 Migrating Images

Migrate the images stored in the image repository of CCE 1.0 to CCE 2.0.

This section takes the Apache image as an example to show how to migrate images step by step.

Procedure

Before pushing container images, ensure that the local Docker client can access your private container registry.

Step 1 Log in to the Docker client as the **root** user.

Step 2 Create an organization. Organizations are used to isolate images and assign access permissions (read, edit, and manage) to different users.

1. Log in to the CCE console. In the navigation pane, choose **Image Repository**.
2. On the Image Repository dashboard page, click **Create Organization**.
3. Enter an organization name, that is, the domain name. When the configuration is complete, click **OK**.

Step 3 Log in to the Docker client as the **root** user.

Step 4 Pull the images stored in the image repository of CCE 1.0 to a local directory.

NOTE

To pull images, log in to the CCE 1.0 console. In the navigation pane, choose **Image Repository**. Click the name of the target images. On the image details pages, you can find corresponding image pull command.

Example:

docker pull *10.154.57.150:443/test/apache-php:latest*

Step 5 Connect to CCE 2.0.

1. Log in to the CCE 2.0 console. In the navigation pane, click **Image Repository**.
2. On the page displayed, click **Upload Through Docker Client**. Click **Generate a temporary Docker login command**.
3. Click ❐ to copy the command.

Step 6 Log in to the server where Docker is installed. Run the Docker login command copied in **Step 5**.

After you log in to the Docker client, the system displays the message "login succeeded".

Step 7 Run the following command to tag the nginx:1.10 image:

docker tag *{imageid} {image address}:tag*

{image address} is the image repository address. If you push an image from an external network, enter the external image address. If you push an image from an internal network, enter the internal image address.

Example:

docker tag *2e233ad9329b 10.125.1.15:20202/test/apache-php:latest*

Step 8 Run the following command to push the image to the image repository:

docker push *{image address}:tag*

Example:

docker push *10.125.1.15:20202/test/apache-php:latest*

If the following information is displayed, the push is successful:

```
6d6b9812c8ae: Pushed
695da0025de6: Pushed
fe4c16cbf7a4: Pushed
1.10: digest: sha256:eb7e3bbd8e3040efa71d9c2cacfa12a8e39c6b2ccd15eac12bdc49e0b66cee63 size: 948
```

On the **SWR** console, choose **My Images**. The pushed images are displayed in the image list.

----End

20.3 Migrating Clusters

Create VM clusters on the CCE 2.0 console. These new VM clusters should have the same specifications with those created on CCE 1.0.

To create clusters using APIs, see *Cloud Container Engine API Reference 2.0*.

Procedure

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Clusters**. Click **Create VM Cluster**.

Step 2 Set cluster parameters. Parameters with * are mandatory.

Table 20-2 Parameters for creating a cluster

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
* Cluster Name	Name	Name of the cluster to be created.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
*Version	This parameter does not exist in CCE 1.0. Retain the default value.	Cluster version, namely, corresponding Kubernetes baseline version.
*Management Scale	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	Maximum number of nodes that can be managed by the cluster.
* High Availability	Cluster type	● **Yes**: The cluster has three master nodes. The cluster is still available even when two master nodes are down. ● **No**: The cluster has only one master node. If the master node is down, the whole cluster becomes unavailable, but existing applications are not affected.
* VPC	VPCs created in CCE 1.0 can be used in CCE 2.0.	VPC where the cluster will be located. If no VPCs are available, click **Create a VPC**.
* Subnet	Subnets created in CCE 1.0 can be used in CCE 2.0.	Subnet in which the cluster will run.
*Network Model	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	● Container tunnel network: indicates a network built on top of a VPC network and can be applied to common scenarios. ● VPC network: delivers better performance and applies to high-performance and intensive interaction scenarios. Only one cluster using the VPC network model can be created in a single VPC.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Container Network Segment	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	An IP address range that can be allocated to container pods. ● If **Automatically select** is deselected, enter a CIDR block manually. If the CIDR block you specify conflicts with a subnet CIDR block, the system prompts you to select another CIDR block. The recommended CIDR blocks are 10.0.0.0/12-19, 172.16.0.0/16-19, and 192.168.0.0/16-19. If different clusters share a container CIDR block, an IP address conflict will occur and access to the applications in the clusters may fail. ● If **Automatically select** is selected, the system automatically assigns a CIDR block that does not conflict with any subnet CIDR block. The mask of the container CIDR block must be appropriate. It determines the number of available nodes in a cluster. A too small mask value will cause the cluster to soon fall short of nodes. After the mask is set, the estimated maximum number of nodes supported by the current CIDR block will be displayed.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Service Network Segment	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	This parameter is left unspecified, by default. This parameter applies only to clusters of v1.11.7 and later versions. This parameter indicates a CIDR block of Kubernetes services. The mask of the service CIDR block must be appropriate. It determines the number of available nodes in a cluster.
Open EIP	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	An independent public IP address that is reachable from public networks. Select an EIP that has not been bound to any node. A cluster's EIP is preset in the cluster's certificate. Do no delete the EIP after the cluster has been created. Otherwise, two-way authentication will fail. ● **Do not configure**: The cluster's master node will not have an EIP. ● **Configure now**: If no EIP is available for selection, create a new EIP.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Authorization Mode	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	By default, **RBAC** is selected. Read **CCE Role Management Instructions** and select **I am aware of the above limitations and read the CCE Role Management Instructions**. After RBAC is enabled, users access resources in the cluster according to fine-grained permissions policies.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Authentication Mode	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	Permission control on resources in a cluster. For example, you can configure user A to read and write application data in a namespace, while user B to only read resource data in a cluster. For details about role-based permission control, see **3.7- Cluster Management Permission Control**. ● By default, X.509 authentication instead of **Enhanced authentication** is enabled. X.509 is a standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols. ● If permission control on a cluster is required, select **Enhanced authentication** and then **Authenticating Proxy**. Click **Upload** next to **CA Root Certificate** to upload a valid certificate. Select the check box to confirm that the uploaded certificate is valid. If the certificate is invalid, the cluster cannot be created. The uploaded certificate file must be smaller than 1 MB and in .crt or .cer format.
Cluster Description	Description	Description of the cluster.

Step 3 After the configuration is complete, click **Next** to add a node.

Step 4 Continue to add a node.

Step 5 Set the parameters based on **Table 20-3**.

Table 20-3 Parameters for adding a node

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Scope		
Current Region	Scope	Physical location of the node.
AZ.		Physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected through an internal network.
Specifications		
Node Name	Specifications	Name of the node.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Node specifications.		● **General-purpose**: provides general computing, storage, and network configurations for the majority of application scenarios. General-purpose instances can be used in web servers, development and test environments, and small database applications. ● **Memory-optimized**: provides higher memory capacity than general-purpose nodes and is suitable for relational databases, NoSQL, and other workloads that are both memory-intensive and data-intensive. ● **General computing-plus**: provides stable performance and exclusive resources to enterprise-class workloads with high and stable computing performance. ● **GPU-accelerated**: provides powerful floating-point computing and is suitable for real-time, highly concurrent massive computing. Calculation scenario. Graphical processing units (GPUs) of P series are suitable for deep learning, scientific computing, and CAE. GPUs of G series are suitable for 3D animation rendering and CAD. Currently, only clusters of v1.11 support GPU-accelerated nodes. If the cluster version is v1.13 or later, **GPU-accelerated** is not displayed on the page. ● **Ultra-high I/O**: provides ultra-low SSD access latency and ultra-high IOPS performance. This type of specifications is suitable for high-performance relational databases, NoSQL databases (such as Cassandra and MongoDB), and Elasticsearch.
OS		Select an operating system for the node pool. Reinstalling OSs or modifying OS configurations could make nodes unavailable. Exercise caution when performing these operations. For details, see **Risky Operations on Cluster Nodes**.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Virtual Private Cloud	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	The node pool inherits VPC settings from the cluster to which it belongs. This parameter is supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0.
Subnet	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	A subnet provides dedicated network resources that are logically isolated from other networks for network security. You can select any subnet in the cluster VPC. Cluster nodes can belong to different subnets. This parameter is supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0.
Nodes	Quantity	Number of nodes to be created.
Network **NOTE** If the nodes to be created require Internet access, select **Automatically assign** or **Use existing** for **EIP**. If an EIP is not bound to a node, applications running on the node cannot be accessed by the external network.		
EIP	EIP	A public IP address that is reachable from public networks. ● **Do Not Use**: A node without an EIP cannot access the Internet. It can be used only as a cloud server for deploying services or clusters on a private network. ● **Automatically assign**: An EIP with exclusive bandwidth is automatically assigned to each ECS. When creating an ECS, ensure that the EIP quota is sufficient. Set the specifications, required quantity, billing mode, and bandwidth as required. ● **Use Existing EIP**: An existing EIP is assigned to the node.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Disks	Storage	Disk type, which can be **System Disk** or **Data Disk**. ● The system disk capacity ranges from 40 to 1024 GB. The default value is 40 GB. ● The data disk capacity ranges from 100 to 32678 GB. The default value is 100 GB. Data disks deliver three levels of I/O performance: ● **Common I/O**: SATA drives are used to store data. EVS disks of this level provide reliable block storage and a maximum IOPS of 1,000 per disk. They are suitable for key applications. ● **High I/O**: SAS drives are used to store data. EVS disks of this level provide a maximum IOPS of 3,000 and a minimum read/write latency of 1 ms. They are suitable for RDS, NoSQL, data warehouse, and file system applications. ● **Ultra-high I/O**: SSD drives are used to store data. EVS disks of this level provide a maximum IOPS of 20,000 and a minimum read/write latency of 1 ms. They are suitable for RDS, NoSQL, and data warehouse applications.
Login information		
Key pair	Key pair	A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click **Create a key pair** and create one.
Advanced ECS Settings		
ECS Group	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	Select an existing ECS group, or click **Create ECS Group** to create a new one. After the ECS group is created, click the refresh icon. An ECS group allows you to create ECSs on different hosts, thereby improving service reliability.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Resource tag.		By adding tags to resources, you can classify resources. You can create **predefined tags** in Tag Management Service (TMS). Predefined tags are visible to all service resources that support the tagging function. You can use predefined tags to improve tag creation and migration efficiency. CCE will automatically create the "CCE-Dynamic-Provisioning-Node=*Node ID*" tag. A maximum of 5 tags can be added.
Agency Management		The agency is created by the account administrator on the IAM console. By creating an agency, you can share your cloud server resources with another account, or entrust a more professional person or team to manage your resources. To authorize ECS or BMS to call cloud services, select **Cloud service** as the agency type, click **Select**, and then select **ECS BMS**.
Script required before the installation.		Script commands. Enter 0 to 1000 characters. The script will be executed before Kubernetes software is installed. Note that if the script is incorrect, Kubernetes software may not be installed successfully. The script is usually used to format data disks.
Script required after the installation.		Script commands. Enter 0 to 1000 characters. The script will be executed after Kubernetes software is installed and will not affect the installation. The script is usually used to modify Docker parameters.
Add Data Disk		Click **Add Data Disk** to add a data disk and set the capacity of the data disk. Enter a disk formatting command in the input box of **Pre-installation Script**.
Subnet IP Address		Select **Automatically assign IP address** (recommended) or **Manually assigning IP addresses**.
Advanced Kubernetes Settings		

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Maximum Instances	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	The maximum number of pods that can be created on a node, including the system's default pods. Value range: 16 to 250. This limit prevents the node from being overloaded by managing too many pods.
insecure-registries		Click **Add insecure-registry** and enter a repository address. Add the address of the custom image repository with no valid SSL certificate to the docker startup option to avoid unsuccessful image pulling from the personal image repository. The address is in the format of IP address:Port number (or domain name). Post-installation script and insecure-registries cannot be used together.
Maximum Data Space per Container	This parameter does not exist in CCE 1.0. Set this parameter based on your requirements.	The maximum data space that can be used by a container. Value range: 10 GB to 80 GB. If the value of this field is larger than the data disk space allocated to Docker resources, the latter will override the value specified here. Typically, 90% of the data disk space is allocated to Docker resources. This parameter is supported only in v1.13.10-r0 and later versions of clusters. It is not displayed in versions earlier than v1.13.10-r0.

Step 6 Click **Next** to install cluster add-ons.

System resource add-ons must be installed. Advanced functional add-ons are optional.

You can also install optional add-ons after the cluster is created. To do so, choose **Add-ons** in the navigation pane of the CCE console and select the add-on you will install. For details, see **13 Add-on Management**.

Step 7 Click **Create Now**. Check all the configurations, and click **Submit**.

It takes about 6 to 10 minutes to create a cluster. Information indicating the progress of the creation process will be displayed.

----End

20.4 Migrating Applications

This section describes how to create a Deployment with the same specifications as that in CCE 1.0 on the CCE 2.0 console.

It is advised to delete the applications on CCE 1.0 only after you have successfully created these applications on CCE 2.0.

20.4.1 Applications Created Through API or kubectl

If the application in CCE 1.0 is created using APIs or kubectl, use the same method to create applications of the same specifications on the cluster created in CCE 2.0. For details, see *Cloud Container Engine API Reference 2.0* or **Using kubectl to Perform Operations on Clusters**.

- If you connect to the cluster through kubectl, you are advised to perform the operation in CCE 2.0. While in CCE 1.0, you need to download three certificate files and manually perform multiple steps on the client. In CCE 2.0, you only need to perform the following steps to connect to the cluster.
  a. Download the kubectl CLI configuration file. (All authentication information and cluster addresses are in the same file.)
  b. Install and configure the kubectl CLI tool. (Run the required Linux commands to complete the interconnection.)
     For details, see **Kubectl Usage Guide**.
- If you access the CCE console through APIs, the cluster management APIs need to be updated. The native Kubernetes interface does not change, so you can use Kubernetes APIs in the same way. For details, see *Cloud Container Engine API Reference 2.0*.

20.4.2 Applications Created Through Component Templates

If the application in CCE 1.0 is created using a component template, follow the steps this section to migrate the application to CCE 2.0.

Migration Method

Create workloads in the CCE 2.0 console. Delete applications from in CCE 1.0 only after applications are successfully run in CCE 2.0.

Procedure

Step 1 In the navigation pane, choose **Workloads** > **Deployments**. Click **Create Deployment**.

Step 2 Set basic workload parameters as described in **Table 20-4**. The parameters marked with an asterisk (*) are mandatory.

Table 20-4 Basic parameters

Name	Configuration
* Workload Name	Name of a workload, which must be unique.
* Cluster Name	Cluster to which the workload belongs.
* Namespace	Namespace to which the new workload belongs. By default, this parameter is set to **default**.

Name	Configuration
* Instances	Number of instances in the workload. Each workload has at least one instance. You can specify the number of instances as required. Each workload pod consists of the same containers. You can configure multiple pods for a workload to ensure high reliability. For such a workload, if one pod is faulty, the workload can still run properly.
Time Zone Synchronization	If this parameter is enabled, the container and the node use the same time zone. **NOTICE** After time zone synchronization is enabled, disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volume** area. Do not modify or delete the disks.
Description	Description of the workload.

Step 3 Click **Next** to add a container.

1. Click **Add Container** and select the image to be deployed.
2. Set image parameters according to **Table 20-5**.

Table 20-5 Setting image parameters

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Image Name	Container Images	Name of the image. You can click **Change Image** to select another image.
Image Version		Version of the image to be deployed.
Container name.		Container name, which is modifiable.
Privileged Container		Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is granted superuser permissions. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Container Specifications	Memory and CPU	**CPU Quota**: – **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior. **Memory Quota**: – **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For more information about **Request** and **Limit**, see **Setting Container Specifications**. **GPU**: configurable only when the cluster contains GPU nodes. – **GPU**: the percentage of GPU resources reserved for a container. Select **Use** and set the percentage. For example, if this parameter is set to 10%, the container is allowed to use 10% of GPU resources. If you do not select **Use** or set this

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
		parameter to **0**, no GPU resources can be used. – **GPU/Graphics Card**: The workload's pods will be scheduled to the node with the specified GPU. If **Any GPU type** is selected, the container uses a random GPU in the node. If you select a specific GPU, the container uses that GPU accordingly.

3. Set the environment variables, data storage, and user-defined log.

Table 20-6 Configuring advanced settings

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Lifecycle	This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter.	Set the commands for starting and running containers. – **Start Command**: executed when the workload is started. For details, see **Setting Container Startup Commands**. – **Post-Start**: executed after a container runs successfully. For details, see **Setting Container Lifecycle Parameters**. – **Pre-Stop**: executed to delete logs or temporary files before a container ends. For details, see **Setting Container Lifecycle Parameters**.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Health check	This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter.	Configure the health check function to check whether containers and services are running properly. Two types of probes are provided: liveness probes and readiness probes. For details, see **Setting Health Check for a Container**. – **Liveness Probe**: used to restart the unhealthy container. – **Readiness Probe**: used to change the container to the unready state when detecting that the container is unhealthy. In this way, service traffic will not be directed to the container.
Adding environment variables	Adding environment variables	In the **Environment Variables** area, click **Add Environment Variable**. Currently, environment variables can be added using any of the following methods: – **Added manually**: Set **Variable Name** and **Variable Value**. – **Added from Secret**: Set **Variable Name** and select the desired secret name and data. A secret must be created in advance. For details, see **Creating a secret**. – **Added from ConfigMap**: Set **Variable Name** and select the desired ConfigMap name and data. A ConfigMap must be created in advance. For details, see **Creating a ConfigMap**.

Parameter in CCE 2.0	Parameter in CCE 1.0	Configuration
Data Storage	Volume	For the applications using the old component template, perform the following operations: 1. Choose **Data Storage** > **Local Volume**. Click **Add Local Volume**. 2. Select **HostPath**. 3. Set the following parameters: ■ **Host Path**: Path of the host to which the local volume is to be mounted, corresponding to **/tmp** of volumes. ■ Click **Add Container Path**, enter the container path to which the data volume is mounted. It corresponds to **/test** of **Volumes**. ■ **Permission**: **Read/Write**. 4. When the configuration is complete, click **OK**.
Security Settings	This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter.	Set container permissions to protect the system and other containers from being affected. Enter the user ID to set container permissions and prevent systems and other containers from being affected.
Container Logs	This parameter does not exist in CCE 1.0. For migrated applications, you do not need to set this parameter.	Set a policy and log directory for collecting application logs and preventing logs from being over-sized. For details, see **Collecting Container Logs from Specified Paths**.

Step 4 Click **Next**. Click **Add Service** and set the workload access type.

If your workload will be reachable to other workloads or public networks, add a Service to define the workload access type.

The workload access type determines the network attributes of the workload. Workloads with different access types can provide different network capabilities. For details, see **Network Management**.

- **Access Type**: Select **LoadBalancer (ELB)**.
- **Service Name**: Specify a service name. You can use the workload name as the service name.
- **Service Affinity:**
  - **Cluster level**: External traffic is routed to all nodes in the cluster while masking clients' source IP addresses.
  - **Node level**: External traffic is routed to the node where the load balancer used by the service is located, without masking clients' source IP addresses.
- **Elastic Load Balancer**: A load balancer automatically distributes Internet access traffic to multiple nodes running the workload. 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).
  - **Public network**: You can select an existing public network load balancer or have the system automatically create a new public network load balancer.

    If you have the system automatically create a public network load balancer, you can click **Change Configuration** to modify its name, EIP type, billing mode, and bandwidth.
  - **Private network**: You can select an existing private network load balancer or have the system automatically create a new private network load balancer.
- **Port Settings**
  - **Protocol**: protocol used by the Service.
  - **Container Port**: a port that is defined in the container image and on which the workload listens. The Nginx application listens on port 80.
  - **Access Port**: a port mapped to the container port at the load balancer's IP address. The workload can be accessed at <load balancer's IP address>:<access port>. The port number range is 1–65535.

Step 5 Click **OK**, and then click **Next**. Skip advanced settings.

Step 6 Click **Create** after the configuration is complete. Click **Back to Deployment List**.

If the deployment is in the **Running** state, the deployment is successfully created.

Workload status is not updated in real time. Click in the upper right corner or press **F5** to refresh the page.

Step 7 To access the workload in a browser, copy the workload's **External Access Address** and paste it into the address box in the browser.

NOTE

External access addresses are available only if the Deployment access type is set to **NodePort** and an EIP is assigned to any node in the cluster, or if the Deployment access type is set to **LoadBalancer (ELB)**.

----End

20.4.3 Applications Created Through App Designer

If the application in CCE 1.0 version is created through App Designer, follow the steps in this section to migrate the application to CCE 2.0. In CCE 2.0, use the custom Helm chart to create applications. For more information, see **Chart Management**.

Procedure

Step 1 Log in to the console of CCE 1.0. In the navigation pane, click **App Manager**. Then, click the name of the application to be migrated to go to the details page. Obtain **Component Name** and **Service Name**.

Step 2 Connect to the cluster. For details, see **Kubectl Usage Guide**.

Step 3 Run the following commands on the kubectl client to obtain the YAML files related to the application.

kubectl get rc *new-appcomponent-1-11d3* **-o yaml >** *new-appcomponent-1-11d3***.yaml**

kubectl get svc *new-port-3* **-o yaml >** *new-port-3***.yaml**

In the preceding commands, replace *new-appcomponent-1-11d3* and *new-port-3* with the values obtained in **Step 1**.

Step 4 Edit the new-appcomponent-1-11d3.yaml file by deleting the lines in bold and modifying the lines in Italic.

```
apiVersion: v1
kind: ReplicationController
metadata:
  annotations:
    cce/app-createTimestamp: 2018-04-19-11-39-27
    cce/app-description: ""
    cce/app-updateTimestamp: 2018-04-19-11-39-27
  creationTimestamp: 2018-04-19T11:37:17Z
  generation: 1
  labels:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
    rollingupdate: "false"
  name: new-appcomponent-1-11d3
  namespace: default
  resourceVersion: "8325781"
  selfLink: /api/v1/namespaces/default/replicationcontrollers/new-appcomponent-1-11d3
  uid: 039ada96-43c6-11e8-8f34-fa163e738aa3
spec:
  replicas: 1
  selector:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
    rollingupdate: "false"
  template:
    metadata:
      annotations:
        scheduler.alpha.kubernetes.io/affinity: '{"nodeAffinity":
{"requiredDuringSchedulingIgnoredDuringExecution":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"failure-domain.beta.kubernetes.io/zone","operator":"NotIn","values":[""]}]}]}}}'
      creationTimestamp: null
      labels:
        cce/appgroup: app-design
        name: new-appcomponent-1-11d3
        rollingupdate: "false"
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: failure-domain.beta.kubernetes.io/zone
                operator: NotIn
                values:
                - ""
      containers:
      - image: 10.125.1.108:6443/test/apache-php:latest -->Change it to the image address of the later
CCE version
        imagePullPolicy: Always
        name: new-container-2-11d3f16
        ports:
        - containerPort: 3333
          protocol: TCP
        resources: {}
        securityContext:
          privileged: true
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      imagePullSecrets:
      - name: myregistry    -----> Change it to default-secret
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

Step 5 Open the **new-port-3.yaml** file, and delete the lines in bold.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: 2018-04-19T11:37:17Z
  labels:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
  name: new-port-3
  namespace: default
  resourceVersion: "8325766"
  selfLink: /api/v1/namespaces/default/services/new-port-3
  uid: 039c918f-43c6-11e8-8f34-fa163e738aa3
spec:
  clusterIP: 10.247.145.136
  externalTrafficPolicy: Cluster
  ports:
  - name: new-port-30
    nodePort: 31471
    port: 3333
    protocol: TCP
    targetPort: 3333
  selector:
    cce/appgroup: app-design
    name: new-appcomponent-1-11d3
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Step 6 Connect the kubectl client to the cluster in CCE 2.0. For details, see **Kubectl Usage Guide**.

Step 7 Run the following commands to create the application again:

kubectl create -f *new-appcomponent-1-11d3***.yaml**

kubectl create -f *new-port-3***.yaml**

----End