Multi-Cloud Container Platform

User Guide

Issue 01

Date 2023-08-03

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Contents

1 Permission Management

Creating a User and Assigning Permissions

This chapter describes how to use **Identity and Access Management (IAM)** to implement fine-grained permissions control for your MCP resources. 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 for accessing MCP resources.
- Assign only the permissions required for users to perform a specific task.
- Entrust an account or cloud service to perform efficient O&M on your MCP resources.

If your Huawei Cloud account does not require individual IAM users, skip this chapter.

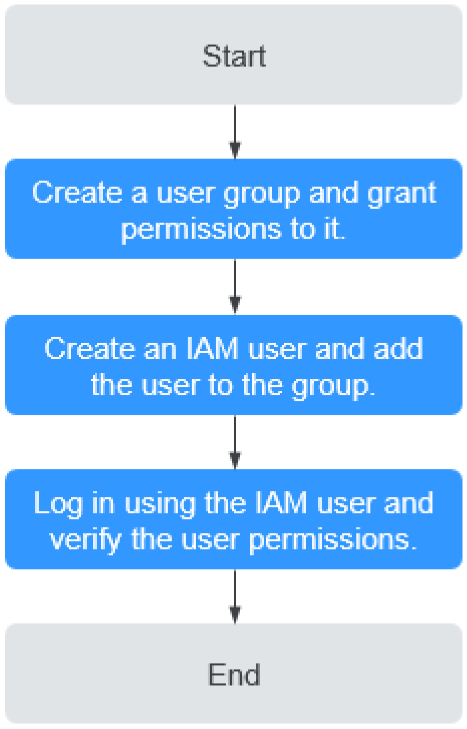
Figure 1-1 shows the procedure for assigning permissions by user group.

Prerequisites

- Before creating an IAM user to use MCP, **authorize access for MCP** in the current region using your Huawei Cloud account.
- Learn about permissions available to an IAM user to use MCP and choose permissions according to your requirements. For the permissions of other services, see **System Permissions**.

Process Flow

Figure 1-1 Process for assigning permissions to an IAM user


Start
Create a user group and grant permissions to it.
Create an IAM user and add the user to the group.
Log in using the IAM user and verify the user permissions.
End

1. **Create a user group and assign permissions**.

   Create a user group on the IAM console, and assign the **VPC Administrator** and **CCE Administrator** permissions to the group.

2. **Create a user and add the user to the user group**.

   Create a user on the IAM console and add the user to the group created in **1**.

3. **Log in** and verify permissions.

   Log in to the management console as the user you created, and verify that the user has the assigned permissions.

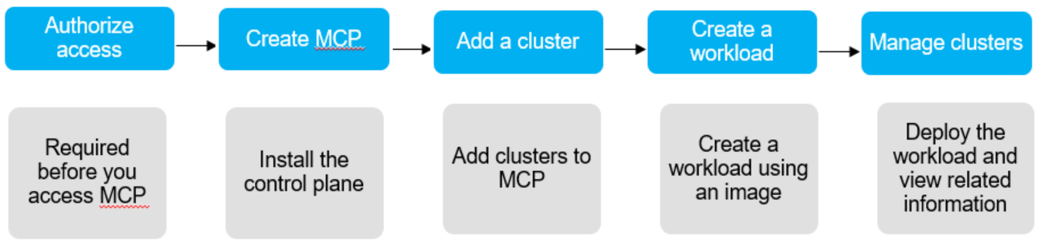
   – Choose **Service List** > **Multi-Cloud Container Platform** to enter the MCP console, and click **Buy MCP**. If you can buy MCP successfully, the permissions have taken effect.

2 Before You Start

Process for Using MCP

Before using MCP, get familiar with the MCP process, as shown in **Figure 2-1**. **To experience MCP, go to the MCP console**.

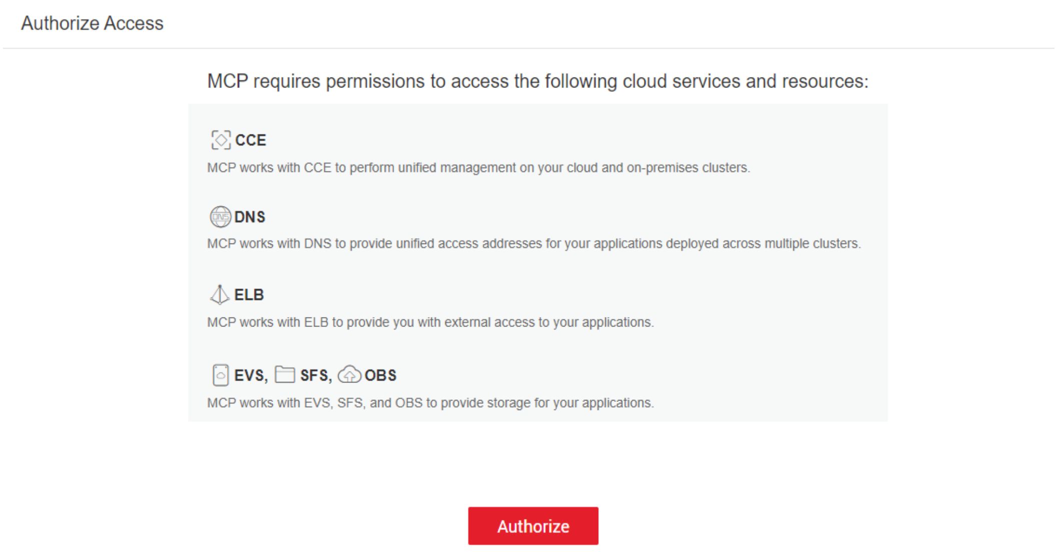
Figure 2-1 Getting started with MCP


Authorize access
Create MCP
Add a cluster
Create a workload
Manage clusters
Required before you access MCP
Install the control plane
Add clusters to MCP
Create a workload using an image
Deploy the workload and view related information

Step 1 Authorize access for MCP.

When you enable the MCP service, authorize MCP to access related cloud services and resources. If you have not authorized MCP before accessing the MCP homepage, the following information will be displayed. Click **Authorize**.

Figure 2-2 Service authorization page


Authorize Access

MCP requires permissions to access the following cloud services and resources:

CCE
MCP works with CCE to perform unified management on your cloud and on-premises clusters.

DNS
MCP works with DNS to provide unified access addresses for your applications deployed across multiple clusters.

ELB
MCP works with ELB to provide you with external access to your applications.

EVS, SFS, OBS
MCP works with EVS, SFS, and OBS to provide storage for your applications.

Step 2 Create an MCP based on the function description.

- **Creating an MCP**

Step 3 Add a cluster to MCP.

- **Adding a CCE Cluster to MCP**
- **Adding a Cluster of Another Public Cloud to MCP**
- **Adding an On-Premises Kubernetes Cluster to MCP**

Step 4 Create a workload.

You can use images in Huawei Cloud SoftWare Repository for Container (SWR), open source images, or third-party images to create workloads.

Step 5 Deploy workloads and add Services, PVCs, and namespaces.

----End

3 Creating an MCP Instance

MCP is a hybrid cloud container platform provided by Huawei Cloud based on years of experience in the cloud container field and advanced cluster technology of the Kubernetes community.

Creating an MCP Instance

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

Step 2 Click **Create MCP** in the upper right corner.

Step 3 Set parameters.

- **Region**

  The current region is selected by default. You can change it to another available region.

  NOTE

  For low network latency and quick resource access, select the nearest region.

- **MCP Name**

  Name of the MCP to be created, which must be unique.

- **Hub VPC**

  Select a virtual private cloud (VPC) to which the MCP instance is to be bound. After the VPC is bound, you can use **Direct Connect** to connect Kubernetes clusters in a private cloud or other public clouds to the hub VPC, or use VPC **peering connections** to connect the MCP instance to CCE clusters in the same region. For details, see **Network Description**.

  NOTICE

  MCP CIDR block, hub VPC CIDR block, and Kubernetes cluster CIDR block must not conflict with each other. Otherwise, the cluster cannot be added.

- **MCP CIDR Block**

  CIDR block internally used by MCP. If the cluster to be added accesses MCP through a public network, use the default CIDR block. If you need to connect the cluster to the MCP network through Direct Connect, set this CIDR block to avoid conflicts with the CIDR block used in the cluster.

- The subnet mask of the MCP CIDR block is fixed at 22 bits. 10.253.0.0/22 and 172.31.0.0/22 are recommended.
- The MCP CIDR block must not conflict with the following CIDR blocks:
  - CIDR block used by the hub VPC
  - CIDR block used by the cluster that is accessed in non-public network mode, including the container CIDR block, ServiceIP CIDR block, and cluster node CIDR block.
- You are advised to select a subnet with the mask 22 from the ranges of 10.2.0.0/16 to 10.246.0.0/16, 10.248.0.0/16 to 10.255.0.0/16, and 172.17.0.0/16 to 172.31.0.0/16.
- Do not use a subnet with a 22-bit mask within the range of 192.168.0.0/16, 172.16.0.0/16, 10.0.0.0/16, 10.1.0.0/16, or 10.247.0.0/16. (The subnet may conflict with the CIDR block in your cluster environment.)

NOTICE

This parameter cannot be modified after MCP is created. Exercise caution when setting this parameter.

Step 4 Click **Create**. The **Details** page is displayed.

Click **Submit**. It takes about 15 minutes to create an MCP.

Step 5 After MCP is created, click **MCP Console** to access the console.

----End

4 Deployments

The configured multi-cluster Deployments can be automatically deployed on specified Kubernetes clusters.

Currently, MCP provides the following two methods for creating workloads:

Using the CCE Console

Using kubectl

Using the CCE Console

Step 1 Log in to the MCP console. In the navigation pane, choose **Workloads** > **Deployments**. On the page displayed, click **Create Deployment**.

Step 2 Configure basic information about the workload.

- **Workload Name**: Name of a workload, which must be unique.
- **Namespace**: Namespace to which the workload belongs.
- **Cluster**: Select the cluster where the workload is to be deployed. The number of clusters depends on your service requirements.
- **Pod Deployment Mode**: Two modes **Common** and **Weight** are supported.
- **Pods**: This parameter is valid only when **Pod Deployment Mode** is set to **Common**. It specifies the number of pods in each cluster of the multi-cluster Deployment. The default value is **2**. Each workload pod consists of the same containers. On MCP, you can set an auto scaling policy to dynamically adjust the number of workload pods based on the workload resource usage.
- **Weight Ratio**: This parameter is valid only when **Pod Deployment Mode** is set to **Weight**. This parameter specifies the total number of pods in the multi-cluster Deployment, and weight and pod range for each cluster in the multi-cluster Deployment.

Step 3 Click **Next**.

1. Click **Add Container** and select the image to be deployed.

NOTE

If you select multiple cross-region clusters to run the Deployment, you can configure automatic image synchronization for the image to be selected. This function can reduce image upload and update operations and accelerate the speed for applications managed by MCP to pull images across regions. For details, see **Configuring Automatic Image Synchronization Between Regions**.

- **My Images**: Create a workload using an image in the HUAWEI CLOUD image repository. If no image is available, click **push an image** to push an image.
  - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
  - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
- **Open Source Images**: Create a workload using an official image in the open source image repository.
  - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
  - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
- **Third-Party Images**: Create a workload using an image from any third-party image repository (image repositories other than Huawei Cloud SWR and open source image repository). Ensure that the node where the workload is running is accessible from public networks.
  - 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 authentication (using account and password), select the corresponding secret, enter an image address, and then click **OK**.

    NOTE

    This secret is a general configuration. To configure the secret of a specific cluster, select **Customized Configuration** next to **Container Settings** in the **Customize Cluster Configurations** step. In the **Basic Information** area, click **Change Image** and select the desired secret.

If your workload is a multi-container workload, click **Add Container** to add more containers.

2. Configure basic container information.
   - **Image Name**: name of the image. You can click **Change Image** to select another image.
   - **Image Tag**: Select the tag of the image to be used to deploy an application.

- **Container Name**: container name, which can be changed.
- **Container Specifications**: resource amount that users can apply for and the maximum amount of resources that are available to users. For details, see **Setting Container Specifications**.
  - **Request**: minimum amount of resources required for running a container.
  - **Limit**: If a container is overloaded, the system may be faulty. To avoid this situation, set the maximum limits for the container resource quotas to ensure that container resources do not exceed the limits.

3. Configure container lifecycle, that is, set the commands that need to be executed in each phase during container management.
   - **Startup Command**: executed when the container is started.
   - **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 is stopped. For details, see **Setting Container Lifecycle Parameters**.
4. Set the health check function that checks 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**: The probe restarts the workload when detecting that the workload pod is unhealthy.
   - **Readiness Probe**: The probe sets the workload to the unready state when detecting that the workload pod is unhealthy. In this way, the service traffic will not be directed to the workload pod.
5. Set environment variables.

   Environment variables can be added to a container. In general, environment variables are used to set parameters.

   In the **Environment Variables** area, click **Add Environment Variable**.
6. Specify security settings.

   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.
7. Configure data storage.

   Mounting data storage to containers applies to scenarios where persistent storage and high disk I/O are required. For details, see **Federated PVCs**.

Step 4 Click **Next** and create a Service.

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**.

- **ClusterIP**: A workload can be accessed from other workloads in the same cluster. For details, see **ClusterIP**.

- **NodePort**: A workload can be accessed from any node in the cluster. For details, see **NodePort**.

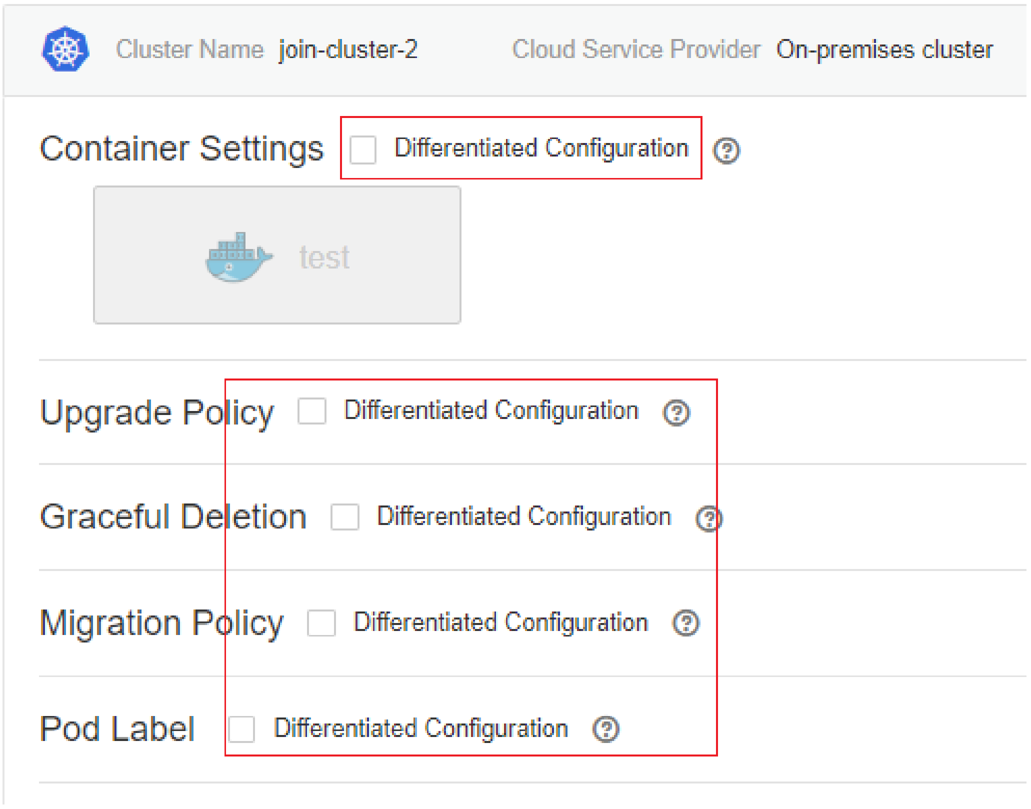
Step 5 Click **Next** and configure advanced settings.

- **Upgrade Policy**
  - **Rolling upgrade**: An old pod is gradually replaced with a new pod. During the upgrade, service traffic is evenly distributed to the 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**: Old pods are deleted before new pods are created. Services will be interrupted during an in-place upgrade.
- **Graceful Deletion**: Set a time window for **pre-stop commands** to finish execution before a workload is forcibly deleted. If workload processes are not terminated after the time window elapses, the workload will be forcibly deleted.
- **Migration Policy**: Set a time window for the system to schedule the workload pod to another available node when the node where the workload pod is running is unavailable. The default value is 300s.
- **Pod Label**: System-defined app labels added during workload creation cannot be modified. You can click **Add Label** to add labels.

Step 6 Click **Next** to customize configurations for the selected cluster. Select **Customized Configuration** to customize the parameters of only this cluster. For details about each configuration option, see **Step 3**. For details about other configurations, see **Step 5**.

Figure 4-1 Customized configurations


Cluster Name join-cluster-2
Cloud Service Provider On-premises cluster

Container Settings ☐ Differentiated Configuration

Upgrade Policy ☐ Differentiated Configuration

Graceful Deletion ☐ Differentiated Configuration

Migration Policy ☐ Differentiated Configuration

Pod Label ☐ Differentiated Configuration

Step 7 Click **OK**.

----End

Using kubectl

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

1. Use kubectl to connect to the cluster. For details, see **Accessing MCP by Using kubectl**.

2. Create and edit the **nginx-deployment.yaml** file. **nginx-deployment.yaml** is an example file name. You can rename it as required.

vi nginx-deployment.yaml

Insert the following information in the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
```

```
metadata:
  labels:
    app: nginx
spec:
  containers:
  - image: nginx    # If you use an image in My Images, obtain the image path from SWR.
    imagePullPolicy: Always
    name: nginx
  imagePullSecrets:
  - name: default-secret
```

NOTE

The foregoing is an example. For more information about Deployments, see **Kubernetes documentation**.

For details about these parameters, see **Table 4-1**.

Table 4-1

Parameter	Description	Mandatory/Optional
apiVersion	API version. Set this parameter based on the version of the current cluster. ● For clusters of v1.17 or later, the apiVersion format of Deployments is **apps/v1**. ● For clusters of v1.15 or earlier, the apiVersion format of Deployments is **extensions/v1beta1**.	Mandatory
kind	Type of the object to be created.	Mandatory
metadata	Metadata definition of the 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. Defaults to **RollingUpdate**. ● RollingUpdate ● ReplaceUpdate	Optional
template	Detailed description of a created container pod.	Mandatory

Parameter	Description	Mandatory/Optional
metadata	Metadata.	Mandatory
labels	**metadata.labels**: Container labels.	Optional
spec: containers	• **image** (mandatory): Container image name. • **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. • To pull an image from 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.	If a private image is used, this parameter is mandatory.

3. Create a Deployment.

kubectl create -f nginx-deployment.yaml

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

```
deployment "nginx" created
```

4. Query the Deployment status.

kubectl get deployment

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

```
NAME          READY    UP-TO-DATE   AVAILABLE   AGE
nginx        1/1       1            1          4m5s
```

Parameter description

o **NAME**: pod name

o **READY**: number of pod replicas that have been deployed

o **STATUS**: status of the Deployment

o **RESTARTS**: restart times

o **AGE**: period the Deployment keeps running

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

5 Resource Management

5.1 Cluster Management

5.1.1 Typical Scenarios and Networking

Typical Scenarios

MCP can manage clusters of Huawei Cloud, other public clouds, and private clouds. It applies to the following scenarios:

- Hybrid-cloud scenario

  In this scenario, you can add both Huawei Cloud CCE clusters and on-premises Kubernetes clusters of private clouds to MCP.

  For details, see **Adding a CCE Cluster to MCP** and **Adding an On-Premises Kubernetes Cluster to MCP**.

- Multi-cloud scenario

  In this scenario, you can add both Huawei Cloud CCE clusters and clusters of other public cloud providers, such as Alibaba Cloud and Tencent Cloud, to MCP.

  For details, see **Adding a CCE Cluster to MCP** and **Adding a Cluster of Another Public Cloud to MCP**.

- Multi-region scenario

  In this scenario, you can add Huawei Cloud CCE clusters in different VPCs and regions to MCP.

  For details, see **Adding a CCE Cluster to MCP**.

Network Description

Before adding Kubernetes clusters to MCP, pay attention to network connections between MCP and each cluster. If both MCP and Kubernetes clusters have public IP addresses, ensure that they can communicate with each other using public IP addresses.

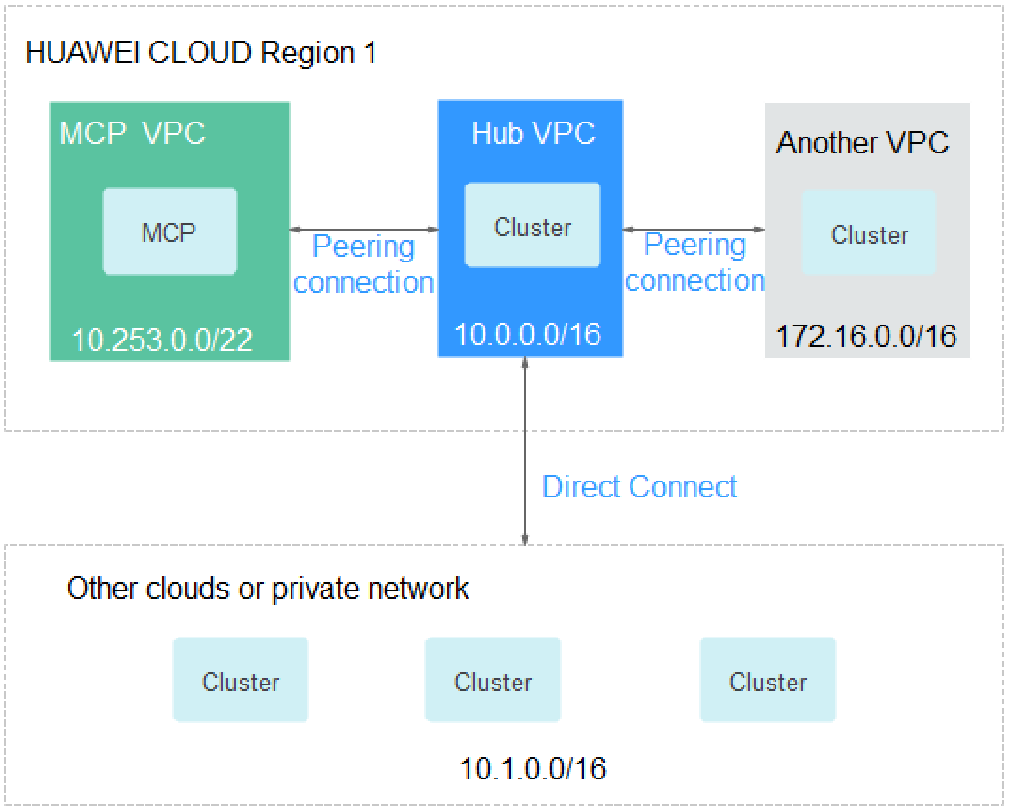
MCP can also manage Kubernetes clusters even though no public IP address is available. When creating an MCP instance, you can bind it to a hub VPC, which has a **peering connection** with the VPC where MCP resides. With the hub VPC, MCP can connect to a Kubernetes cluster in any of the following ways:

- If the cluster is in the hub VPC and MCP is bound to the hub VPC, the cluster and MCP can communicate with each other.
- If the cluster is in another VPC in the same region as MCP, you can create a **peering connection** between the VPC where the cluster resides and the hub VPC so that MCP and the cluster can communicate with each other.
- If the cluster is in another public cloud or private network, you can use **Direct Connect** to connect the physical network to the hub VPC so that MCP and the cluster can communicate with each other.

NOTICE

MCP CIDR block, hub VPC CIDR block, and Kubernetes cluster CIDR block must not conflict with each other when a hub VPC is used. Otherwise, the cluster cannot be added.

Figure 5-1 Connecting to MCP through a hub VPC



5.1.2 Adding a CCE Cluster to MCP

Prerequisites

- The CCE cluster is v1.13 or later, and the cluster is in the normal state.
- The cluster can communicate with the MCP network.

Constraints

The CCE cluster to be added must have not been added to an MCP instance. That is, a CCE cluster cannot be added to two or more MCP instances.

CAUTION

MCP cannot distinguish the same cluster using different access modes or different IP addresses to join an MCP instance. For example, if you have added a cluster to an MCP instance through public network and you want to add the cluster again through hub VPC, the cluster can be added but the status will be abnormal. Similar is the case when you have added a cluster to an MCP instance through a public IP address and want to add it again using another public IP address. Therefore, you are not advised to add the same cluster to MCP.

Procedure

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Cluster Management**. On the page displayed, click **Create Cluster**.

Step 2 On the **Create Cluster** page displayed, click **Huawei Cloud** under **Public Clouds** and click **Next** in the **Select Cloud Service Provider** step.

Step 3 Click **Next** and set parameters.

- **Select Region**

  The current region is used by default, which can be changed based on your service requirements.

- **Select Cluster**

  Select an available CCE cluster.

- **Set Cluster Name**

  Set a cluster name to be used in MCP. The name must be globally unique.

- **Select Access Mode**

  Two options are available:

  – **Public network access**: MCP accesses the CCE cluster by using an EIP. Ensure that the CCE cluster has an EIP.

  – **VPC peering connection**: MCP accesses the CCE cluster in the same region through the hub VPC. Currently, the hub VPC can communicate with the VPC where the CCE cluster resides through a peering connection. For details about the hub VPC, see **Network Description**.

NOTE

A VPC peering connection will be automatically created.

Step 4 Click **OK**. The **Verify Cluster Admission** page is displayed. It takes about 1 minute to complete admission verification.

Step 5 After the verification is successful, click **OK**.

It takes about 1 minute to add a cluster. After the cluster is added, you can view the added cluster on the **Cluster Management** page and the cluster status is **Normal**.

----End

5.1.3 Adding a Cluster of Another Public Cloud to MCP

Prerequisites

- A cluster of another public cloud cluster is available. The cluster version is v1.13 or later, and the cluster is in the normal state.
- The cluster can communicate with the MCP network.

Constraints

The cluster to be added must have not been added to an MCP instance. That is, a cluster cannot be added to two or more MCP instances.

CAUTION

MCP cannot distinguish the same cluster using different access modes or different IP addresses to join an MCP instance. For example, if you have added a cluster to an MCP instance through public network and you want to add the cluster again through hub VPC, the cluster can be added but the status will be abnormal. Similar is the case when you have added a cluster to an MCP instance through a public IP address and want to add it again using another public IP address. Therefore, you are not advised to add the same cluster to MCP.

Procedure

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Cluster Management**. On the page displayed, click **Add Cluster**.

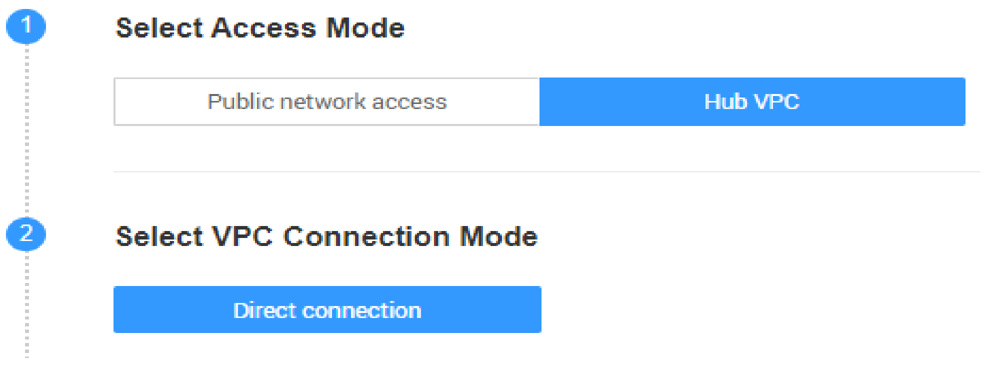
Step 2 On the **Create Cluster** page displayed, select a cloud service provider.

Step 3 Click **Next** and select an access mode.

Two options are available:

- **Public network access**: MCP accesses the cluster by using an EIP. Ensure that the cluster has an EIP.
- **Hub VPC**: MCP connects to the hub VPC through **Direct Connect** to enable mutual access between MCP and clusters. You must **buy a direct connection**. For details about hub VPC connections, see **Network Description**.

Figure 5-2 Selecting an access mode



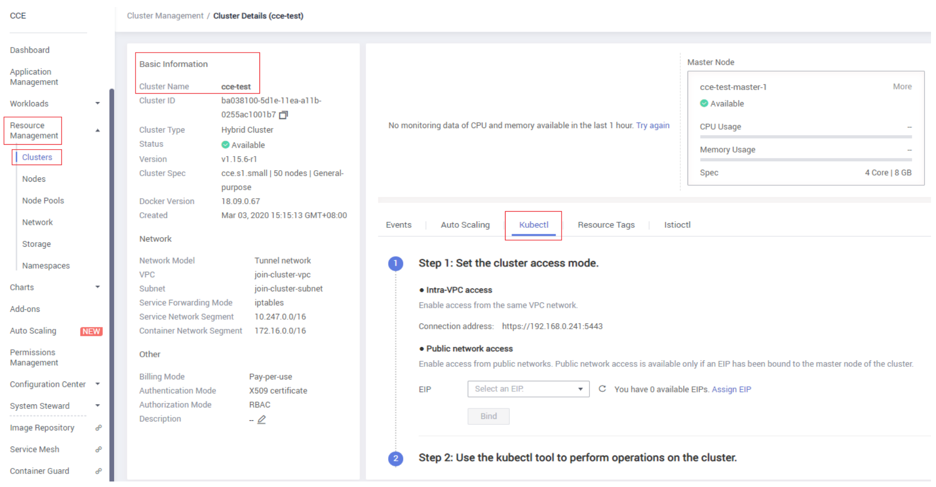
Step 4 Upload the KubeConfig file.

- For other public cloud clusters that do not support direct export of KubeConfig files or need tools for parsing, see **Obtaining KubeConfig** to obtain the KubeConfig file.
- **KUBECONFIG_PATH** is the local path for storing the KubeConfig file. In the Linux OS, it defaults to **$KUBECONFIG** if the environment variables have been set; otherwise, it defaults to **$HOME/.kube/config**.
- **Upload KubeConfig File**: Click **Select File** to upload the obtained KubeConfig file of the cluster. Ensure that the cluster network in the configuration file is connected. The file can be in YAML or JSON format.

NOTE

A KubeConfig file will be generated after a cluster is created. You can download the KubeConfig file from the cluster management page. For example, to download the KubeConfig file of a CCE cluster, log in to the CCE console, choose **Resource Management** > **Clusters** in the navigation pane, click the cluster name, and download the KubeConfig file on the **kubectl** tab page.

Figure 5-3 Downloading the KubeConfig file of a CCE cluster



Basic Information

Cluster Name	cce-test
Cluster ID	ba038100-5d1e-11ea-a11b-0255ac1001b7
Cluster Type	Hybrid Cluster
Status	Available
Version	v1.15.6-r1
Cluster Spec	cce.s1.small | 50 nodes | General-purpose
Docker Version	18.09.0.67
Created	Mar 03, 2020 15:15:13 GMT+08:00

Network

Network Model	Tunnel network
VPC	join-cluster-vpc
Subnet	join-cluster-subnet
Service Forwarding Mode	iptables
Service Network Segment	10.247.0.0/16
Container Network Segment	172.16.0.0/16

Other

Billing Mode	Pay-per-use
Authentication Mode	X509 certificate
Authorization Mode	RBAC
Description	--

No monitoring data of CPU and memory available in the last 1 hour.

Master Node

cce-test-master-1	Available
CPU Usage	--
Memory Usage	--
Spec	4 Core | 8 GB

Step 1: Set the cluster access mode.

● Intra-VPC access

Enable access from the same VPC network.

Connection address: https://192.168.0.241:5443

● Public network access

Enable access from public networks. Public network access is available only if an EIP has been bound to the master node of the cluster.

EIP You have 0 available EIPs.

Step 2: Use the kubectl tool to perform operations on the cluster.

- **Select Context**: Select a context.
- **Set Cluster Name**: Enter the name of the cluster to be added. The name must be globally unique.

Step 5 Click **OK**. The **Verify Cluster Admission** page is displayed.

After the verification is successful, click **OK**.

Step 6 Click **OK**. It takes about 1 minute to add a cluster.

After the cluster is added, you can view the added cluster on the **Cluster Management** page and the cluster status is **Normal**.

----End

5.1.4 Adding an On-Premises Kubernetes Cluster to MCP

Prerequisites

- An on-premises Kubernetes cluster is available. The CCE cluster is v1.13 or later, and the cluster is in the normal state.
- The cluster can communicate with the MCP network.

Constraints

The on-premises Kubernetes cluster to be added must have not been added to an MCP instance. That is, a cluster cannot be added to two or more MCP instances.

⚠ CAUTION

MCP cannot distinguish the same cluster using different access modes or different IP addresses to join an MCP instance. For example, if you have added a cluster to an MCP instance through public network and you want to add the cluster again through hub VPC, the cluster can be added but the status will be abnormal. Similar is the case when you have added a cluster to an MCP instance through a public IP address and want to add it again using another public IP address. Therefore, you are not advised to add the same cluster to MCP.

Procedure

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Cluster Management**. On the page displayed, click **Create Cluster**.

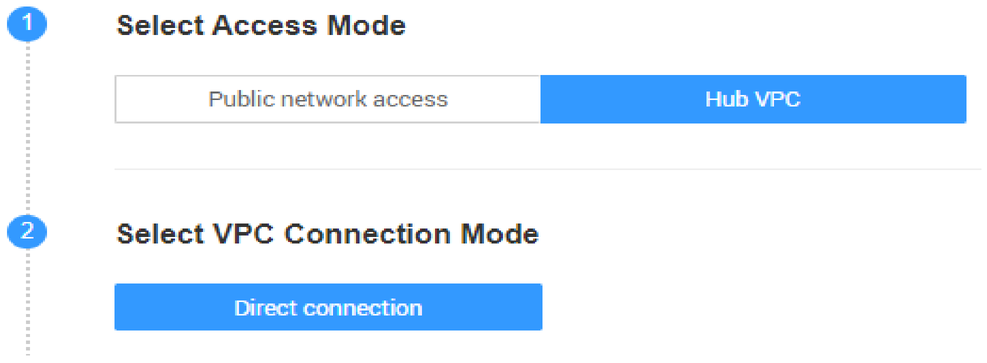
Step 2 On the **Create Cluster** page displayed, click **On-premises cluster** under **Private Clouds** and click **Next** in the **Select Cloud Service Provider** step.

Step 3 Click **Next** and select an access mode.

Two options are available:

- **Public network access**: MCP accesses the cluster by using an EIP. Ensure that the cluster has an EIP.
- **Hub VPC**: MCP connects to the hub VPC through **Direct Connect** to enable mutual access between MCP and clusters. You must **buy a direct connection**. For details about hub VPC connections, see **Network Description**.

Figure 5-4 Selecting an access mode



Step 4 Upload the KubeConfig file.

- A KubeConfig file will be generated after a cluster is created. You can download the KubeConfig file from the cluster management page.

  If the current cluster does not support direct export of the KubeConfig file or needs tools for parsing, see **Obtaining KubeConfig**.
- **KUBECONFIG_PATH** is the local path for storing the KubeConfig file. In the Linux OS, it defaults to **$KUBECONFIG** if the environment variables have been set; otherwise, it defaults to **$HOME/.kube/config**.
- **Upload KubeConfig File**: Click **Select File** to upload the obtained KubeConfig file of the cluster. Ensure that the network in the configuration file is connected to the cluster. The file can be in YAML or JSON format.
  - **Select Context**: Select a context.
  - **Set Cluster Name**: Set a cluster name to be used in MCP. The name must be globally unique.

Step 5 Click **OK**. The **Verify Cluster Admission** page is displayed.

Step 6 After the verification is successful, click **OK** to add a cluster.

It takes about 1 minute to add a cluster. After the cluster is added, you can view the added cluster on the **Cluster Management** page and the cluster status is **Normal**.

----End

5.1.5 Obtaining KubeConfig

If the KubeConfig file of the current cluster cannot be directly exported, perform the following operations to obtain the KubeConfig file that can be used in MCP.

Step 1 (Optional) Configure kubectl.

If kubectl can be properly executed in the cluster, skip this step.

```
which kubectl
```

Obtain the absolute path of the kubectl file **xxx/kubectl**, and the path of the KubeConfig file **yyy/config**.

```
alias kubectl='xxx/kubectl --kubeconfig=yyy/config'
```

Step 2 Create the **my-user-sa.yaml**, **my-user-role.yaml** and **my-user-role-binding.yaml** files.

my-user-role.yaml:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: my-user-role
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - get
```

my-user-role-binding.yaml:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: my-user-role-binding
subjects:
  - kind: ServiceAccount
    name: my-user
    namespace: default
roleRef:
  kind: ClusterRole
  name: my-user-role
  apiGroup: rbac.authorization.k8s.io
```

my-user-sa.yaml:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-user
```

Step 3 Run the following commands on the managed cluster:

```
kubectl apply -f my-user-sa.yaml
kubectl apply -f my-user-role.yaml
kubectl apply -f my-user-role-binding.yaml
```

Step 4 Run the following command to obtain the token:

```
kubectl get secret -n default `kubectl get secret -n default | grep my-user | awk '{print $1}'` -oyaml | grep
token: | awk '{print $2}' | base64 -d
```

Step 5 Configure the KubeConfig file.

In the **kubeconfig.json** file, replace the value of the **server** parameter with that in the **yyy/config** file obtained in **Step 1**, and the value of the **token** parameter with the value obtained in **Step 4**.

kubeconfig.json:

```
kind: Config
apiVersion: v1
preferences: {}
clusters:
  - name: internalCluster
    cluster:
      server: 'https://119.xxx.xxx.xxx:5443'
      insecure-skip-tls-verify: true
users:
  - name: user
    user:
      token: 'MIIFbAYJKo*****'
```

```
contexts:
  - name: internal
    context:
      cluster: internalCluster
      user: user
current-context: internal
```

Step 6 Use the KubeConfig file configured in **Step 5** to add the cluster. After the cluster is added, you can delete the ClusterRole, ClusterRoleBinding, and ServiceAccount created in **Step 3**.

----End

5.1.6 Accessing MCP by Using kubectl

kubectl can be used to configure resources such as Deployments of a specific MCP, as well as manage all MCP resources.

Prerequisites

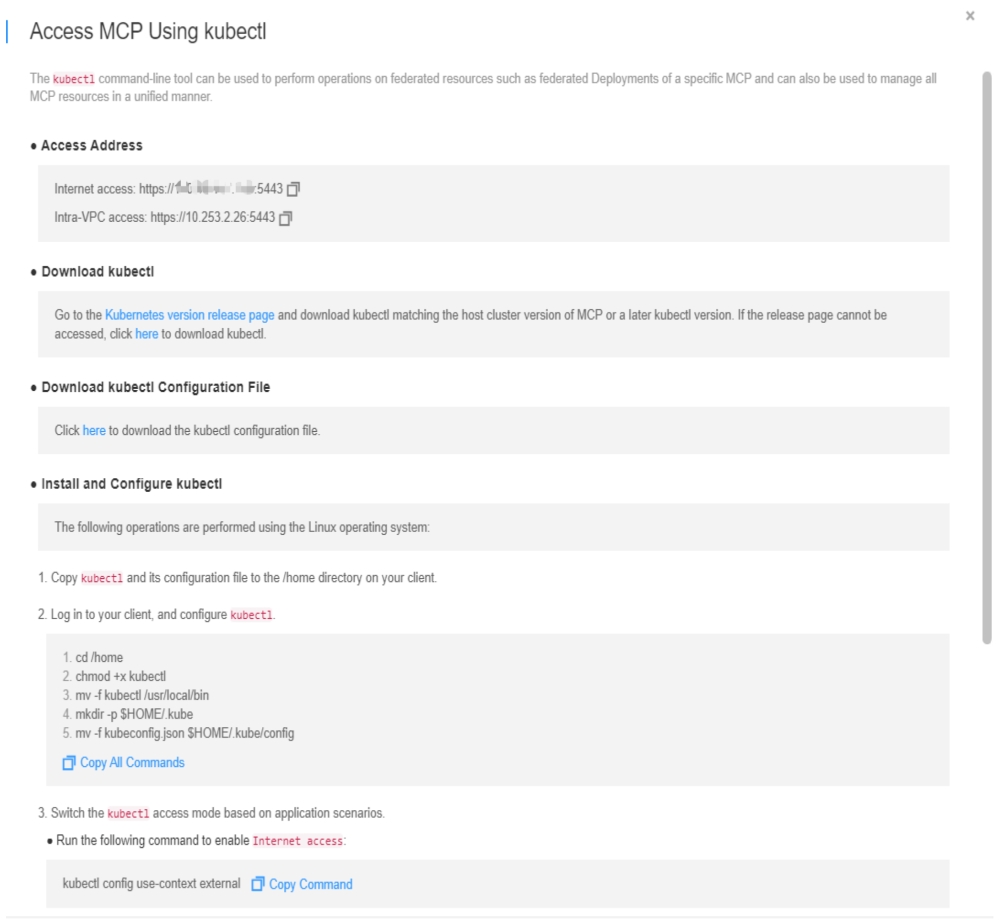
If you select public network access, you must prepare an ECS that can connect to a public network.

Accessing MCP by Using kubectl

Step 1 Log in to the MCP console and click **kubectl** on the **Dashboard** page.

Step 2 Obtain the MCP access address and download the kubectl and its configuration file according to the instruction provided in **Figure 5-5**.

Figure 5-5 Accessing MCP by using kubectl



Access MCP Using kubectl

The kubectl command-line tool can be used to perform operations on federated resources such as federated Deployments of a specific MCP and can also be used to manage all MCP resources in a unified manner.

- Access Address

Internet access: https://[redacted]:5443

Intra-VPC access: https://10.253.2.26:5443

- Download kubectl

Go to the Kubernetes version release page and download kubectl matching the host cluster version of MCP or a later kubectl version. If the release page cannot be accessed, click here to download kubectl.

- Download kubectl Configuration File

Click here to download the kubectl configuration file.

- Install and Configure kubectl

The following operations are performed using the Linux operating system:

1. Copy kubectl and its configuration file to the /home directory on your client.

2. Log in to your client, and configure kubectl.

```
cd /home
chmod +x kubectl
mv -f kubectl /usr/local/bin
mkdir -p $HOME/.kube
mv -f kubeconfig.json $HOME/.kube/config
```

3. Switch the kubectl access mode based on application scenarios.

- Run the following command to enable Internet access:

```
kubectl config use-context external
```

Step 3 Install and configure kubectl (A Linux OS is used as an example).

1. Copy kubectl and its configuration file to the **/home** directory on your client.
2. Log in to your client, and configure kubectl.

```
cd /home
chmod +x kubectl
mv -f kubectl /usr/local/bin
mkdir -p $HOME/.kube
mv -f kubeconfig.json $HOME/.kube/config
```

3. Switch the kubectl access mode based on application scenarios.
   - Run the following command to enable Internet access:
     ```
     kubectl config use-context external
     ```
   - Run the following command to enable intra-VPC access:
     ```
     kubectl config use-context internal
     ```
     This mode can be used only if you have already established an internal network connection.

----End

Deploying a Workload by Using kubectl

The following describes how to deploy a multi-cluster Nginx workload on different clusters.

Step 1 Access MCP by following the procedure described in **Accessing MCP by Using kubectl**.

Step 2 Query the clusters managed by MCP.

```
kubectl get clusters
```

Step 3 Create a Deployment.

```
kubectl create deployment nginx --image=nginx
```

Step 4 Create a propagation policy.

```
cat <<EOF | kubectl apply -f -
apiVersion: policy.karmada.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: nginx-propagation
spec:
  resourceSelectors:
    - apiVersion: apps/v1
      kind: Deployment
      name: nginx
  placement:
    clusterAffinity:
      clusterNames:
        - cluster1   #Cluster name obtained in step 2
        - cluster2
EOF
```

----End

5.1.7 Other Operations

Upgrading an MCP Instance

After MCP of a new version is released, you can upgrade the MCP components.

Step 1 Log in to the MCP console.

Step 2 Move the cursor to the MCP name and click **Upgrade**.

NOTE

You can also upgrade the MCP instance by choosing **More** > **Upgrade MCP** on the top of the **Dashboard** page.

----End

Deleting a Cluster from MCP

This operation only deletes the cluster from MCP, but does not delete the cluster itself and compute resources in the cluster.

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

Step 2 In the cluster list, locate the row that contains the target cluster, click **Delete** in the **Operation** column, and delete the cluster as prompted.

----End

Deleting an MCP Instance

Deleting an MCP instance does not delete the clusters under the MCP.

Step 1 Log in to the MCP console. In the navigation pane, choose **Dashboard**.

Step 2 Choose **More** > **Delete MCP** and delete the MCP instance as prompted.

----End

5.2 Network Management

5.2.1 Network Overview

A single cluster provides different workload access types to address diverse scenarios. After a Service or ingress is created on the MCP console, the Service or ingress with the same name will be automatically created on each cluster deployed on this MCP instance. When a Service or ingress created on the MCP console is modified or deleted on the cluster console, a message is displayed indicating that the operation is successful. However, a Service or ingress with the same name will be automatically re-created.

- **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**.

- **NodePort**

  A workload can be accessed by other workloads in the same VPC using the IP address of a cluster node. The NodePort access type is applicable to the scenario in which other workloads in the same VPC in the cloud need to access the workload in the Kubernetes cluster.

- **LoadBalancer**

  A workload can be accessed from a public network through a load balancer. This access type is applicable to Services that need to be exposed to a public network in the system. The access address is in the format of <IP address of public network load balancer>:<access port>, for example, **10.117.117.117:80**.

- **Ingress**

  Enhanced load balancer is used for an ingress. Compared with layer-4 load balancing, layer-7 load balancing newly supports Uniform Resource Identifiers (URI) configurations and distributes access traffic to the corresponding service based on the corresponding URIs. In addition, different functions are implemented based on various URIs. The access address is in the format of <IP address of public network load balancer>:<access port><defined URI>, for example, **10.117.117.117:80/helloworld**.

5.2.2 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**.

Methods of Setting the Access Type

You can set the access type by using either of the following methods:

- Set the access type when creating a Deployment. For details, see **Setting the Access Type When Creating a Deployment**.
- Set the access type after creating a Deployment. For details, see **Setting the Access Type After Creating a Deployment**.

Setting the Access Type When Creating a Deployment

Step 1 In the **Configure Access Settings** step, click **Create Service**. For details, see **Deployments**.

- **Service Name**: Name of the Service to be created, which can be self-defined.
- **Access Mode**: Select **ClusterIP**.
- **Protocol**: Select the protocol used by the Service.
- **Container Port**: Port on which the workload in the container image listens. For example, the Nginx application listens on port 80 (container port).
- **Access Port**: Port mapped to the container port at the cluster-internal IP address. The application can be accessed at <cluster-internal IP address>:<access port>. The port number range is 1–65535.

Step 2 Click **OK**.

Step 3 Click **Next** and configure advanced settings.

Step 4 Click **Next** and customize cluster configurations. Click **OK**.

Step 5 Obtain the access address.

1. Click **Cluster Console** in the upper part of the page to access the cluster console.
2. On the cluster console, choose **Workloads** > **Deployments**, click the name of the added workload to access the details page, and click **View Access Mode** to obtain the access address.

Step 6 Log in to any node in the cluster where the application is located. For details, see **Logging In to a Linux ECS**.

Step 7 Run the **curl** command to check whether the application can be accessed normally. You can perform the verification by using the IP address or domain name.

NOTE

Only applications using TCP support the preceding verification modes.

- By using the IP address

  curl *10.247.84.62:8888*

  10.247.84.62:8888 is the access address obtained in **Step 5**.

  If the following information is displayed, the application 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:8888*

  nginx.default.svc.cluster.local is the domain name obtained in **Step 5**.

  If the following information is displayed, the application 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 Deployment

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Network Management**. On the page displayed, click **Create Service** under the **Services** tab.

Step 2 Set access parameters.

- **Service Name**: Name of a Service to be created. You can use the workload name as the Service name.
- **Namespace**: Namespace to which the service belongs.

- **Access Type**: Select **ClusterIP**.
- **Workload**: Select an associated workload.
- **Port Configuration**
  - **Protocol**: Protocol used by the Service.
  - **Container Port**: Port on which the workload listens. For example, the Nginx application listens on port 80 (container port).
  - **Access Port**: 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.
- **Cluster**: The cluster is the same as that on which the associated workload is deployed. It cannot be manually changed.

Step 3 Click **Create**. After the Service is created, you can view it in the list on the **Services** tab page.

----End

5.2.3 NodePort

A workload can be accessed by other workloads in the same VPC using the IP address of a cluster node. The NodePort access type is applicable to the scenario in which other workloads in the same VPC in the cloud need to access the workload in the Kubernetes cluster.

Methods of Setting the Access Type

You can set the access type by using either of the following methods:

- Set the access type when creating a Deployment. For details, see **Setting the Access Type When Creating a Deployment**.
- Set the access type after creating a Deployment. For details, see **Setting the Access Type After Creating a Deployment**.

Setting the Access Type When Creating a Deployment

The following procedure uses an Nginx workload as an example.

Step 1 In the **Configure Access Settings** step, click **Create Service**. For details, see **Deployments**.

- **Service Name**: Name of the Service to be created, which can be self-defined.
- **Access Mode**: Select **NodePort**.
- **Protocol**: Protocol 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**: Specify a port to which the container port will be mapped when the node private IP address is used for accessing the application. The port number range is 30000–32767. You are advised to select **Automatically generated**.
  - **Automatically generated**: The system automatically assigns a port number.

- **Manually specified**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

Step 2 Click **OK**.

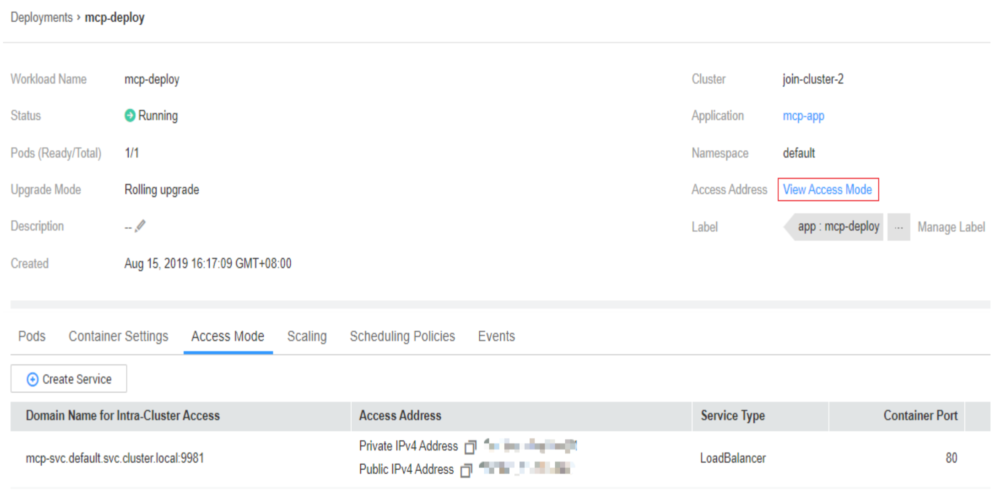
Step 3 Click **Next** and configure advanced settings.

Step 4 Click **Next** and customize cluster configurations. Click **OK**.

Step 5 Obtain the access address.

1. Click **Cluster Console** in the upper part of the page to access the cluster console.
2. On the cluster console, choose **Workloads** > **Deployments**, click the name of the added workload to access the details page, and click **View Access Mode** to obtain the access address.

Figure 5-6 Obtaining the node IP address



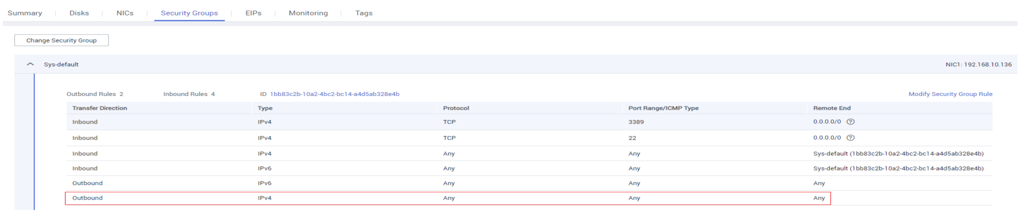
Workload Name	mcp-deploy	Cluster	join-cluster-2
Status	Running	Application	mcp-app
Pods (Ready/Total)	1/1	Namespace	default
Upgrade Mode	Rolling upgrade	Access Address	View Access Mode
Description	--	Label	app : mcp-deploy ··· Manage Label
Created	Aug 15, 2019 16:17:09 GMT+08:00		

Domain Name for Intra-Cluster Access	Access Address	Service Type	Container Port
mcp-svc.default.svc.cluster.local:9981	Private IPv4 Address [redacted] Public IPv4 Address [redacted]	LoadBalancer	80

Step 6 On the homepage of the Huawei Cloud management console, choose **Elastic Cloud Server** under **Compute**.

Step 7 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.

Figure 5-7 Confirming that the security group is open



Outbound Rules 2 Inbound Rules 4 ID 1bb83c2b-10a2-4bc2-bc14-a4d5ab328e4b

Transfer Direction	Type	Protocol	Port Range/ICMP Type	Remote End
Inbound	IPv4	TCP	3389	0.0.0.0/0
Inbound	IPv4	TCP	22	0.0.0.0/0
Inbound	IPv4	Any	Any	Sys-default (1bb83c2b-10a2-4bc2-bc14-a4d5ab328e4b)
Inbound	IPv6	Any	Any	Sys-default (1bb83c2b-10a2-4bc2-bc14-a4d5ab328e4b)
Outbound	IPv6	Any	Any	Any
Outbound	IPv4	Any	Any	Any

Step 8 Click **Remote Login**. On the login page, enter the username and password.

Step 9 Run the **curl** command to check whether the application can be accessed normally.

NOTE

NodePort services will also 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*:80

192.168.0.160:80 is the access address obtained in **Step 5**.

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 Deployment

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Network Management**. On the page displayed, click **Create Service** under the **Services** tab.

Step 2 Set access parameters.

- **Service Name**: Name of a Service to be created. You can use the workload name as the Service name.
- **Namespace**: Namespace to which the service belongs.
- **Access Type**: Select **NodePort**.
- **Workload**: Select an associated workload.
- **Port Configuration**
  – **Protocol**: Select the protocol used by the Service.
  – **Container Port**: Port on which the workload listens. For example, the Nginx application listens on port 80 (container port).
  – **Access Port**: Specify a port to which the container port will be mapped when the node 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.
  - **Manually specified**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

- **Cluster**: The cluster is the same as that on which the associated workload is deployed. It cannot be manually changed.

Step 3 Click **Create**. After the Service is created, you can view it in the list on the **Services** tab page.

----End

5.2.4 LoadBalancer

A workload can be accessed from a public network through a load balancer. This access type is applicable to Services that need to be exposed to a public network in the system. The access address is in the format of <IP address of public network load balancer>:<access port>, for example, **10.117.117.117:80**.

Prerequisites

A workload is available. If no workload is available, create one by following the procedure described in **Deployments**.

Setting the Access Type

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Network Management**. On the page displayed, click **Create Service** under the **Services** tab.

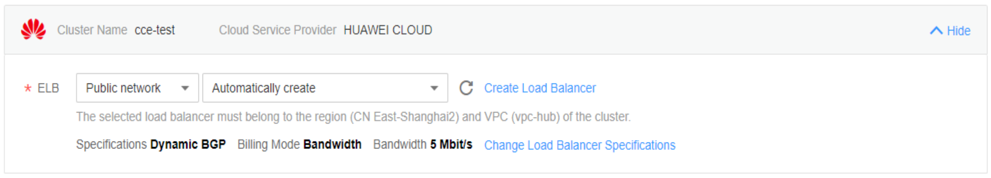
Step 2 Set access parameters.

- **Service Name**: Name of the Service to be created, which can be self-defined.
- **Namespace**: Namespace to which the service belongs.
- **Access Type**: Select **LoadBalancer**.
- **Workload**: Select an associated workload.
- **Port Configuration**
  - **Protocol**: Select the protocol used by the Service.
  - **Container Port**: Port on which the workload listens. For example, the Nginx application listens on port 80 (container port).
  - **Access Port**: Specify a port to map a container port to the ELB service. The port range is 1–65535. The port will be used when the application is accessed through the ELB service.
- **Domain Name**: Domain name used for access.
- **Cluster**: The cluster on which the Service is to be deployed is the same as that on which workload is deployed. It cannot be manually selected.

Step 3 Click **Next** and customize cluster configurations. Enhanced load balancers can be configured for each cluster.

- Huawei Cloud: Select an existing load balancer or let the system automatically create one. If you select **Automatically create**, click **Change Load Balancer Specifications**, and modify the specifications, billing mode, and bandwidth of the enhanced load balancer to be created.

Figure 5-8 Customized LoadBalancer Service settings - Huawei Cloud


Cluster Name cce-test Cloud Service Provider HUAWEI CLOUD

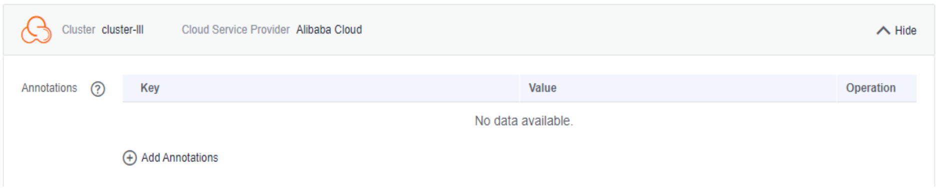
* ELB Public network Automatically create Create Load Balancer

The selected load balancer must belong to the region (CN East-Shanghai2) and VPC (vpc-hub) of the cluster.

Specifications **Dynamic BGP** Billing Mode **Bandwidth** Bandwidth **5 Mbit/s** Change Load Balancer Specifications

- Other clouds: The access annotation supports the key-value pair format. Alibaba Cloud is used as an example.Configure labels based on your service and vendor requirements.

Figure 5-9 Customized LoadBalancer Service settings - Alibaba Cloud


Cluster cluster-lll Cloud Service Provider Alibaba Cloud

Annotations

Key	Value	Operation

No data available.

  – To create an internal load balancer, add annotations based on the cloud service provider of your cluster. For details, see **Internal load balancer**.

Step 4 Click **Create**. After the Service is created, you can configure domain name access to interconnect the Service of the LoadBalancer type with the DNS service.

Step 5 Obtain the access address.

1. Click **Cluster Console** in the upper part of the page to access the cluster console.
2. On the cluster console, choose **Workloads** > **Deployments**, click the name of the added workload to access the details page, and click **View Access Mode** to obtain the access address.

----End

5.2.5 Ingresses

Enhanced load balancer is used for an ingress. Compared with layer-4 load balancing, layer-7 load balancing newly supports Uniform Resource Identifiers (URI) configurations and distributes access traffic to the corresponding service based on the corresponding URIs. In addition, different functions are implemented based on various URIs. The access address is in the format of <IP address of public network load balancer>:<access port><defined URI>, for example, **10.117.117.117:80/helloworld**.

Prerequisites

A workload is available. If no workload is available, create one by following the procedure described in **Deployments**.

Methods of Setting the Access Type

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Network Management**. On the page displayed, click **Create Ingress** under the **Ingresses** tab.

Step 2 Set general configuration parameters. General configurations refer to common parameter settings for all clusters.

- **Ingress Name**: Name of the ingress to be created, which can be self-defined.
- **Namespace**: Namespace to which the ingress belongs.
- **Cluster**: Select the cluster on which the ingress is to be deployed. If you select a CCE cluster, ensure that a NodePort Service has been created. For details on how to create a NodePort Service, see **NodePort**.

Step 3 Click **Next** and customize cluster configurations.

Customized cluster configurations on Huawei Cloud

- **ELB**: Select an existing load balancer or let the system automatically create one.
  - If you select an existing load balancer, make sure that the load balancer and the selected cluster belong to the same VPC and subnet.
  - If you set the load balancer type to **Public network** and select **Automatically create**, you can click **Change Load Balancer Specifications**, and modify the specifications, billing mode, and bandwidth of the instance to be created.
- **External Protocol**: HTTP and HTTPS are supported.
- **Key Certificate**: If **External Protocol** is set to **HTTPS**, select a key certificate.
- **External Port**: Port number that is open to the ELB service address. The port number can be specified randomly.
- **Domain Name**: Enter a domain name, for example, example.com.
- **Route Configuration**
  - **Route Match Rule**: **Prefix matching**, **Exact matching**, and **Regular expression matching** are available.
    - **Prefix matching**: If the URL is set to **/healthz**, the URL that meets the prefix can be accessed. For example, **/healthz/v1** and **/healthz/v2**.
    - **Exact matching**: Only the URL that is the same as the specified URL can be accessed. For example, if the URL is set to **/healthz**, only /healthz can be accessed.
    - **Regular expression matching**: 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**: Enter a mapping URL.
  - **Service Name**: Select a Service name.
  - **Container Port**: Select a container port.

Customized cluster configuration on other clouds

- **Annotations**: For details about the parameters in the metadata.annotations field, see the documentation of the corresponding cloud service provider.
- **External Protocol**: HTTP and HTTPS are supported.
- **Key Certificate**: If **External Protocol** is set to **HTTPS**, select a key certificate.
- **Domain Name**: Enter a domain name, for example, **example.com**.
- **Route Configuration**
  - **Mapping URL**: Enter a mapping URL.
  - **Service Name**: Select a Service name.
  - **Container Port**: Select a container port.

Step 4 Click **Create**. After the ingress is created, you can view it in the list on the **Ingresses** tab page.

Step 5 Obtain the access address.

1. Click **Cluster Console** in the upper part of the page to access the cluster console.
2. On the cluster console, choose **Workloads** > **Deployments**, click the name of the added workload to access the details page, and click **View Access Mode** to obtain the access address.

----End

5.3 Federated PVCs

5.3.1 Using Container Storage

When using container storage, you need to create a PersistentVolumeClaim (PVC). A PVC manages container storage volumes in multiple clouds in a unified manner. The volumes can be mounted to containers based on actual requirements, ensuring high reliability of applications. After a multi-cluster PVC is created on the MCP console, a PVC with the same name is created in all deployed clusters. After you modify or delete a multi-cluster PVC on the cluster console, a message is displayed indicating that the operation is successful. However, MCP will automatically re-create the PVC.

NOTE

A non-CCE cluster must support the container storage functions. Run the following command to query the StorageClass configuration and the interconnected backend storage service of the cluster. For more information about StorageClass, see **Storage Classes**.

```
kubectl get storageclass
```

Creating a PVC

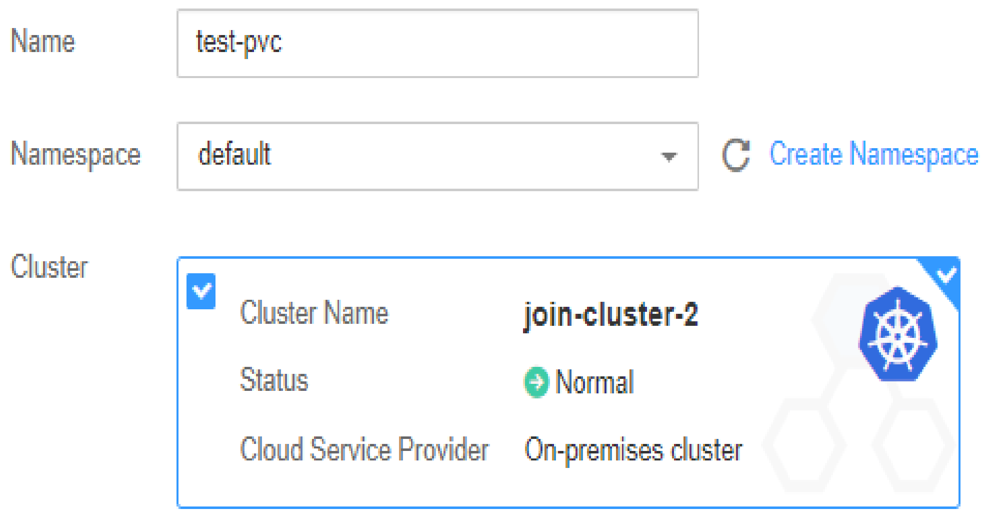
Step 1 Log in to the MCP console. In the navigation pane, choose **PVCs**. On the page displayed, click **Create PVC**.

Step 2 Specify basic information.

- **Name**: Enter the name of a PVC to be added. The name must be unique.

- **Namespace**: Namespace to which the PVC belongs. If you do not specify this parameter, the value **default** is used by default.
- **Cluster**: Select the cluster where the PVC is to be deployed.

Figure 5-10 Selecting a cluster


Name: test-pvc

Namespace: default

Cluster:

Cluster Name	join-cluster-2
Status	Normal
Cloud Service Provider	On-premises cluster

Step 3 Click **Next** and select a creation mode. Two creation modes are supported: **Dynamic** and **Standard NFS**.

Step 4 Customize cluster configurations.

- If you select **Dynamic**, set the parameters listed in **Table 5-1**.

Table 5-1 Parameters for dynamically creating a PVC

Parameter	Description
File System Source	If a Huawei Cloud cluster is selected, this parameter can be set to one of the following values: – **EVS**: block storage volumes. You can mount EVS volumes to a container path. When the container is migrated, the mounted EVS volume is migrated together. This storage class is applicable when data needs to be stored permanently. – **OBS**: object storage volumes. You can create OBS object storage volumes and mount them to a container path. Object storage applies to scenarios such as cloud workload, data analysis, content analysis, and hotspot objects. – **SFS**: file system volumes. You can create SFS volumes and mount them to a container path. The volumes created by the underlying SFS service can also be used. SFS volumes apply to workload scenarios where data needs to be persisted and read by and written to multiple nodes. Such scenarios include media processing, content management, big data analysis, and workload analysis. If a third-party cluster is selected, this parameter can be set only to **Other**.
Type	This parameter is valid only when **File System Source** is set to **Other**. The storage class supported by the cluster depends on the actual environment of the cluster.
Size (GB)	Capacity of the storage to be created.
Access Mode	– If **File System Source** is set to **EVS**, **Access Mode** must be set to **ReadWriteOnce**, that is, the volume can be mounted as read-write by only a single node. – If **File System Source** is set to **OBS** or **SFS**, **Access Mode** must be set to **ReadWriteMany**, that is, the volume can be mounted as read-write by multiple nodes. – If the selected cluster is a third-party cluster, the access mode can be **ReadWriteOnce** or **ReadWriteMany**.
Annotations	Annotations are attached to PVCs in the form of key-value pairs. 1. Click **Add Annotations**. 2. Set **Key** and **Value**.

- If you select **Standard NFS**, set the parameters listed in **Table 5-2**.

Table 5-2 Parameters for creating a PVC using standard NFS

Parameter	Description
Access Mode	Set this parameter to **ReadWriteMany**, that is, the volume can be attached as read-write by multiple nodes.
Mount Point Address	Enter the address of an existing storage server in the format of IP address (for example, **192.168.0.0**) or domain name (for example, **www.xx.com**).
Mount Point Path	Enter the shared directory of an existing storage device. The value must be an absolute path, for example, **/dir**.

Step 5 Click **Create**. After the PVC is created, click **Back to PVC List** to view the PVC status.

----End

Using Cloud Volumes

Step 1 After you add a container by referring to **Deployments**, choose **Data Storage** > **Cloud Volume**, and then click **Add Cloud Volume**.

Step 2 Configure parameters.

Table 5-3 Parameters for mounting a file system

Parameter	Description
Type	● File storage (NFS) ● Other
PVC Name	Select a created PVC. If no PVC is available, create a PVC first. You can also create a PVC in advance. For details about how to create a PVC, see **Creating a PVC**. **NOTICE** ● If you select an existing PVC, the selected PVC cluster must be the same as the cluster where the workload is deployed. Otherwise, the workload pods fail to be scheduled and the PVC cannot be found. ● The selected PVC must exist and be available. Otherwise, the workload will fail to mount volumes.

Parameter	Description
Add Container Path	1. Set **subPath** to a path of the database logical volume in Kubernetes. It is the subpath of the volume instead of the root path. If this parameter is left empty, the root path is used by default. Currently, only NFS supports subpath. 2. Set **Mount Path** to a path to which the database logical volume is mounted. You can click **Add Container Path** to add multiple paths. **NOTICE** – The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, the container may not function properly. You are advised to mount an empty directory into the container. If the directory is not empty, ensure that the directory does not contain any files that affect container startup. Otherwise, the files will be replaced, making it impossible for the container to be properly started. As a result, the workload creation will fail. – If the file system 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 database logical volumes mounted to the path. – **Read/Write**: You can modify the database logical volumes mounted to the path. Newly written data is not migrated if the container is migrated, which may cause a data loss.

Step 3 Click **OK**.

----End

Related Operations

After creating a PVC, you can update or delete it as described in **Table 5-4**.

Table 5-4 Other operations

Operation	Description
Deleting a PVC	1. Choose **More** > **Delete** in the **Operation** column of the target PVC. You can also delete the PVC records in batches. 2. Follow the prompts to delete the PVC. 3. After the deletion, the PVC on the cluster console is also deleted.
Viewing a YAML file	Click **View YAML** in the **Operation** column of the target PVC.

Operation	Description
Viewing events	View the event information about the PVC. **NOTE** Event data will be retained for 1 hour and then automatically deleted.

5.4 Namespaces

Namespaces enable division of cluster resources among multiple users through quotas. Namespaces are suited for environments in which multiple users spread across multiple teams or projects. A namespace created on the MCP console is a multi-cluster namespace. After it is created, a namespace with the same name will be created in all clusters that join the MCP instance.

NOTICE

- You can modify the resource quota of a multi-cluster namespace on the cluster console. The modified quota applies only to the current cluster. The namespace quota in other clusters does not change.
- A multi-cluster namespace will be created again after it is deleted from the cluster console.
- The namespaces mentioned in this section are multi-cluster namespaces.

Creating a Namespace

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Namespaces**. Click **Create Namespace**.

Step 2 Set namespace parameters based on **Table 5-5**.

Table 5-5 Parameters for creating a namespace

Parameter	Description
Namespace Name	Name of a namespace, which must be unique in a cluster.
Tag	Tag in the key-value pair format. Add tags to namespaces and define different attributes. You can learn the characteristics of each namespace through these tags.
Annotations	Annotation in the key-value pair format. Add annotations to the namespace.
Description	Description of the namespace.

Step 3 Click **Create** after the configuration is complete.

----End

Using Namespaces

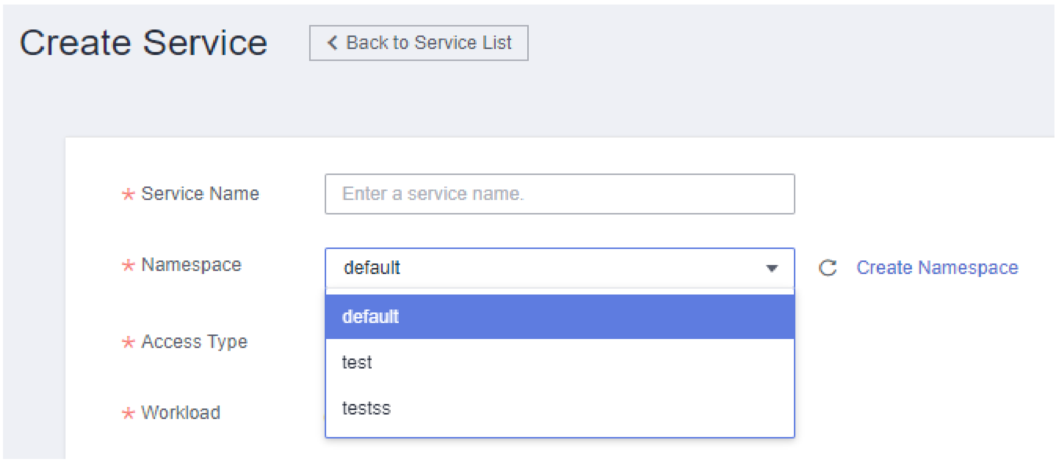
Namespaces can be used when creating Services, ingresses, domain name access, and PVCs. The following describes how a namespace is used during creation of a Service.

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

Step 2 Click **Create Service**.

Step 3 Select the namespace to which the Service resides.

Figure 5-11 Selecting a namespace



----End

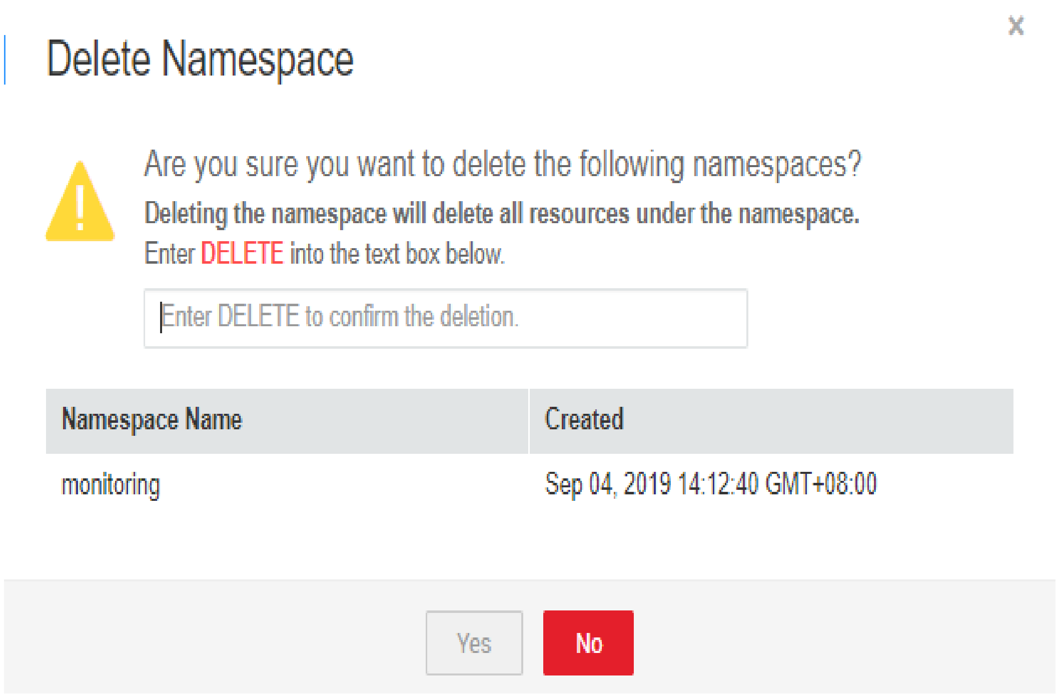
Deleting a Namespace

Deleting a multi-cluster namespace will delete all data resources related to it. Exercise caution when performing this operation.

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

Step 2 In the namespace list, select the target namespace and click **Delete** in the **Operation** column.

Figure 5-12 Deleting a namespace



Are you sure you want to delete the following namespaces?

Deleting the namespace will delete all resources under the namespace.

Enter DELETE into the text box below.

Enter DELETE to confirm the deletion.

Namespace Name	Created
monitoring	Sep 04, 2019 14:12:40 GMT+08:00

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

----End

5.5 Configuration Center

5.5.1 ConfigMaps

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

ConfigMaps provide the following benefits:

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

After a ConfigMap is created on the MCP console, a ConfigMap with the same name will be created in all clusters deployed on this MCP.

Creating a ConfigMap

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Configuration Center**, and click **Create ConfigMap** under the **ConfigMaps** tab.

Step 2 Set the parameters listed in **Table 5-6**.

Table 5-6 Parameters for creating a ConfigMap

Parameter	Description
Name	Name of a ConfigMap, which must be unique in a namespace.
Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value **default** is used by default.
Data	The workload configuration data can be used in a container or used to store the configuration data. **Key** specifies the file name, and **Value** specifies the file content. 1. Click **Add Data**. 2. Set **Key** and **Value**.
Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define identified attributes of these objects and can be used to manage and select objects. 1. Click **Add Label**. 2. Set **Key** and **Value**.
Description	Description of the ConfigMap.
Cluster	Cluster that will use the ConfigMap you created.

Step 3 Click **Create** after the configuration is complete.

----End

Using a ConfigMap

After the ConfigMap is created, you can mount the ConfigMap to a container during workload creation (for details, see **Deployments**). Then, you can read the ConfigMap data from the mount path of the container.

Related Operations

After creating a ConfigMap, you can update or delete it as described in **Table 5-7**.

Table 5-7 Other operations

Operation	Description
Viewing a YAML file	Click **View YAML** in the row where the target ConfigMap resides to view its YAML file.

Operation	Description
Updating a ConfigMap	1. Choose **More** > **Update** in the **Operation** column of the target ConfigMap. 2. Modify the ConfigMap information according to **Table 5-6**. 3. Click **Update**.
Deleting a ConfigMap	1. Select the ConfigMaps to be deleted. 2. Choose **More** > **Delete** in the **Operation** column. 3. Click **Yes**.
Viewing events	Select a ConfigMap and choose **More** > **View Event** in the **Operation** column to view the events of the ConfigMap. **NOTE** Event data will be retained for 1 hour and then automatically deleted.

5.5.2 Secrets

A secret is a type of resource that holds sensitive data, such as authentication and key information, required by a workload. Its content is user-defined. After a secret is created on the MCP console, a secret with the same name is created in all clusters deployed on this MCP instance.

Creating a Secret

Step 1 Log in to the MCP console. In the navigation pane, choose **Resource Management** > **Configuration Center**, and click **Create Secret** under the **Secrets** tab.

Step 2 Set the parameters listed in **Table 5-8**.

Table 5-8 Parameters for creating a secret

Parameter	Description
Name	Name of a secret, which must be unique in the same namespace.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Cluster	Cluster whether the secret is to be deployed.

Parameter	Description
Type	Type of the secret. ● Opaque: common secret. In high-sensitive scenarios, you are advised to encrypt sensitive data using data encryption services and then store the encrypted data in secrets. ● kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository. Enter an image repository address. ● IngressTLS: a secret that stores the certificate required by ingresses (layer-7 load balancing Services). ● Other: another type of secret, which is specified manually.
Data	Workload secret data can be used in containers. ● If the secret type is **Opaque**, enter the key and value. The value must be a Base64-encoded value. You can click **Perform Base64 Encoding** in the **Operation** column to convert the entered value to a Base64-encoded value. For details about the Base64 encoding method, see **Base64 Encoding**. ● If the secret type is **kubernetes.io/dockerconfigjson**, enter the account and password of the private image repository.
Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define identified attributes of these objects and can be used to manage and select objects. 1. Click **Add Label**. 2. Set **Key** and **Value**.
Description	Description of the secret.

Step 3 Click **Create** after the configuration is complete.

The new secret is displayed in the secret list.

----End

Using a Secret

After the secret is created, you can mount the secret to a container during workload creation (for details, see **Deployments**). Then, you can read the secret data from the mount path of the container.

Base64 Encoding

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

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

Related Operations

After a secret is created, you can perform the operations described in **Table 5-9**.

Table 5-9 Other operations

Operation	Description
Viewing a YAML file	Click **View YAML** in the row where the target secret resides to view its YAML file.
Updating a secret	1. Choose **More** > **Update** in the row where the target secret resides. 2. Modify the secret information according to **Table 5-8**. 3. Click **Update**.
Deleting a secret	1. Select the secrets to be deleted. 2. Choose **More** > **Delete** in the **Operation** column. 3. Click **Yes**.
Viewing events	Select a secret and choose **More** > **View Event** in the **Operation** column to view the events of the secret. **NOTE** Event data will be retained for 1 hour and then automatically deleted.

6 Policy Center

6.1 Propagation Policies

Karmada provides an independent policy API to configure resource propagation policies. For more information about Karmada, see **Karmada**.

Creating a Propagation Policy

Step 1 Log in to the MCP console. In the navigation pane, choose **Policy Center** > **Propagation Policies**, and click **Create Propagation Policy**.

Step 2 Set basic information about the propagation policy.

- **Policy Name**: Name of a propagation policy to be created, which must be unique.
- **Namespace**: Namespace to which the workload belongs.
- **Associated Resource Type**: Select the type of the resources to be associated. For a CRD resource, you can set the kind, API version, and name for it.
- **Associated Resource**: Select resources to be associated.

  NOTE

  Create a resource template in Karmada before associating the resources.
- **Cluster**: Cluster to which associated resources can be scheduled. If this parameter is not set, resources in any cluster can be scheduled.
- **Provider**: Vendor of clusters to which the associated resource can be scheduled. If this parameter is not set, the resource can be scheduled to clusters of any vendor.
- **Region**: Region to which associated resources can be scheduled. If this parameter is not set, the resource can be scheduled to clusters in any region.
- **Label**: Label of the cluster to which associated resources can be scheduled. If this parameter is not set, labels are not considered during resource scheduling.
- **Toleration**: Tolerance of associated resources to cluster stains. If this parameter is not set, cluster taint is not considered during scheduling.

NOTE

You can set the cluster taint by choosing **Cluster Overview** > **Taints** tab page on the cluster console.

Karmada calculates the clusters to which associated resources can be scheduled based on the cluster, provider, region, label, and taint tolerance.

Step 3 After configuration, click **Create**.

Step 4 Check the scheduling result.

1. Choose **Policy Center** > **Propagation Policies**, and click the name of the target propagation policy to open its details page.

2. In the **Cluster Distribution** section, you can view the target cluster to which associated resources are scheduled.

Step 5 Click **Cluster Console** to go to the console of the target cluster. You can see that the associated resources have been scheduled to the cluster.

NOTE

When multiple propagation policies are associated with the same resource, only the last policy (based on the lexicographical order of the policy names) takes effect. You are not advised to associate multiple propagation policies with the same resource.

----End

Updating a Propagation Policy

Step 1 Log in to the MCP console. In the navigation pane, choose **Policy Center** > **Propagation Policies**.

Step 2 In the policy list, click **Update** in the **Operation** column of the propagation policy to be updated.

Step 3 Update the label and taint information of the propagation policy.

- **Label**: Label of the cluster to which associated resources can be scheduled. If this parameter is not set, labels are not considered during resource scheduling.
- **Toleration**: Tolerance of associated resources to cluster stains. If this parameter is not set, cluster taint is not considered during scheduling.

Step 4 Click **Update**.

----End

6.2 Differentiation Policies

Karmada provides independent differentiated policy APIs to configure differentiated configurations related to the clusters. For more information about Karmada, see **Karmada**.

Creating a Differentiation Policy

Step 1 Log in to the MCP console. In the navigation pane, choose **Policy Center** > **Differentiation Policies**, and click **Create Using YAML**.

Step 2 Edit the YAML parameters. The following is an example:

```
apiVersion: policy.karmada.io/v1alpha1
kind: OverridePolicy
metadata:
  name: nginx-ccecluster-deployment
  namespace: default
spec:
  overriders:
    plaintext:
      - operator: replace
        path: /spec/replicas
        value: 2
      - operator: add
        path: /spec/template/spec/containers
        value:
          - image: 'nginx:perl'
            imagePullPolicy: IfNotPresent
            name: container-0
            resources:
              limits:
                cpu: 250m
                memory: 512Mi
              requests:
                cpu: 250m
                memory: 512Mi
  resourceSelectors:
    - apiVersion: apps/v1
      kind: Deployment
      name: nginx
      namespace: default
  targetCluster:
    clusterNames:
      - ccecluster
```

Table 6-1 Differentiation policy parameters

Parameter	Type	Description
apiVersion	String	API version. The value is fixed at **policy.karmada.io/v1alpha1** and cannot be changed.
kind	String	API type. The value is fixed at **OverridePolicy** and cannot be changed.
metadata	**metadata** object	Basic information about the differentiation policy. Metadata is a collection of attributes.
spec	**spec** object	Detailed properties of the differentiation policy. Objects are created or updated based on the description of **spec**.

Table 6-2 metadata

Parameter	Type	Description
name	String	Name of the differentiation policy. The name must be unique and contains 1 to 56 characters starting with a letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
namespace	String	Namespace to which the differentiation policy belongs.

Table 6-3 spec

Parameter	Type	Description
overriders	**overriders** object	Content of the differentiation policy.
resourceSelectors	**resourceSelectors** object	Associated resources. The differentiation policy takes effect for resources that meet the filter criteria.
targetCluster	**targetCluster** object	Cluster where the differentiation policy takes effect.

Table 6-4 overriders

Parameter	Type	Description
plaintext	Array of **plaintext** object	Content of the differentiation policy. Multiple policies can be added.

Table 6-5 plaintext

Parameter	Type	Description

operator	String	Differentiation operations performed on the target resource, that is, adding, deleting, or modifying the content in the operation path.
		● **add**: Add a field to the target resource template. The field is specified by the operation path.
		● **remove**: Delete a field from the target resource template. The field is specified by the operation path.
		● **replace**: Update a field in the target resource template. The field is specified by the operation path.
		In the following example, the **replace** operation changes the value of **/spec/replicas** in the YAML file of the target resource to **2**. That is, the number of pods is changed to 2.
		`- operator: replace`
		`  path: /spec/replicas`
		`  value: 2`

path	String	Operation path, which indicates the location of the differentiated content in the YAML file of the target resource. For example, if this parameter is set to **/spec/replicas**, the operation is performed on the **/spec/replicas** field of the target resource.
value	Type of the target resource parameter	Content to be added, deleted, or modified in the specified operation path. ● If **operator** is set to **add**, set this parameter to the content to be added to the target resource path. ● If **operator** is set to **remove**, set this parameter to any value. This parameter is meaningless during deletion. ● If **operator** is set to **replace**, set this parameter to the new content for replacement in the target resource path.

Table 6-6 resourceSelectors

Parameter	Type	Description
apiVersion	String	API version of the target resource.
kind	String	API type of the target resource.

Parameter	Type	Description
name	String	Name of the target resource.
namespace	String	Namespace to which the workload belongs.

Table 6-7 targetCluster

Parameter	Type	Description
clusterNames	Array of strings	Name of the cluster where the differentiation policy takes effect.

Step 3 Click **Create** after you finish the editing.

Step 4 Click **Cluster Console** to go to the console of the differentiated cluster to view the differentiation result of the associated resources.

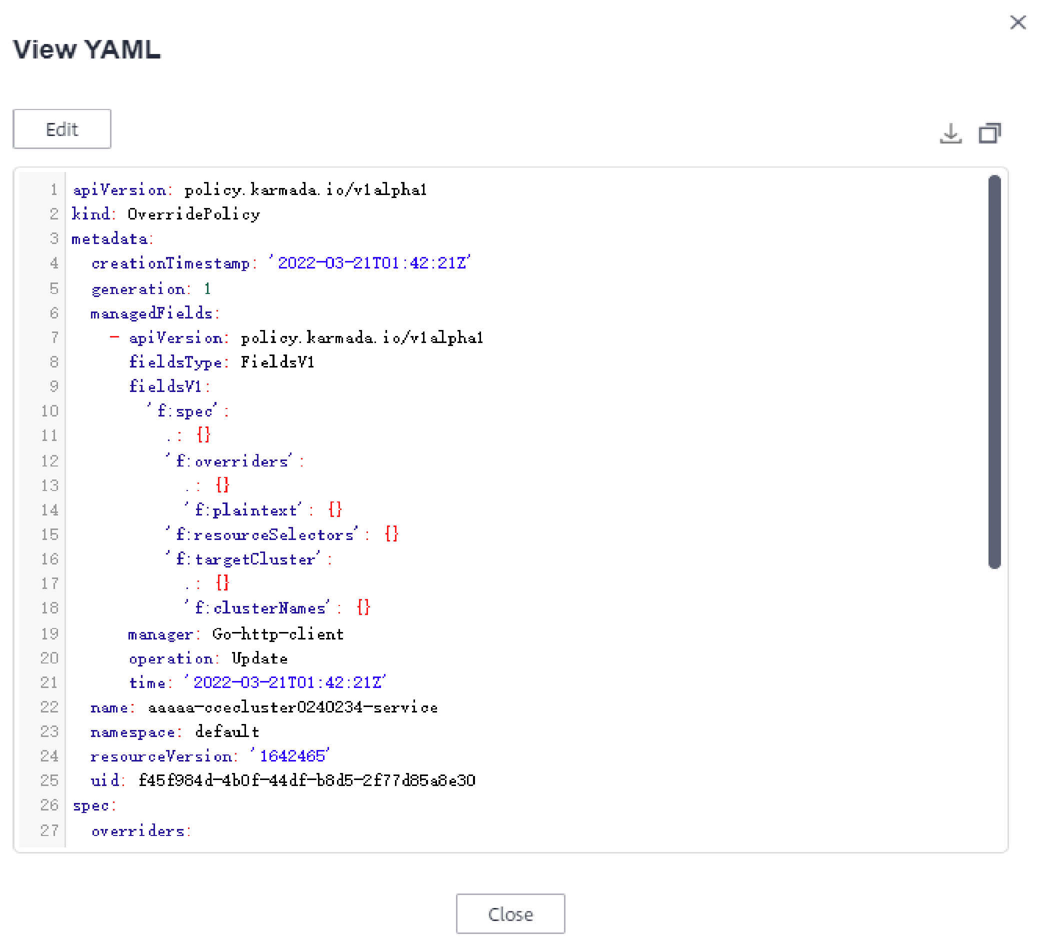
----End

Modifying a Differentiation Policy

Step 1 Log in to the MCP console. In the navigation pane, choose **Policy Center** > **Differentiation Policies**.

Step 2 In the policy list, click **View YAML** in the **Operation** column of the differentiation policy to be updated.

Step 3 In the displayed dialog box, click **Edit** to modify the YAML file.



```yaml
apiVersion: policy.karmada.io/v1alpha1
kind: OverridePolicy
metadata:
  creationTimestamp: '2022-03-21T01:42:21Z'
  generation: 1
  managedFields:
    - apiVersion: policy.karmada.io/v1alpha1
      fieldsType: FieldsV1
      fieldsV1:
        'f:spec':
          .: {}
          'f:overriders':
            .: {}
            'f:plaintext': {}
          'f:resourceSelectors': {}
          'f:targetCluster':
            .: {}
            'f:clusterNames': {}
      manager: Go-http-client
      operation: Update
      time: '2022-03-21T01:42:21Z'
  name: aaaaa-ccecluster0240234-service
  namespace: default
  resourceVersion: '1642465'
  uid: f45f984d-4b0f-44df-b8d5-2f77d85a8e30
spec:
  overriders:
```

Step 4 Click **Update** after you finish the modification.

Step 5 Click **Cluster Console** to go to the console of the differentiated cluster to view the differentiation result of the associated resources.

----End

6.3 Domain Name Access

Applications deployed in different clusters can be accessed using a unified public domain name. After you configure a public domain name, MCP can use it as a root domain name to generate a complete domain name for applications. You can configure domain name access to interconnect a Service and ingress with Huawei Cloud DNS so that applications deployed across clusters can be accessed through the unified domain name. In addition, you can customize the traffic distribution ratio to best suit your needs.

Configuring a Domain Name

Step 1 Create a public zone.

- If you have not created a public zone, **create a public zone**.
- If you have created a public zone, go to **Step 2**.

Step 2 Configure a domain name.

Select the domain name that has been configured and click **Configure Domain Name**.

----End

Creating Domain Name Access

After a Deployment is created, you can click **Create Service** to create a Service of the LoadBalancer type so that the Deployment can provide services for external systems. On the page indicating that the LoadBalancer Service is created, click **Create Domain Name Access**.

Step 1 Log in to the MCP console, choose **Domain Name Access** in the navigation pane, and click **Create Domain Name Access**.

Step 2 Set parameters of the associated Service.

- **Namespace**: Select a namespace.
- **Target Service**: Select a target service. If no LoadBalancer Service has been associated, create such a LoadBalancer Service first. For details on how to create a Service, see **LoadBalancer**.

Step 3 Click **Next** and set the access mode.

- **Active/Standby**: The traffic will be distributed only to the selected active cluster. You can change the traffic ratio to change the role of active and standby clusters. If automatic switchover is enabled, the DNS traffic resolution will be automatically switched to the standby cluster when the active cluster becomes faulty.
- **Adaptive**: The traffic is automatically distributed based on the number of pods in each cluster. In addition, you can enable region affinity to allow users in a specific region to access a specific cluster.
- **Custom**: You can customize the traffic distribution ratio across all the clusters. In addition, you can enable region affinity to allow users in a specific region to access a specific cluster.

Step 4 Click **Create**. The creation task will take a period of time. You can click **Back to Domain Name Access List** or **View Domain Name Access Details** to view the created domain name access.

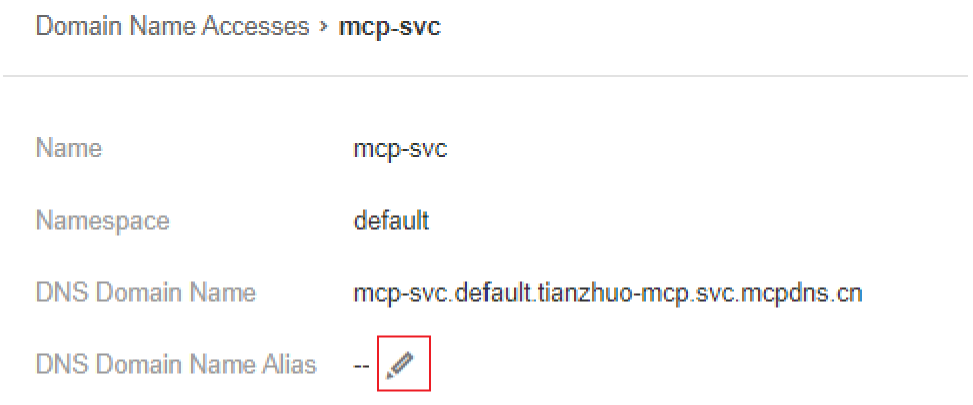
----End

Modifying an Alias

Step 1 Log in to the MCP console, and choose **Domain Name Access** in the navigation pane. Then, click a record to access the details page.

Step 2 Click ✎, enter an alias, and click ✓.

Figure 6-1 Modifying an alias



Domain Name Accesses › mcp-svc

Name	mcp-svc
Namespace	default
DNS Domain Name	mcp-svc.default.tianzhuo-mcp.svc.mcpdns.cn
DNS Domain Name Alias	--

----End

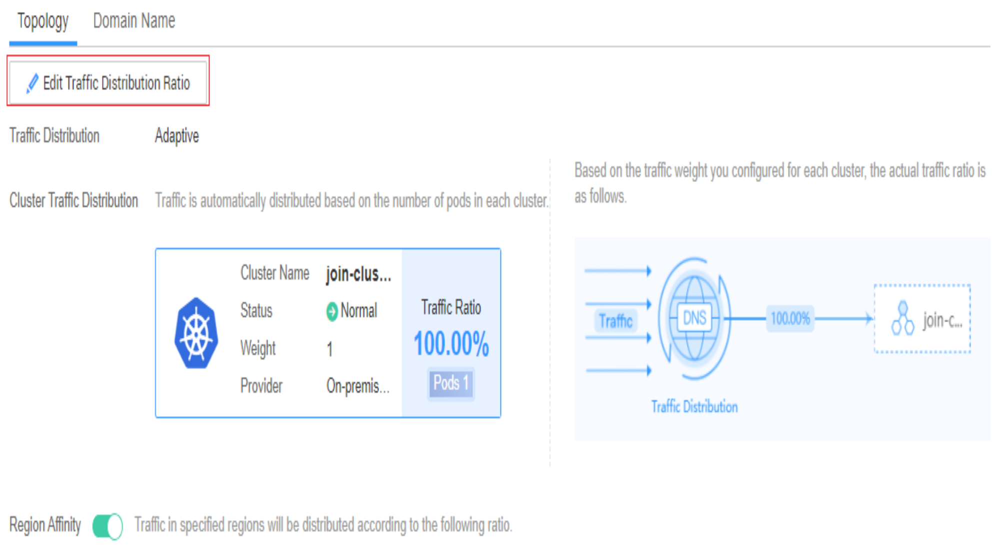
Modifying the Traffic Distribution Ratio

Step 1 Log in to the MCP console, and choose **Domain Name Access** in the navigation pane. Then, click a record to access the details page.

Step 2 On the **Topology** tab page, click **Edit Traffic Distribution Ratio**.

Step 3 Modify parameters and click **OK**.

Figure 6-2 Modifying the traffic distribution ratio



Traffic Distribution Adaptive

Cluster Traffic Distribution Traffic is automatically distributed based on the number of pods in each cluster.

Cluster Name join-clus...
Status Normal
Weight 1
Provider On-premis...
Traffic Ratio 100.00%
Pods 1

Based on the traffic weight you configured for each cluster, the actual traffic ratio is as follows.

Region Affinity Traffic in specified regions will be distributed according to the following ratio.

----End

Viewing the Domain Name Access Address

After a domain name access record is created, you can view the domain name access address in the domain name access list.

Step 1 Log in to the MCP console, and choose **Domain Name Access** in the navigation pane.

Step 2 In the domain name access list, view the value in the **Domain Name** column.

----End

Deleting Domain Name Access

Step 1 Log in to the MCP console, and choose **Domain Name Access** in the navigation pane.

Step 2 Click **Delete** in the **Operation** column of the target domain name access record.

Step 3 In the **Delete Domain Name Access** dialog box, click **Yes**.

----End

7 Monitoring Center

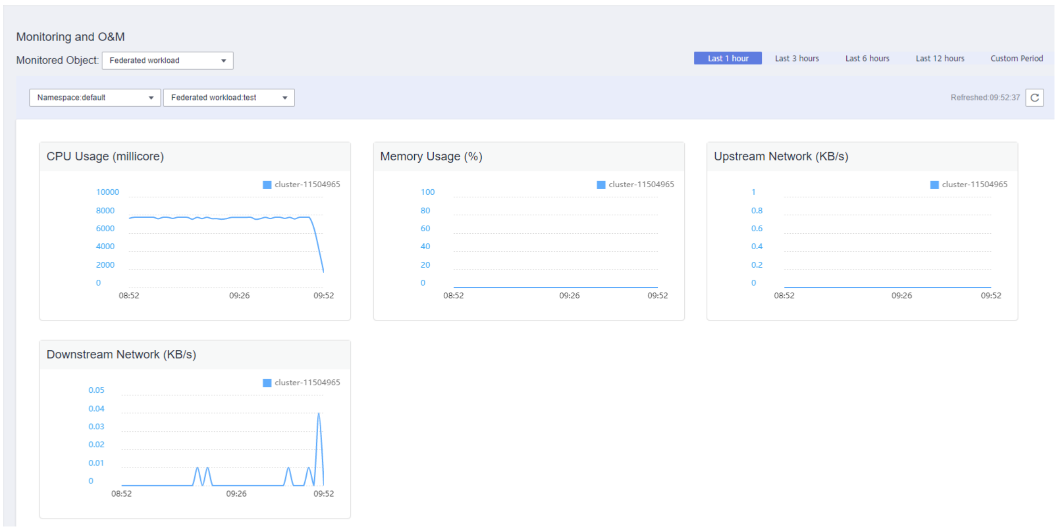
7.1 Monitoring and O&M

- If Prometheus is installed and connected to the cluster, you can view all monitoring information.
- If only the Metrics Server is interconnected, you can view the monitoring information about the CPU usage and memory usage of only nodes.
- If the Metrics Server is not interconnected, no monitoring information is displayed on the **Monitoring and O&M** page.

7.1.1 Monitoring Workloads

Step 1 Log in to the MCP console. In the navigation pane, choose **Monitoring Center** > **Monitoring and O&M**. Select **Workload** from the **Monitored Object** drop-down list.

Step 2 Select a namespace and a workload to view the monitoring information of the workload, such as the CPU usage, memory usage, uplink network, and downlink network.

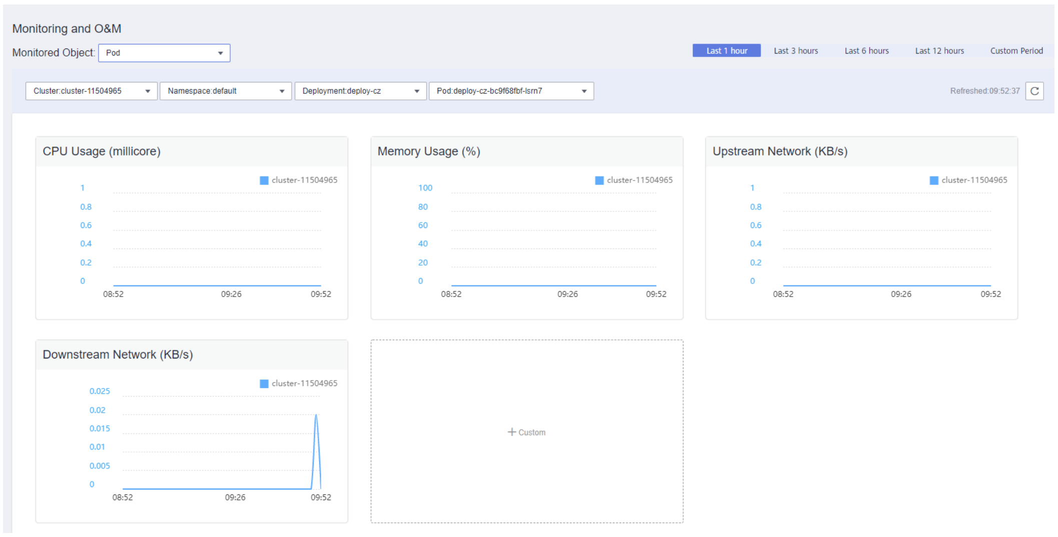


----End

7.1.2 Monitoring Pods

Step 1 Log in to the MCP console. In the navigation pane, choose **Monitor Center** > **Monitor and O&M**. Select **Pod** from the **Monitored Object** drop-down list.

Step 2 Select a cluster, namespace, workload, and pod to view the monitoring information of the pod, such as the CPU usage, memory usage, uplink network, and downlink network.


Monitoring and O&M
Monitored Object: Pod
Cluster:cluster-11504965 Namespace:default Deployment:deploy-cz Pod:deploy-cz-bc9f68fbf-lsrn7
Refreshed:09:52:37

Step 3 If you need to add a monitoring metric, click **Custom**.

Step 4 Select the metric name and tag name, and click **OK**.

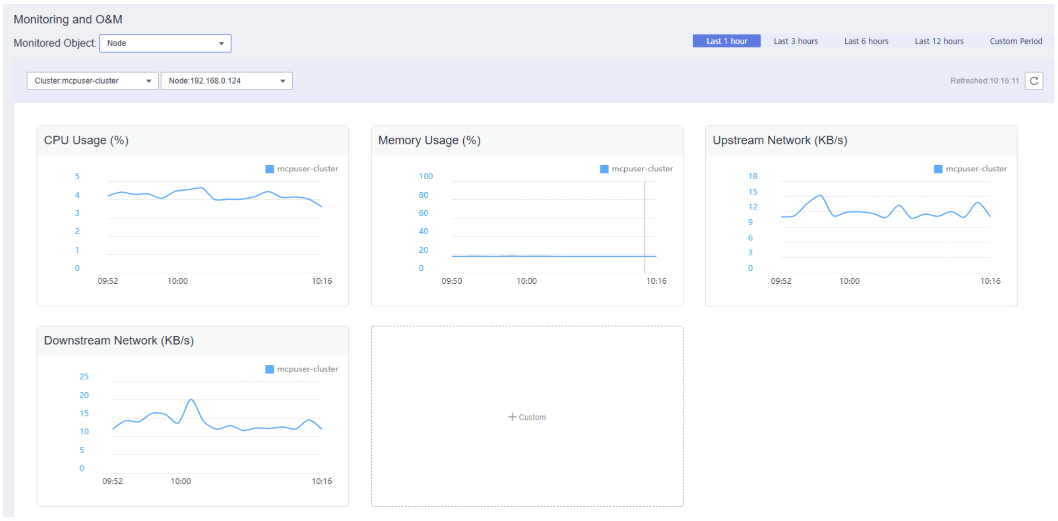
The information about the new monitoring metric will be displayed on the **Monitoring and O&M** page.

----End

7.1.3 Monitoring Nodes

Step 1 Log in to the MCP console. In the navigation pane, choose **Monitor Center** > **Monitor and O&M**. Select **Node** from the **Monitored Object** drop-down list.

Step 2 Select a cluster and node to view monitoring information of the node, such as the CPU usage, memory usage, uplink network, and downlink network.



Step 3 If you need to add a monitoring metric, click **Custom**.

Step 4 Select the metric name and tag name, and click **OK**.

The information about the new monitoring metric will be displayed on the **Monitoring and O&M** page.

----End

7.2 Connection Management

MCP can monitor Kubernetes clusters and show monitoring information on workloads, pods, and nodes, facilitating unified O&M.

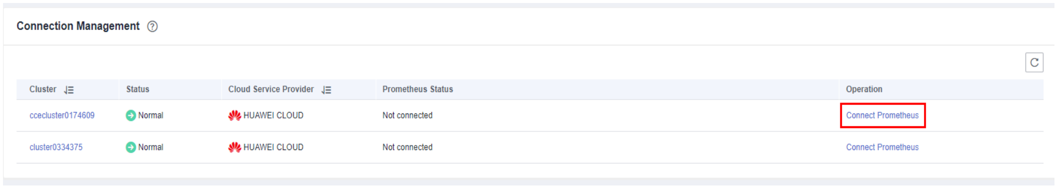
To enable monitoring data to be reported to MCP, install the Metric Server and Prometheus in Kubernetes clusters. For specific installation procedures, see the following sections:

- **Installing the Metrics Server**
- **Installing Prometheus**

After Prometheus is installed, connect it to MCP.

Step 1 Log in to the MCP console. In the navigation pane, choose **Monitor Center** > **Connection Management**. In the **Operation** column, click **Connect Prometheus**.

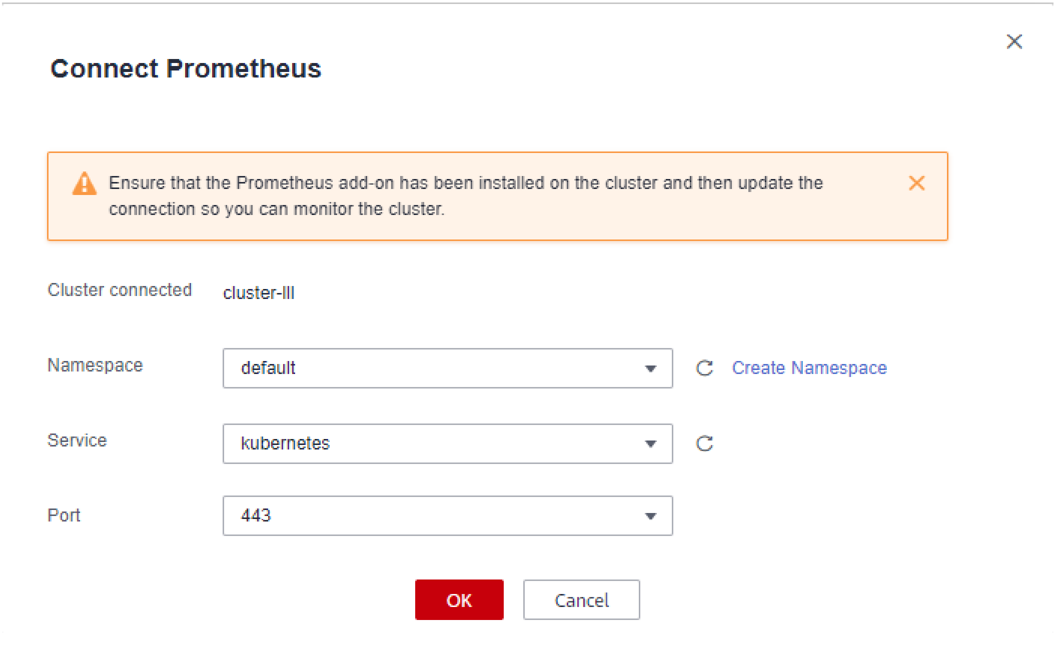
Figure 7-1 Connection management


Connection Management

Cluster	Status	Cloud Service Provider	Prometheus Status	Operation
cceccluster0174609	Normal	HUAWEI CLOUD	Not connected	Connect Prometheus
cluster0334375	Normal	HUAWEI CLOUD	Not connected	Connect Prometheus

Step 2 On the page that is displayed, select the namespace to which Prometheus belongs and select the service name and port of Prometheus.

Figure 7-2 Connecting to Prometheus



Connect Prometheus

Ensure that the Prometheus add-on has been installed on the cluster and then update the connection so you can monitor the cluster.

Cluster connected: cluster-lll

Namespace: default

Service: kubernetes

Port: 443

After the connection is complete, choose **Monitor Center** > **Monitoring and O&M** in the navigation pane. You can view the monitoring information about Kubernetes clusters from three dimensions: workload, pod, and node.

----End

7.3 Installing the Metrics Server

The cross-cluster Horizontal Pod Autoscaler (HPA) uses the Metrics API in Kubernetes to obtain the resource usage in clusters. To use cross-cluster HPA, install the Metrics Server in the clusters. The Metrics Server obtains data through the Kubelet Summary API to implement the Metrics API. The Metrics API can query only the current data. Historical data cannot be queried in this way.

Prerequisites

- The cluster version must be later than 1.13.
- The VM where the Metrics Server is to be installed can be connected to the Internet and can perform operations on the cluster using kubectl.

Installing the Metrics Server

Step 1 Enable the API aggregation layer on the VM where the Metrics Server is to be installed. To be specific, set the following startup parameters on kube-APIServer:

NOTE

For CCE clusters of Kubernetes 1.11, you need to **submit a service ticket** to enable the API aggregation layer. For CCE clusters of Kubernetes 1.13 or later version, the API aggregation layer is enabled by default.

```
--requestheader-client-ca-file=
--proxy-client-cert-file=
--proxy-client-key-file=
--requestheader-allowed-names=aggregator
--requestheader-extra-headers-prefix=X-Remote-Extra-
--requestheader-group-headers=X-Remote-Group
--requestheader-username-headers=X-Remote-User
```

Parameters are described as follows:

- **--requestheader-client-ca-file**: Root certificate bundle used to verify client certificates on incoming requests before trusting usernames in headers specified by **--requestheader-username-headers**.
- **--proxy-client-cert-file**: Client certificate used to prove the identity of the aggregator or kube-apiserver when it must call out during a request.
- **--proxy-client-key-file**: Private key for the client certificate used to prove the identity of the aggregator or kube-apiserver when it must call out during a request. This includes proxying requests to a user api-server and calling out to webhook admission plug-ins.
- **--requestheader-allowed-names**: List of common client certificate names to allow to provide usernames in headers specified by **--requestheader-username-headers**. If this parameter is left empty, any client certificate validated by the authorities in **--requestheader-client-ca-file** is allowed.
- **--requestheader-extra-headers-prefix**: List of header prefixes to be checked.
- **--requestheader-group-headers**: List of headers of groups to be checked.
- **--requestheader-username-headers**: List of headers of usernames to be checked.

If you are not running kube-proxy on a host running the API server, you must also configure the following parameter:

```
--enable-aggregator-routing=true
```

Step 2 Deploy the Metrics Server.

```
wget https://github.com/kubernetes-incubator/metrics-server/archive/v0.3.2.tar.gz
tar zxvf v0.3.2.tar.gz
cd metrics-server-0.3.2
kubectl create -f deploy/1.8+/
```

Step 3 Check whether metrics pods are successfully started.

```
kubectl get pods -nkube-system| grep metric
```

Step 4 If any pod in **Step 3** is not started, run the following command to query logs:

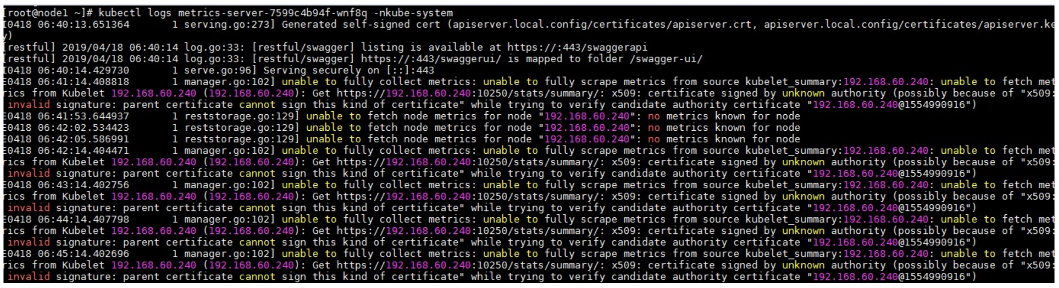
```
kubectl logs {pod-name} -nkube-system
```

Query logs. If the error information shown in **Figure 7-3** is displayed, run the following command:

```
kubectl edit deployment metrics-server -nkube-system
```

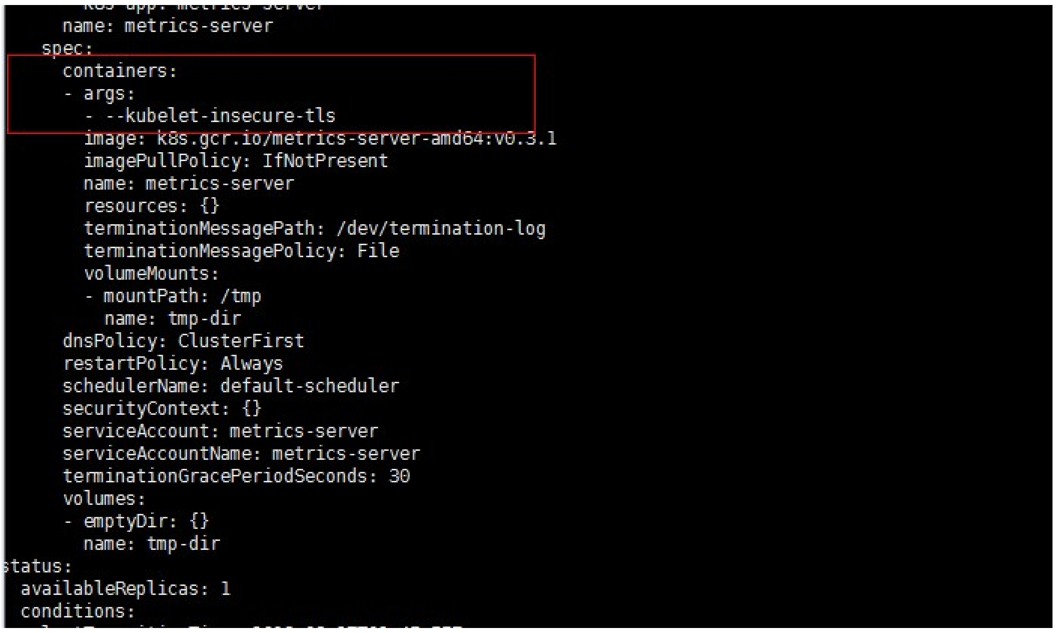
Add the startup parameter **--kubelet-insecure-tls** as shown in **Figure 7-4**, or configure a correct certificate (you need to provide the CA certificate of the Kubernetes cluster for metrics-server).

Figure 7-3 Error information



```
[root@node1 ~]# kubectl logs metrics-server-7599c4b94f-wnf8q -nkube-system
I0418 06:40:13.651364       1 serving.go:273] Generated self-signed cert (apiserver.local.config/certificates/apiserver.crt, apiserver.local.config/certificates/apiserver.ke
y)
[restful] 2019/04/18 06:40:14 log.go:33: [restful/swagger] listing is available at https://:443/swaggerapi
[restful] 2019/04/18 06:40:14 log.go:33: [restful/swagger] https://:443/swaggerui/ is mapped to folder /swagger-ui/
I0418 06:40:14.429730       1 serve.go:96] Serving securely on [::]:443
E0418 06:41:14.408818       1 manager.go:102] unable to fully collect metrics: unable to fully scrape metrics from source kubelet_summary:192.168.60.240: unable to fetch met
rics from Kubelet 192.168.60.240 (192.168.60.240): Get https://192.168.60.240:10250/stats/summary/: x509: certificate signed by unknown authority (possibly because of "x509:
 invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate "192.168.60.240@1554990916")
E0418 06:41:53.644937       1 reststorage.go:129] unable to fetch node metrics for node "192.168.60.240": no metrics known for node
E0418 06:42:02.534423       1 reststorage.go:129] unable to fetch node metrics for node "192.168.60.240": no metrics known for node
E0418 06:42:05.586991       1 reststorage.go:129] unable to fetch node metrics for node "192.168.60.240": no metrics known for node
E0418 06:42:14.404471       1 manager.go:102] unable to fully collect metrics: unable to fully scrape metrics from source kubelet_summary:192.168.60.240: unable to fetch met
rics from Kubelet 192.168.60.240 (192.168.60.240): Get https://192.168.60.240:10250/stats/summary/: x509: certificate signed by unknown authority (possibly because of "x509:
 invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate "192.168.60.240@1554990916")
E0418 06:43:14.402756       1 manager.go:102] unable to fully collect metrics: unable to fully scrape metrics from source kubelet_summary:192.168.60.240: unable to fetch met
rics from Kubelet 192.168.60.240 (192.168.60.240): Get https://192.168.60.240:10250/stats/summary/: x509: certificate signed by unknown authority (possibly because of "x509:
 invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate "192.168.60.240@1554990916")
E0418 06:44:14.407798       1 manager.go:102] unable to fully collect metrics: unable to fully scrape metrics from source kubelet_summary:192.168.60.240: unable to fetch met
rics from Kubelet 192.168.60.240 (192.168.60.240): Get https://192.168.60.240:10250/stats/summary/: x509: certificate signed by unknown authority (possibly because of "x509:
 invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate "192.168.60.240@1554990916")
E0418 06:45:14.402696       1 manager.go:102] unable to fully collect metrics: unable to fully scrape metrics from source kubelet_summary:192.168.60.240: unable to fetch met
rics from Kubelet 192.168.60.240 (192.168.60.240): Get https://192.168.60.240:10250/stats/summary/: x509: certificate signed by unknown authority (possibly because of "x509:
 invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate "192.168.60.240@1554990916")
```

Figure 7-4 Adding a startup parameter



```
      k8s-app: metrics-server
    name: metrics-server
  spec:
    containers:
    - args:
      - --kubelet-insecure-tls
      image: k8s.gcr.io/metrics-server-amd64:v0.3.1
      imagePullPolicy: IfNotPresent
      name: metrics-server
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /tmp
        name: tmp-dir
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    serviceAccount: metrics-server
    serviceAccountName: metrics-server
    terminationGracePeriodSeconds: 30
    volumes:
    - emptyDir: {}
      name: tmp-dir
status:
  availableReplicas: 1
  conditions:
```

Step 5 Check whether data is obtained correctly.

```
kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes
```

If any command output is displayed, the installation is successful.

----End

7.4 Installing Prometheus

Prometheus is an open-source software used for event monitoring and alarm reporting.

MCP can use Prometheus to report monitoring data of Kubernetes clusters to the monitoring center. To use this function, install Prometheus in the Kubernetes clusters.

Procedure

Use the Prometheus chart in the Kubernetes chart repository to install Prometheus. The Prometheus chart contains the following components. You can determine whether to deploy the components by setting corresponding parameters. By default, all components are deployed. This example shows how to set installation parameters for customized deployment.

- alertmanager: alarm center of the add-on. It receives alarms sent by Prometheus and manages alarm information through deduplication, grouping, and distribution.
- kube-state-metrics: converts the Prometheus metric data into a format that can be identified by Kubernetes APIs.
- node-exporter: deployed on each node to collect node monitoring data.
- pushgateway: intermediate component used to receive metrics. The Prometheus server obtains data from it.
- prometheus server: data collection server used for monitoring.

For details about the Prometheus chart, see **https://github.com/prometheus-community/helm-charts/tree/main/charts/prometheus**.

Step 1 Connect to the cluster using kubectl.

Step 2 Install Helm3.

```
wget https://get.helm.sh/helm-v3.6.1-linux-amd64.tar.gz
```

Decompress the file and save the helm binary file to the **/usr/local/bin** directory.

```
tar zxvf helm-v3.6.1-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin/helm
```

Step 3 Add a helm image repository.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

Run the **helm repo list** command to view the added helm image repository.

```
NAME                    URL
prometheus-community    https://prometheus-community.github.io/helm-charts
```

Step 4 Create a **value.yaml** file and set installation parameters.

```
vi value.yaml
```

You can configure the file content by referring to the following example.

```
pushgateway:
  enabled: false                          # Do not install pushgateway.
alertmanager:
  persistentVolume:
    existingClaim: pvc-name                   # The PVC must be created in advance.
server:                                 # Configuration of the Prometheus server
  persistentVolume:                       # Local persistent storage
    existingClaim: pvc-name
  global:
    external_labels:
```

Configuration description:

- pushgateway: generally not required to be deployed. You can set **pushgateway.enabled** to **false** to skip the deployment.
- alertermanager: deployed through a Deployment by default.
  - persistentVolume.existingClaim: Use an existing PVC name in the cluster and persistent storage to store local data.

    If persistent storage is not required, you can use **empty-dir** to store data. In this case, set this field in the preceding file as follows:

```
alertmanager:
  persistentVolume:
    enabled: false
```

- server: deployed through a Deployment by default.
  - persistentVolume.existingClaim: Use an existing PVC name in the cluster and persistent storage to store local data. By default, data is stored for 15 days.

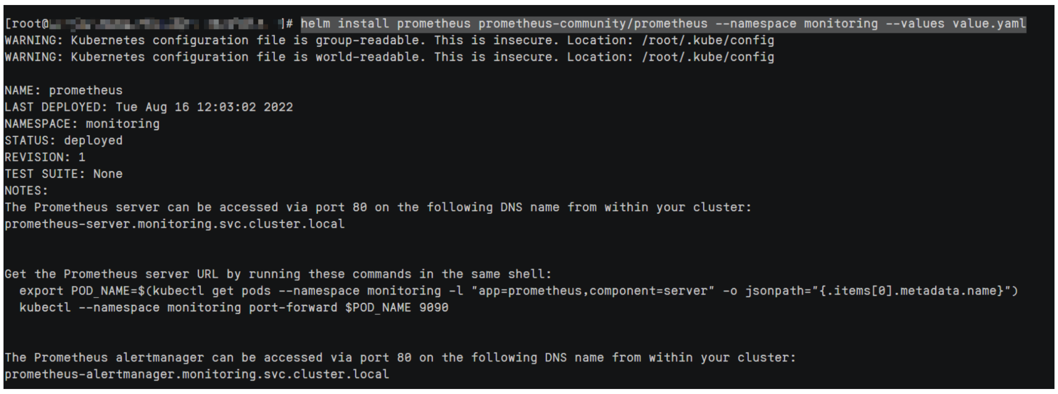
    If persistent storage is not required, you can use **empty-dir** to store data. In this case, set this field in the preceding file as follows:

    ```
    server:
      persistentVolume:
        enabled: false
    ```

Step 5 Install Prometheus.

```
helm install prometheus prometheus-community/prometheus --namespace monitoring --values value.yaml
```

- **--namespace**: Specify a namespace for installing Prometheus. You need to manually create it in advance.
- **--values**: Specify Prometheus installation parameters as described in **Step 4**.



```
[root@[redacted]]# helm install prometheus prometheus-community/prometheus --namespace monitoring --values value.yaml
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /root/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /root/.kube/config

NAME: prometheus
LAST DEPLOYED: Tue Aug 16 12:03:02 2022
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.monitoring.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-alertmanager.monitoring.svc.cluster.local
```

----End

7.5 Installing Prometheus Adapter

The Custom Metric API in Kubernetes is required if HPA is implemented based on custom metrics. Currently, most users use the Prometheus Adapter to provide the Custom Metric API. The Prometheus Adapter converts the received custom metric APIs to Prometheus requests and returns data queried from Prometheus to the Custom Metric API Server. The Prometheus Adapter code repository is available at **https://github.com/DirectXMan12/k8s-prometheus-adapter**.

Prerequisites

The API aggregation layer in the cluster has been enabled. For details, see **Configure the Aggregation Layer**.

Procedure

Use the Prometheus Adapter chart in the Kubernetes chart repository to install the Prometheus Adapter. For details about the chart, see **https://github.com/prometheus-community/helm-charts/tree/main/charts/prometheus-adapter**.

Step 1 Connect to the cluster using kubectl.

Step 2 Install Helm3.

```
wget https://storage.googleapis.com/kubernetes-helm/helm-v3.6.1-linux-amd64.tar.gz --no-check-certificate
```

Decompress the file and save the helm binary file to the **/usr/local/bin** directory.

```
tar zxvf helm-v3.6.1-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin/helm
```

Step 3 Add a helm image repository.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

Run the **helm repo list** command to view the added helm image repository.

```
NAME                    URL
prometheus-community    https://prometheus-community.github.io/helm-charts
```

Step 4 Check configurations in the **value.yaml** file.

- Information used by the Prometheus Adapter to connect to the Prometheus server includes the URL and port number. By default, the URL is **http://prometheus.default.svc** and the port number is **9090**. Set the URL and port number based on the actual Prometheus server. If Prometheus is installed by following the procedure provided in **Installing Prometheus**, use the following recommended configurations:

  Create a **value.yaml** file and set installation parameters.

  ```
  vi value.yaml
  ```

  ```
  prometheus:
    url: http://prometheus-server.monitoring.svc
    port: 80
    path: ""
  ```

- Customized rule configuration covers metric discovery rules, mapping between metrics and Kubernetes resources, exposed metric names in APIs, and corresponding Prometheus query statements. For details, see **https://github.com/DirectXMan12/k8s-prometheus-adapter/blob/master/docs/config.md**.

Step 5 Install the Prometheus Adapter.

```
helm install prometheus-adapter prometheus-community/prometheus-adapter --values value.yaml --
namespace monitoring
```

- **--namespace**: Specify a namespace for installing the Prometheus Adapter. You need to manually create it in advance.
- **--values**: Specify the installation parameters of the Prometheus Adapter as described in **Step 4**.

After the installation is complete, run the following command to query all custom metrics:

```
kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1"
```

----End

8 Image Repository

An image repository is provided by SoftWare Repository for Container (SWR) for you to easily store, manage, and deploy container images.

- If you use a Huawei Cloud image, upload the image first. For details, see **Uploading an Image Through a Container Engine Client**.
- To use a third-party image, you can configure it in the **Customize Cluster Configurations** step during Deployment creation. For details, see **Using Third-Party Images**.

Using Huawei Cloud Images

You can use Huawei Cloud images for clusters added on MCP (including other cloud clusters). When creating a Deployment, you can choose images from **My Images**. This section describes how to use an image when creating a Deployment named **game**.

Step 1 Log in to the MCP 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**: Select the cluster in which the application is to run.

Step 3 Click **Next** to add a container.

Click **Add Container**. On the **My Images** tab page, select a pushed image and click **OK**.

NOTE

If multiple clusters are deployed, you can customize configurations for these clusters when adding a container, or replace the image of a certain cluster.

Step 4 Complete the Deployment creation by referring to **Deployments**.

----End

Using Third-Party Images

You can use a third-party image for the clusters (including Huawei Cloud clusters) added on MCP. Select **Customized Configuration** next to **Container Settings** of the **Customize Cluster Configurations** step when creating a Deployment. This section describes how to use a third-party image when creating a Deployment named **game**.

Step 1 Log in to the MCP 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**: Select the cluster in which the application is to run.

Step 3 Click **Next** to add a container.

1. Click **Add Container**. On the **Select Image** tab page, select a third-party image.
2. Select a secret. If no secret is available, create one first.

   If your image repository does not need to be authenticated, select **No** for **Secret Authentication**.
3. Enter the IP address of the image.
4. Click **OK**.

Step 4 Complete the Deployment creation by referring to **Deployments**.

----End

9 Cluster Console

9.1 Viewing Nodes in a Cluster

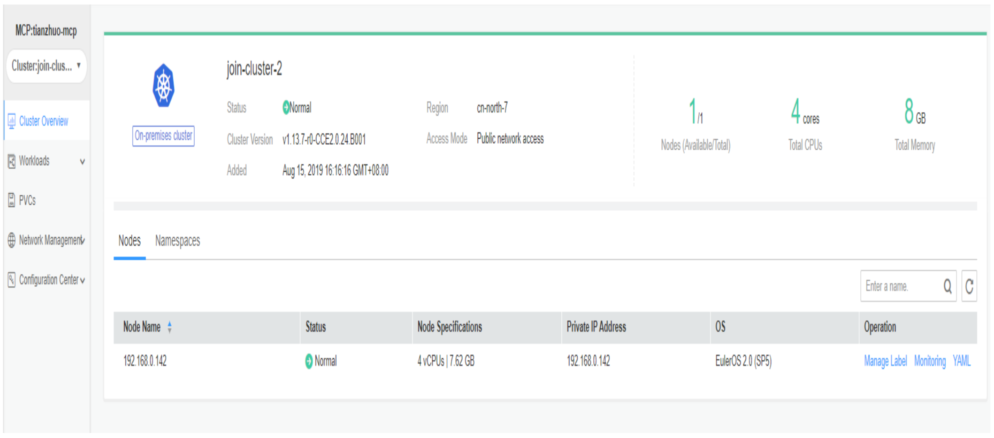
After a cluster is added to MCP, you can access the cluster console from MCP to view node information in the cluster.

Procedure

Step 1 Log in to the console of the corresponding cluster.

Step 2 In the navigation pane, choose **Cluster Overview**. On the **Nodes** tab page, view the node information in the cluster.

Figure 9-1 Viewing node information in a cluster



join-cluster-2

Status: Normal
Cluster Version: v1.13.7-r0-CCE2.0.24.B001
Added: Aug 15, 2019 16:16:16 GMT+08:00
Region: cn-north-7
Access Mode: Public network access

1 /1 Nodes (Available/Total)
4 cores Total CPUs
8 GB Total Memory

Node Name	Status	Node Specifications	Private IP Address	OS	Operation
192.168.0.142	Normal	4 vCPUs | 7.62 GB	192.168.0.142	EulerOS 2.0 (SP5)	Manage Label Monitoring YAML

----End

9.2 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 Scenarios

Node labels are mainly used in the following scenarios:

- Node management: Node labels are used to classify nodes.
- Affinity and anti-affinity between workloads and nodes:
  - Different workloads have different resource requirements such as CPU, memory, and I/O. If a workload consumes too many resources in a cluster, other workloads in the same cluster may fail to run properly. In this case, you are advised to add different labels to nodes. When deploying a workload, you can configure node affinity and anti-affinity based on node labels.
  - A system can be divided into multiple modules. Each module consists of multiple microservices. To ensure efficient O&M, you can add a module label to each node so that each module can be deployed on the corresponding node. In this way, modules do not interfere with each other and microservices can be easily maintained on their nodes.

Inherent Node Labels

After a node is created, some inherent labels are generated for the node. These labels cannot be edited or deleted. **Table 9-1** describes these labels.

Table 9-1 Inherent labels of a node

Key	Value
failure-domain.beta.kubernetes.io/region	Indicates the region where the node is located.
failure-domain.beta.kubernetes.io/zone	Indicates the AZ where the node is located.
beta.kubernetes.io/arch	Indicates the processor architecture of the node. For example, **amd64** indicates a AMD64-bit processor.
beta.kubernetes.io/os	Indicates the operating system of the node. For example, **linux** indicates that the node uses Linux as its operating system.
kubernetes.io/availablezone	Indicates the AZ where the node is located.
kubernetes.io/hostname	Indicates the host name of the node.
os.architecture	Indicates the processor architecture of the node. For example, **amd64** indicates a AMD64-bit processor.

Key	Value
os.name	Indicates the operating system name of the node. For example, **EulerOS_2.0_SP2** indicates that the node uses EulerOS 2.2 as its operating system.
os.version	Indicates the kernel version of the node.

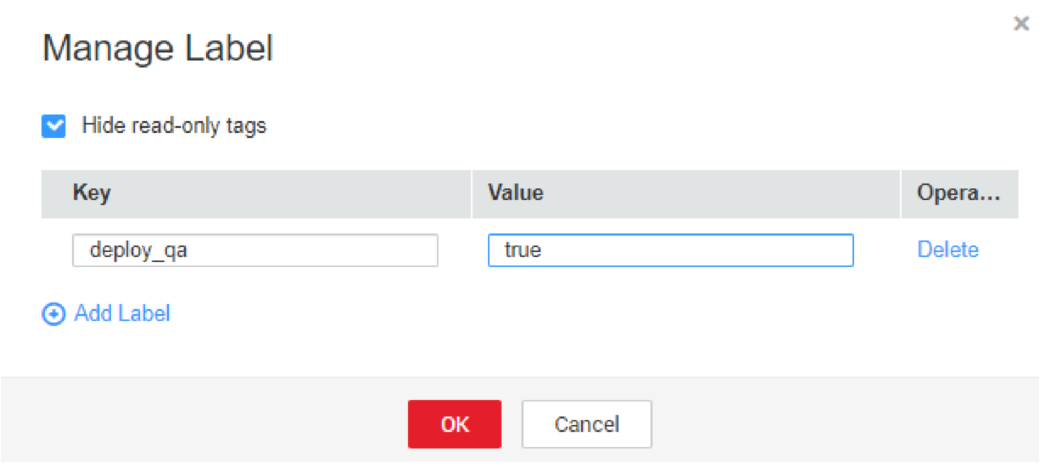
Adding a Node Label

Step 1 Log in to the cluster console. In the navigation pane, choose **Cluster Overview**. On the **Nodes** tab page, click **Manage Label** in the row where the node to be added with labels resides.

Step 2 Click **Add Label**, enter a key and value, and click **OK**.

As shown in the following figure, enter **deploy_qa** for **Key** and **true** for **Value**, and click **OK**.

Figure 9-2 Adding a label


Manage Label

Hide read-only tags

Key	Value	Opera...
deploy_qa	true	Delete

----End

Deleting a Node Label

Only labels added by users can be deleted. Those inherent node labels cannot be deleted.

Step 1 Log in to the cluster console. In the navigation pane, choose **Cluster Overview**. On the **Nodes** tab page, click **Manage Label** in the row where the node whose labels are to be deleted resides.

Step 2 Click **Delete** in the row where the label to be deleted resides, and then click **OK**.

----End

9.3 Managing Namespaces

Namespaces created on the cluster console apply only to the current cluster. You can create workloads and jobs in these namespaces. In addition, you can manage resource quotas in namespaces or delete namespaces. However, all operations apply only to the current cluster.

- The **default** namespace created by MCP supports quota management but cannot be deleted.
- Namespaces created by a cluster, such as **karmada-cluster**, **kube-public**, and **kube-system**, do not support quota management and cannot be deleted.

Namespace Types

Namespaces are classified into four types based on creation types:

- Default namespace created by MCP, namely, **default**. The **default** namespace is used if no namespace is specified.
- Default namespaces created by a cluster, namely, **kube-public**, **kube-system**, and **karmada-cluster**.
  – **kube-public**: used to deploy public add-ons and container templates.
  – **kube-system**: used to deploy Kubernetes system components.
  – **karmada-cluster**: used to deploy system components.
- Namespaces created by users on the MCP console.
- Namespaces created by users on the cluster console. These namespaces apply only to the current cluster.

Creating a Namespace

Step 1 Log in to the cluster console. In the left navigation pane, choose **Cluster Overview**.

Step 2 On the page displayed, click **Namespaces** and then **Create Namespace**. Configure parameters.

- **Namespace Name**: Name of the namespace, which must be unique in a cluster.
- **Cluster**: Name of the current cluster, which cannot be changed.
- **Description**: Description of the namespace.
- **Manage Quota**: Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace.
  – After this option is enabled, you can set resource quotas on this page.
  – If you do not enable this option, you need to click **Manage Quota** in the namespace list to set resource quotas after the namespace is created.

Step 3 Click **OK**.

----End

Deleting a Namespace

On the cluster console, you can delete only the namespaces created for the current cluster. If a namespace created on the MCP console is deleted on the cluster console, it will be re-created.

Deleting a namespace will delete all data resources related to the namespace. Exercise caution when performing this operation.

Step 1 Log in to the cluster console. In the left navigation pane, choose **Cluster Overview**. On the page displayed, click **Namespaces**.

Step 2 In the row where the namespace to be deleted resides, click **Delete**.

----End

Setting Resource Quotas in a Namespace

Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace.

Namespace-level resource quotas limit the amount of resources available to teams or users when these teams or users use the same cluster. The quotas include the total number of a type of objects and the total amount of compute resources (CPU and memory) consumed by the objects.

Step 1 Log in to the cluster console. In the left navigation pane, choose **Cluster Overview**. Then click **Namespaces**.

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

The **kube-public**, **kube-system**, and **karmada-cluster** namespaces do not support resource quota settings.

Step 3 Click **Edit Quota**, set the resource quotas, and click **OK**.

NOTICE

- The values of resource quotas are integers. If the quota of a resource is set to **0**, no limit is posed on the resource. If you want to limit the CPU or memory quota, you must specify the CPU or memory request value when creating a workload.
- Accumulated quota usage includes the default resources created by the system, such as the Kubernetes Service (view this Service using the kubectl tool) created in the **default** namespace. Therefore, you are advised to set a resource quota greater than what you expect.

- **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 jobs that can be created in 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.

----End

9.4 Workload Management

9.4.1 Creating a Workload

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

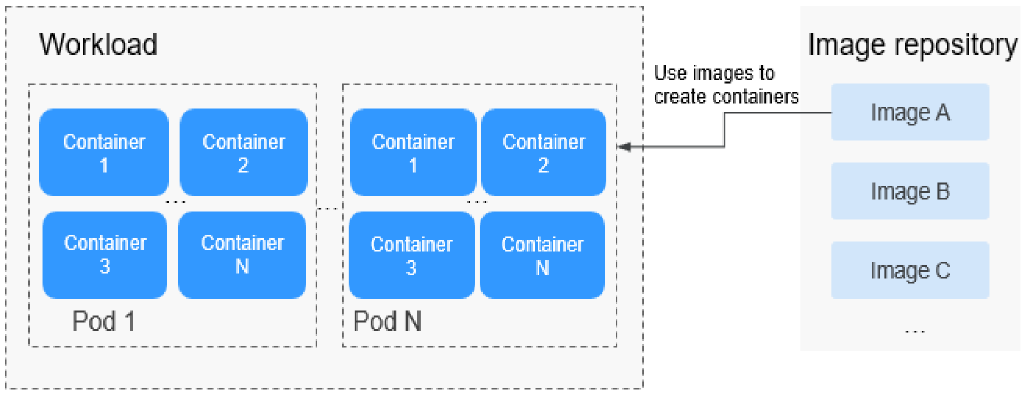
Basic Concepts

- Deployments: Pods are completely independent of each other and functionally identical. They feature auto scaling and rolling upgrade. For example, Nginx and WordPress. For details on how to create a Deployment, see **Creating a Deployment**.
- StatefulSets: Pods are not completely independent of each other. They have stable persistent storage and network identifiers, and feature orderly deployment, scale-in, and deletion. For example, MySQL-HA and etcd. For details on how to create a StatefulSet, see **Creating a StatefulSet**.

Relationship Between Workloads and Containers

As shown in **Figure 9-3**, a workload controls one or more 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 9-3 Relationship between workloads and containers



Workload Lifecycle

Table 9-2 Status description

Status	Description
Running	All pods are running.
Unready	All pods are in the pending state.
Upgrading	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.
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.

Creating a Deployment

Step 1 (Optional) If you create a workload using **My Images**, first push the image to SWR. For details about how to push an image, see **Image Management** If you create a workload from an open source image, you do not need to upload the image.

Step 2 Log in to the cluster console. In the navigation pane, choose **Workloads** > **Deployments**. On the page displayed, click **Create Deployment**.

Step 3 Set basic workload parameters as described in **Table 9-3**. The parameters marked with asterisks (*) are mandatory.

Table 9-3 Basic workload parameters

Parameter	Description
*Name	Name of a workload, which must be unique.
*Cluster	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.

Parameter	Description
*Pods	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. Configuring multiple pods for a workload ensures that the workload can still run properly even if a pod is faulty. If only one pod is used, a node or pod exception may cause service exceptions.
Description	Description of the workload.

Step 4 Click **Next** to add a container.

1. Click **Add Container** and select the image to be deployed.
   - **My Images**: Create a workload using an image in the Huawei Cloud image repository. If no image is available, click **push an image** to push an image.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
     - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
   - **Open Source Images**: Create a workload using an official image in the open source image repository.
     - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
     - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
   - **Third-Party Images**: Create a workload using an image from any third-party image repository (image repositories other than Huawei Cloud SWR and open source image repository). Ensure that the node where the workload is running is accessible from public networks.
     - 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 authentication (using account and password), select the corresponding secret, enter an image address, and then click **OK**.
2. Configure image parameters.

Table 9-4 Image parameters

Parameter	Description
Image Name	Name of the image. You can click **Change Image** to select another image.
Image Tag	Tag of the image to be deployed.
Container Name	Container name, which can be changed.
Container Specifications	Resource amount that users can apply for and the maximum amount of resources that are available to users. For details, see **Setting Container Specifications**. – **Request**: Minimum amount of resources required for running a container. – **Limit**: If a container is overloaded, the system may be faulty. To avoid this situation, set the maximum limits for the container resource quotas to ensure that container resources do not exceed the limits.

3. Configure container lifecycle, that is, set the commands that need to be executed in each phase during container management.
   - **Startup Command**: executed when the container is started.
   - **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 is stopped. For details, see **Setting Container Lifecycle Parameters**.
4. Set the health check function that checks 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**: The probe restarts the workload when detecting that the workload pod is unhealthy.
   - **Readiness Probe**: The probe sets the workload to the unready state when detecting that the workload pod is unhealthy. In this way, the service traffic will not be directed to the workload pod.
5. Set environment variables. Environment variables can be added to a container. In general, environment variables are used to set parameters.

   In the **Environment Variables** area, click **Add Environment Variable**.
6. 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.
7. Mount data storage to containers for persistent storage and high disk I/O. For details, see **Federated PVCs**.

NOTE

After the workload is created, you can access the workload details page of the cluster console, click the **Container Settings** tab, and click **Modify Container Settings** to modify the container settings.

Step 5 Click **Next**. Then, click **Create 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**.

- 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**.

- NodePort

  A workload can be accessed by other workloads in the same VPC using the IP address of a cluster node. The NodePort access type is applicable to the scenario in which other workloads in the same VPC in the cloud need to access the workload in the Kubernetes cluster.

- LoadBalancer

  A workload can be accessed from a public network 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**.

Step 6 Click **Next** and configure advanced settings.

- **Upgrade Policy**: **In-place upgrade** and **Rolling upgrade** are available.
  - **In-place upgrade**: Old pods are deleted before new pods are created. Services will be interrupted during an in-place upgrade.
  - **Rolling upgrade**: An old pod is gradually replaced with a new pod. During the upgrade, service traffic is evenly distributed to the 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
- **Graceful Deletion**: Set a time window (0–9,999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
- **Migration Policy**: Set a time window for the system to schedule the workload pod to another available node when the node where the workload pod is running is unavailable. The default value is 300s.
- **Scheduling Policies**: You can combine static global scheduling policies or dynamic runtime scheduling policies as required.

- **Pod Label**: System-defined app labels added during workload creation cannot be modified. You can click **Add Label** to add labels.

Step 7 After the configuration is complete, click **Create**. You can view the Deployment status in the Deployment list.

If the Deployment is in the **Running** state, the Deployment is successfully created.

----End

Creating a StatefulSet

Step 1 (Optional) If you create a workload using **My Images**, first push the image to SWR. For details about how to push an image, see **Image Management** If you create a workload from an open source image, you do not need to upload the image.

Step 2 Log in to the cluster console. In the navigation pane, choose **Workloads** > **StatefulSets**. On the page displayed, click **Create StatefulSet**.

Step 3 Set basic workload parameters as described in **Table 9-5**. The parameters marked with asterisks (*) are mandatory.

Table 9-5 Basic workload parameters

Parameter	Description
*Name	Name of a workload, which must be unique.
*Cluster	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.
*Pods	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. Configuring multiple pods for a workload ensures that the workload can still run properly even if a pod is faulty. If only one pod is used, a node or pod exception may cause service exceptions.
Description	Description of the workload.

Step 4 Click **Next**, and add a container.

1. Click **Add Container** and select the image to be deployed.
   - **My Images**: Create a workload using an image in the Huawei Cloud image repository. If no image is available, click **push an image** to push an image.

    - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
    - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
  - **Open Source Images**: Create a workload using an official image in the open source image repository.
    - If your image repository does not require authentication, set **Secret Authentication** to **No**, select the corresponding image, and then click **OK**.
    - If your image repository requires authentication (using account and password), select the corresponding secret, and then click **OK**. If no secret is available, create a secret by following the procedure described in **Creating a Secret**.
  - **Third-Party Images**: Create a workload using an image from any third-party image repository (image repositories other than Huawei Cloud SWR and open source image repository). Ensure that the node where the workload is running is accessible from public networks.
    - 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 authentication (using account and password), select the corresponding secret, enter an image address, and then click **OK**.

2. Configure image parameters.

Table 9-6 Image parameters

Parameter	Description
Image Name	Name of the image. You can click **Change Image** to select another image.
Image Tag	Tag of the image to be deployed.
Container Name	Container name, which can be changed.
Container Specifications	Resource amount that users can apply for and the maximum amount of resources that are available to users. For details, see **Setting Container Specifications**. – **Request**: Minimum amount of resources required for running a container. – **Limit**: If a container is overloaded, the system may be faulty. To avoid this situation, set the maximum limits for the container resource quotas to ensure that container resources do not exceed the limits.

3. Configure container lifecycle, that is, set the commands that need to be executed in each phase during container management.
   – **Startup Command**: executed when the container is started.
   – **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 is stopped. For details, see **Setting Container Lifecycle Parameters**.
4. Set the health check function that checks 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**: The probe restarts the workload when detecting that the workload pod is unhealthy.
   – **Readiness Probe**: The probe sets the workload to the unready state when detecting that the workload pod is unhealthy. In this way, the service traffic will not be directed to the workload pod.
5. Set environment variables. Environment variables can be added to a container. In general, environment variables are used to set parameters.

   In the **Environment Variables** area, click **Add Environment Variable**.
6. 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.
7. Mount data storage to containers for persistent storage and high disk I/O. For details, see **Federated PVCs**.

NOTE

After the workload is created, you can access the workload details page of the cluster console, click the **Container Settings** tab, and click **Modify Container Settings** to modify the container settings.

Step 5 Click **Next**, and set headless Service parameters, as shown in **Table 9-7**.

Table 9-7 Headless Service parameters

Parameter	Description
Service Name	Name of the Service corresponding to the workload for mutual access between workloads in the same cluster. This Service is used for internal discovery of pods, and does not require an independent IP address or load balancing.
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 6 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**.

- 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**.
- NodePort

  A workload can be accessed by other workloads in the same VPC using the IP address of a cluster node. The NodePort access type is applicable to the scenario in which other workloads in the same VPC in the cloud need to access the workload in the Kubernetes cluster.
- LoadBalancer

  A workload can be accessed from a public network 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**.

Step 7 Click **Next** and configure advanced settings.

- **Upgrade Policy**: Only **Rolling upgrade** is supported.

  NOTE

  During a rolling upgrade, old pods are gradually replaced with new ones, and service traffic is evenly distributed to both old and new pods to ensure service continuity.
- **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 continues only after the previous pod is ready or deleted.) This is the default policy.

  Parallel: The StatefulSet will create pods in parallel to match the desired scale without waiting, and will delete all pods at once.
- **Graceful Deletion**: Set a time window (0–9,999s) for pre-stop commands to finish execution before a workload is deleted. The default value is 30s.
- **Migration Policy**: Set a time window for the system to schedule the workload pod to another available node when the node where the workload pod is running is unavailable. The default value is 300s.
- **Scheduling Policies**: You can combine static global scheduling policies or dynamic runtime scheduling policies as required.
- **Pod Label**: System-defined app labels added during workload creation cannot be modified. You can click **Add Label** to add labels.

Step 8 After the configuration is complete, click **Create**. You can view the StatefulSet status in the StatefulSet list.

If the StatefulSet is in the **Running** state, the StatefulSet is successfully created.

----End

9.5 Creating a Job

Job Overview

In Kubernetes, there are two types of jobs: one-off jobs and cron jobs.

A job (one-off job) is a resource object that Kubernetes uses to control batch tasks. Jobs are different from long-term servo tasks (such as Deployments and StatefulSets). The former are started and terminated at specific times, while the latter run 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 depending on the **spec.completions** policy. A single-pod job is considered successful if one pod completes successfully. A job with a fixed success count is considered successful if N pods complete successfully. A queue job is considered successful based on the global success confirmed by the application.

Similar to the Cron tab in Linux OS, a cron job can:

- 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.

Creating a Job

A job runs pods that perform a completable task. The pods automatically exit after successfully completing the task. Before creating a workload, you can run a job to upload an image to the image repository.

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

Step 2 Log in to the cluster console. In the navigation pane, choose **Workloads > Jobs**. On the page displayed, click **Create Job**.

Step 3 Configure parameters as prompted. For details, see **Creating a Job**.

Step 4 After the job is created, you can view the job in the job list.

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

----End

Creating a Cron Job

A cron job can run a scheduled job once or periodically at the specified time. The job automatically exits after successfully completing the task. For example, you can perform time synchronization for all active nodes at the specified time.

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

Step 2 Log in to the cluster console. In the navigation pane, choose **Workloads > Cron Jobs**. On the page displayed, click **Create Cron Job**.

Step 3 Configure parameters as prompted. For details, see **Creating a Cron Job**.

Step 4 After the cron job is created, you can view the cron job in the cron job list.

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

----End

9.6 Pod

A pod is the smallest and simplest unit in the Kubernetes object model that you create or deploy. A pod encapsulates an application's container (or, in some cases, multiple containers), storage resources, a unique network identity (IP address), as well as options that govern how the container(s) should run. A pod represents a single instance of an application in Kubernetes, which might consist of either a single container or a small number of containers that are tightly coupled and that share resources.

Creating a Pod from a YAML File

Step 1 Log in to the cluster console. Choose **Workloads** > **Pods**, and click **Create Using YAML**.

Step 2 On the **Create Pod Using YAML** page , edit the YAML file.

Step 3 Click **Create**.

----End

9.7 Configuration Center

9.7.1 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.

ConfigMaps provide the following benefits:

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

Creating a ConfigMap

Step 1 Log in to the cluster console. In the left navigation pane, choose **Configuration Center** > **ConfigMaps**. You can create a ConfigMap directly or using YAML. If you want to create a ConfigMap using YAML, go to **Step 3**.

Step 2 Create a ConfigMap directly by clicking **Create ConfigMap**.

Set the parameters listed in **Table 9-8**. The parameters marked with asterisks (*) are mandatory.

Table 9-8 Parameters for creating a ConfigMap

Parameter	Description
*Name	Name of a ConfigMap, which must be unique in a namespace.
*Cluster	Select the cluster for which you want to create a ConfigMap.
*Namespace	Namespace to which the ConfigMap belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of the ConfigMap.
Data	The workload configuration data can be used in a container. **Key** specifies the file name, and **Value** specifies the file content. 1. Click **Add Data**. 2. Set **Key** and **Value**.
Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define identified attributes of these objects and can be used to manage and select objects. 1. Click **Add Label**. 2. Set **Key** and **Value**.

Step 3 Create a ConfigMap using a YAML file by clicking **Create Using YAML**.

NOTE

To create a resource by uploading a file, ensure that the resource description file has been created. MCP supports files in JSON or YAML format. For details, see **ConfigMap Resource File Configuration**.

You can import or directly write the file content in YAML or JSON format.

- Method 1: Import an orchestration file.

  Click **Upload YAML File** to import a YAML or JSON file. The content of the YAML or JSON file is displayed in the orchestration content area.

- Method 2: Directly orchestrate the content.

  In the orchestration content area, enter the content of the YAML or JSON file.

Step 4 Click **Create** after the configuration is complete.

The new ConfigMap is displayed in the ConfigMap list.

----End

ConfigMap Resource File Configuration

A ConfigMap resource file can be in JSON or YAML format, and the file size cannot exceed 2 MB.

- JSON format

  The file name is **configmap.json** and the configuration example is as follows:

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "paas-broker-app-017",
    "namespace": "test"
  },
  "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 file name is **configmap.yaml** and the configuration example is as follows:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
data:
  data-1: value-1
  data-2: value-2
```

Related Operations

After creating a ConfigMap, you can perform the operations described in **Table 9-9**.

Table 9-9 Other operations

Operation	Description
Viewing a YAML file	Click **View YAML** in the row where the target ConfigMap resides to view its YAML file.

Operation	Description
Updating a ConfigMap	1. Click **Update** in the row where the target ConfigMap resides. 2. Modify the ConfigMap data according to **Table 9-8**. 3. Click **Update**.
Deleting a ConfigMap	Click **Delete** in the row where the target ConfigMap resides. Delete the ConfigMap as prompted.

9.7.2 Creating a Secret

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

Creating a Secret

Step 1 Log in to the cluster console. In the left navigation pane, choose **Configuration Center** > **Secrets**. You can create a secret directly or using YAML. If you want to create a secret using YAML, go to **Step 3**.

Step 2 Click **Create Secret**.

Set the parameters listed in **Table 9-10**. The parameters marked with asterisks (*) are mandatory.

Table 9-10 Parameters for creating a secret

Parameter	Description
*Name	Name of a secret, which must be unique in a namespace.
*Cluster	Select the cluster for which you want to create a secret.
*Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of the secret.

Parameter	Description
*Type	Type of the secret you create. ● Opaque: common secret. ● kubernetes.io/dockerconfigjson: a secret that stores the authentication information required for pulling images from a private repository. ● IngressTLS: a secret that stores the certificate required by ingresses (layer-7 load balancing Services). ● Other: another type of secret, which is specified manually.
*Data	Workload secret data can be used in containers. ● If the secret is of the Opaque type: 1. Click **Add Data**. 2. Set **Key** and **Value**. The value must be encoded using Base64. For details on Base64 encoding, see **Base64 Encoding**. ● If the secret is of the kubernetes.io/dockerconfigjson type, enter the username and password of a private image repository. ● If the secret is of the IngressTLS type, upload a certificate file and a private key file.
Label	Labels are attached to objects such as workloads, nodes, and Services in key-value pairs. Labels define identified attributes of these objects and can be used to manage and select objects. 1. Click **Add Label**. 2. Set **Key** and **Value**.

Step 3 Create a secret using a YAML file by clicking **Create Using YAML**.

NOTE

To create a resource by uploading a file, ensure that the resource description file has been created. MCP supports files in JSON or YAML format. For details, 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 **Upload YAML File** to import a YAML or JSON file. The content of the YAML or JSON file is displayed in the orchestration content area.
- Method 2: Directly orchestrate the content.

  In the orchestration content area, enter the content of the YAML or JSON file.

Step 4 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: bXktdXNlcm5hbWUK  #Username, which must be encoded using Base64.
  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": "bXktdXNlcm5hbWUK",
    "password": "******"
  },
  "type": "Opaque"
}
```

Related Operations

After a secret is created, you can perform the operations described in **Table 9-11**.

NOTE

The secret list contains system-defined secrets that can only be viewed but cannot be updated or deleted.

Table 9-11 Other operations

Operation	Description
Viewing a YAML file	Click **View YAML** in the row where the target secret resides to view its YAML file.

Operation	Description
Updating a secret	1. Click **Update** in the row where the target secret resides. 2. Modify the secret data according to **Table 9-10**. 3. Click **Update**.
Deleting a 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 secrets as prompted.

Base64 Encoding

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

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