Ubiquitous Cloud Native Service

User Guide

Issue 01

Date 2023-08-30

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

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 UCS Clusters

1.1 Overview

UCS supports unified management of clusters across clouds and regions. The following types of clusters are supported:

- **Huawei Cloud clusters**: Huawei Cloud CCE clusters and CCE Turbo clusters
- **On-premises clusters**: Kubernetes clusters that are provisioned by UCS but running on your on-premises data center, such as UCS on Bare Metal and UCS on VMware
- **Attached clusters**: Third-party Kubernetes clusters that comply with the Cloud Native Computing Foundation (CNCF) standard, such as AWS EKS clusters, Google Cloud GKE clusters, and Kubernetes clusters that are deployed and run by third parties
- **Multi-cloud clusters**: Kubernetes clusters that are provisioned by UCS but running on the platform of other cloud service providers, such as UCS on AWS and UCS on Azure

CAUTION

If a cluster contains nodes with ultra large compute capacity and you do not want them to be counted in the CPU and memory allocation rate metrics in the cluster list on the UCS console, add the **type:virtual-kubelet** label to the nodes so that you can accurately identify cluster resource allocation. For details about how to label nodes, see **Adding Labels/Taints to Nodes**.

1.2 Huawei Cloud Clusters

You can register Huawei Cloud clusters (CCE standard and CCE Turbo clusters) with UCS with just a few clicks. After the registration is complete, clusters can be managed centrally.

Constraints

- Only **Huawei Cloud accounts** or users with the **UCS FullAccess** permission can register Huawei Cloud clusters.
- If you are connecting a cluster outside the Chinese mainland to UCS, the connection and the subsequent actions you will take must comply with local laws and regulations.
- Registered Kubernetes clusters must be between v1.19 and v1.28.

Prerequisites

You have created a CCE standard cluster or CCE Turbo cluster to be connected to UCS, and the cluster is in the **Running** state.

Procedure

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 In the **Huawei Cloud cluster** card view, click **Register Cluster**.

Step 3 Select the CCE cluster, select a fleet, and click **OK**.

If you do not select a fleet when registering a cluster, the cluster will be displayed on the **Clusters Not in Fleet** tab after registration. You can add it to a fleet later. For details, see **Managing Clusters Not in the Fleet**.

NOTE

When registering a cluster, you cannot select a fleet with cluster federation enabled. To add your cluster to the fleet with cluster federation enabled, register your cluster with UCS first. For details about cluster federation, see **Enabling Cluster Federation**.

----End

1.3 On-Premises Clusters

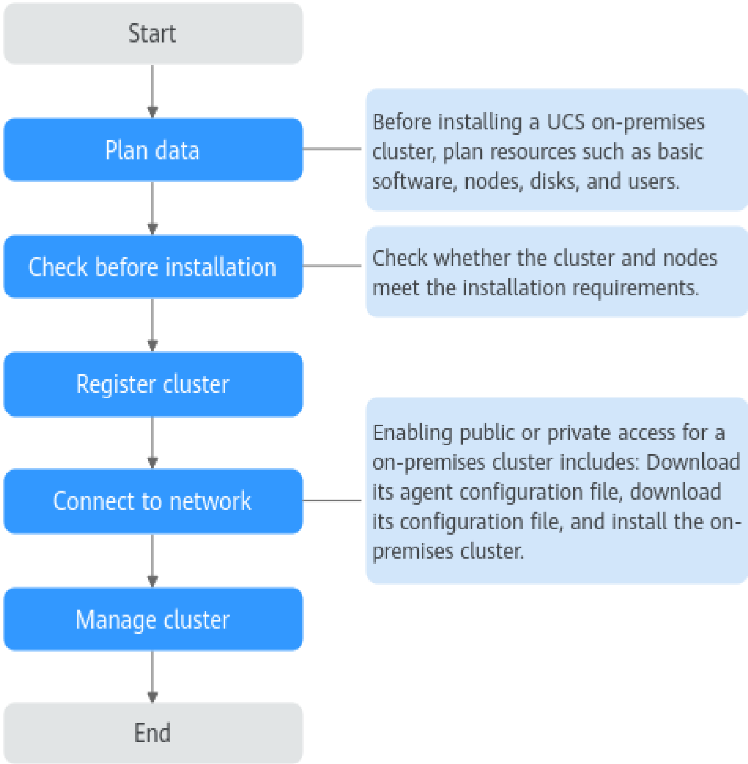
1.3.1 Overview

On-premises clusters refer to Kubernetes clusters that are provisioned by UCS but running on your on-premises data center. You only need to prepare the required physical resources. The cloud platform will be responsible for installing Kubernetes software and connecting your clusters to UCS.

On-premises clusters are compatible with multiple underlying infrastructures. They can be deployed on virtualized types such as BMS and VMware. The container network can be connected to the underlying network. CSI can be used to connect to multiple underlying storage services (such as VMware vSphere) to provide persistent storage.

Figure 1-1 shows the on-premises cluster management process.

Figure 1-1 On-premises cluster management process



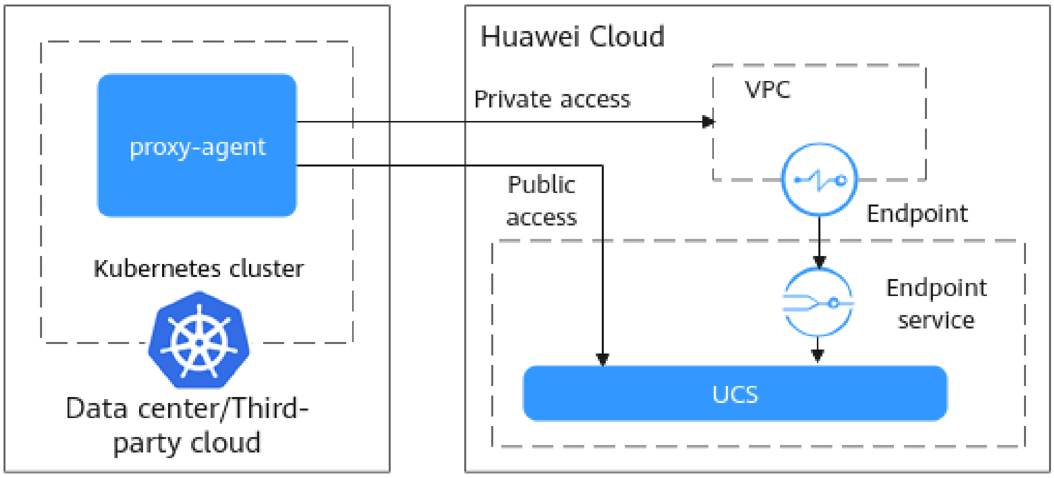
Access Mode

UCS uses the cluster network agent to connect to clusters, as shown in **Figure 1-2**. You do not need to enable any inbound port on the firewall. Instead, only the cluster agent program is required to establish sessions with UCS in the outbound direction.

There are two methods with different advantages for on-premises clusters to connect to UCS:

- Over a public network: flexibility, cost-effectiveness, and easy access
- Over a private network: high speed, low latency, stability, and security

Figure 1-2 How clusters are connected to UCS



1.3.2 Service Planning for On-Premises Cluster Installation

1.3.2.1 Basic Software Planning

Basic software of the on-premises nodes must meet the requirements listed in **Table 1-1**.

Table 1-1 Basic software planning

System Architecture	OS Type	Network Model	OS Version	Kernel Version
x86	Ubuntu 22.04	Cilium	Run **cat /etc/lsb-release** to check the version. DISTRIB_DESCRIPTION="Ubuntu 22.04.1 LTS"	Run **uname -r** to check the version. 5.10.0-46-generic or later
	Red Hat 8.6	Cilium	Run **cat /etc/os-release** to check the version. Red Hat Enterprise Linux release 8.6 (Ootpa)	Run **uname -r** to check the version. 4.18.0-372.9.1.el8.x86_64
	HCE OS 2.0	Cilium	Run **cat /etc/os-release** to check the version. Huawei Cloud EulerOS release 2.0 (West Lake)	Run **uname -r** to check the version. 5.10.0-60.18.0.50.r865_35.hce2.x86_64

NOTE

- Cilium is a networking solution that supports network protocols such as BGP and eBPF. For details, see **Cilium official documentation**.
- Huawei Cloud EulerOS is a Linux OS developed based on openEuler to provide a cloud-native, high-performance, secure, and stable environment for developing and running applications. Huawei Cloud EulerOS supports hardware architectures such as x86 and Arm (64-bit). To install Huawei Cloud EulerOS 2.0, submit a service ticket to get technical support.

1.3.2.2 Data Planning

Firewalls

Firewalls must meet the requirements listed in the **Table 1-2**.

Table 1-2 Firewalls

Source Device	Source IP Address	Source Port	Target Device	Target IP Address	Destination Port (Listening)	Protocol	Port Description	Listening Port Configurable	Authentication Mode	Encryption Mode
ucsctl executors	IP address of each ucsctl executor	All	All nodes	IP address of each node	22	TCP	SSH	No	Certificate/Username and password	TLS 1.2
All nodes	IP address of each node	All	NTP server	IP address of the NTP server	123	UDP	NTP	No	None	None
All nodes	IP address of each node	All	DNS server	IP address of the DNS server	53	UDP	DNS	No	None	None
All nodes	IP address of each node	All	Self-built apt sources	IP address of each apt source	80/443	TCP	HTTP	No	None	None

Source Device	Source IP Address	Source Port	Target Device	Target IP Address	Destination Port (Listening)	Protocol	Port Description	Listening Port Configurable	Authentication Mode	Encryption Mode
All nodes	IP address of each node	All	Load balancer or virtual IP address	IP address of the load balancer or virtual IP address bound to the nodes	5443	TCP	kube-apiserver	No	HTTPS and certificate	TLS 1.2
All nodes	IP address of each node	1024-65535	All nodes	IP address of each node	1024-65535	All	None	No	None	None
All nodes	IP address of each node	All	All nodes	IP address of each node	8472	UDP	VXLAN	No	None	None
Nodes that need to access the Ingress	IP address of each node that needs to access the Ingress	All	Network nodes	IP address of each network node	80, 443, or a specified port	TCP	HTTP	No	HTTPS and certificate	TLS 1.2

Source Device	Source IP Address	Source Port	Target Device	Target IP Address	Destination Port (Listening)	Protocol	Port Description	Listening Port Configurable	Authentication Mode	Encryption Mode
All nodes	IP address of each node	All	Three master nodes	IP address of each master node	5444	TCP	kube-apiserver	No	HTTPS and certificate	TLS 1.2
ucsctl executors	IP address of each ucsctl executor	All	Huawei Cloud Object Storage Service (OBS)	IP address of the OBS endpoint	443	TCP	HTTP	No	HTTPS and certificate	TLS 1.2
Three master nodes	IP address of each master node	All	UCS	124.70.21.61 proxyurl.ucs.myhuaweicloud.com	30123	TCP	gRPC	No	HTTPS and certificate	TLS 1.2
Three master nodes	IP address of each master node	All	Identity and Access Management (IAM)	Domain name used by external systems to access IAM	443	TCP	HTTP	No	HTTPS and certificate	TLS 1.2

Source Device	Source IP Address	Source Port	Target Device	Target IP Address	Destination Port (Listening)	Protocol	Port Description	Listening Port Configurable	Authentication Mode	Encryption Mode
All nodes	IP address of each node	All	SoftWare Repository for Container (SWR)	IP address of the SWR endpoint	443	TCP	HTTP	No	HTTPS and certificate	TLS 1.2
All nodes	IP address of each node	All	Official Ubuntu repositories/Proxy repositories in China	IP address of each repository	80/443	TCP	HTTP	No	None	None
Monitoring nodes	IP address of each monitoring node	All	Application Operations Management (AOM)	IP address mapping a domain name	443	TCP	HTTP	No	HTTPS and certificate	TLS 1.2
Monitoring nodes	IP address of each monitoring node	All	Log Tank Service (LTS)	IP address mapping a domain name	443	TCP	HTTP	No	HTTPS and certificate	TLS 1.2

Resource Specifications

UCS on-premises clusters are installed in HA mode to meet DR requirements for commercial use. The following tables list resource specifications.

Table 1-3 Resource specifications for basic container platform capabilities

Node Type	Quantity	CPU (vCPUs)	Memory (GiB)	System Disk (GiB)	High-Performance Disk (GiB)	Data Disk (GiB)	Remarks
Cluster manage nodes	3	8	16	100	50	300	A virtual IP address is required for HA.
Cluster compute nodes	As required	2	4	40	-	100	You can increase the number of nodes as required.

Table 1-4 Resource specifications for Container Intelligent Analysis (CIA) nodes

Node Type	CPU (vCPUs)	Memory (GiB)
prometheus node	Requests: 1 Limits: 4	Requests: 2 Limits: 12
log-agent node	Requests: 0.5 Limits: 3	Requests: 1.5 Limits: 2.5

Table 1-5 Resource specifications for Operator Service Center (OSC) compute nodes

Node Type	Quantity	CPU (vCPUs)	Memory (GiB)	System Disk (GiB)	High-Performance Disk (GiB)	Data Disk (GiB)
operator-chef	1	Requests: 0.5 Limits: 2	Requests: 0.5 Limits: 2	N/A	N/A	10 (for storing logs)
helm-operator	1	Requests: 0.3 Limits: 1.5	Requests: 0.3 Limits: 1.5	N/A	N/A	10 (for storing logs)

Node Type	Quantity	CPU (vCPUs)	Memory (GiB)	System Disk (GiB)	High-Performance Disk (GiB)	Data Disk (GiB)
ops-operator	1	Requests: 0.3 Limits: 1.5	Requests: 0.3 Limits: 1.5	N/A	N/A	10 (for storing logs)

External Dependencies

Table 1-6 External dependencies

Dependency	Function
DNS server	The DNS server can resolve the domain names of services such as OBS, SWR, IAM, and DNS. For details about the domain names, see **Regions and Endpoints**. If a node is accessed over a public network, the node can automatically identify the default DNS settings. You only need to configure a public upstream DNS server in advance. If a node is accessed over a private network, the node cannot identify the default DNS settings. You need to configure the DNS resolution for VPC endpoints in advance. For details, see Preparations. If you have not set up a DNS server, set up it by referring to **DNS**.
Apt source	An apt source provides dependency packages for adding nodes to on-premises clusters if required by some components such as NTP.
NTP server	(Optional) The NTP server is used for time synchronization between nodes in a cluster. An external NTP server is recommended.

Disk Volumes

Table 1-7 Disk volumes

Node Type	Disk Mount Point	Available Size (GiB)	Used For
Cluster manage nodes	/var/lib/containerd	50	Directory for storing containerd images
	/run/containerd	30	Directory for storing container runtimes

Node Type	Disk Mount Point	Available Size (GiB)	Used For
	/var/paas/run	50	Directory for storing etcd data (SSDs are recommended.)
	/var/paas/sys/log	20	Directory for storing logs
	/mnt/paas	40	Directory where volumes are mounted when containers are running.
	/tmp	20	Directory for storing temporary files
Cluster compute nodes	/var/lib/containerd	100	Directory for storing containerd images
	/run/containerd	50	Directory for storing container runtimes
	/mnt/paas	50	Directory where volumes are mounted when containers are running.

Load Balancing

If master nodes in an on-premises cluster are deployed in HA mode for DR, a unified IP address is required for the access from cluster compute nodes and other external services. There are two ways to provide access: virtual IP address and load balancer.

- IP addresses

  An idle IP address must be planned as a virtual IP address that can be shared by the three master nodes. The virtual IP address is randomly bound to a master node. When the node becomes abnormal, the virtual IP address is automatically switched to another node to ensure HA.

  Table 1-8 IP addresses

IP Type	IP Address	Used For
Virtual IP address	10.10.11.10 (example)	An IP address used for HA. Plan the IP address based on site requirements.

- Load balancers

  If you have an external load balancer, on-premises clusters can connect to it for HA. Configurations are as follows:

  – Listeners: 3 TCP listeners with three different ports (80, 443, and 5443)

– Backend server groups: 3 TCP backend server groups with three different ports (corresponding to ports 80, 443, and 5444 of the three master nodes)

Table 1-9 lists the requirements for the TCP backend server groups associated with the listeners.

Table 1-9 Listeners and TCP backend server groups

Listener (Protocol/ Port)	Backend Server Group	Backend Server Group Node Mapping and Port		
TCP/80	ingress-http	master-01-IP:80	master-02-IP:80	master-03-IP:80
TCP/443	ingress-https	master-01-IP:443	master-02-IP:443	master-03-IP:443
TCP/5443	kube-apiserver	master-01-IP:5444	master-02-IP:5444	master-03-IP:5444

NOTE

- The configuration page varies depending on the external load balancer. Configure the preceding mappings based on site requirements.
- Before installing on-premises clusters, configure the mappings between the TCP listeners and TCP backend server groups for the external load balancer and ensure that the external load balancer is available.
- The load balancer can route traffic from processes (such as the kubelet process) on all nodes (including master nodes) to three master nodes. In addition, the load balancer can automatically detect and stop routing traffic to unavailable processes, which improves service capabilities and availability. You can also use load balancers provided by other cloud vendors or related hardware devices or use Keepalived and HAproxy to provide HA for master nodes.
- **Recommended configuration**: Enable source IP transparency for the preceding listening ports and disable loop checking. If loop checking cannot be disabled separately, disable source IP transparency. To check whether loop checking exists, perform the following steps:
  1. Create an HTTP service on a server that can be accessed over external networks, change default listening port **80** to **88**, and add the **index.html** file for testing.
     ```
     yum install -y httpd
     sed -i 's/Listen 80/Listen 88/g' /etc/httpd/conf/httpd.conf
     echo "This is a test page" > /var/www/html/index.html
     systemctl start httpd
     ```
     Enter **${*IP address of the server*}:88** in the address box of a browser. "This is a test page" is displayed.
  2. Configure a listening port, for example, **30088**, for the load balancer to route traffic to port **88** of the server, and enable source IP transparency.
  3. Use the private IP address of the load balancer to access the HTTP service.
     ```
     curl -v ${ELB_IP}:30088
     ```
     Check whether the HTTP status code is 200. If the status code is not 200, loop checking exists.

Users

Table 1-10 Users

User	User Group	User ID	User Group ID	Password	Used For
root	root	0	0	-	Default user used for installing on-premises clusters. You can also specify another user that meets the following requirements: ● The user password must be the same on all cluster manage nodes. ● The user has all the permissions of user **root**. **NOTE** After an on-premises cluster is installed, you can change the password or restrict the **root** permissions.
paas	paas	10000	10000	-	User and user group created during the installation of on-premises clusters and used to run on-premises cluster services. The user name and user group name are in the format of **paas**:**paas**, and the user ID and user group ID are in the format of **10000**:**10000**. Ensure that the user name, user group name, user ID, and user group ID are not occupied before the installation. If any of them are occupied, delete the existing one in advance.

1.3.3 Registering an On-Premises Cluster

This section describes how you can register an on-premises cluster with UCS.

Constraints

Only **Huawei Cloud accounts** and users with the **UCS FullAccess** permission can register on-premises clusters.

Prerequisites

- You have applied for an on-premises cluster trial on the UCS console.
- The UCS cluster quota is sufficient.
- At least 20 GB space is available in the **/tmp** directory on the node.
- The executor check items meet the requirements listed in **Installing an On-Premises Cluster**.
- You have prepared an executor that is connected with the cluster network.

Registering a Cluster

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 In the **On-premises cluster** card, click **Register Cluster**.

Step 3 Configure the cluster parameters listed in **Table 1-11**. The parameters marked with an asterisk (*) are mandatory.

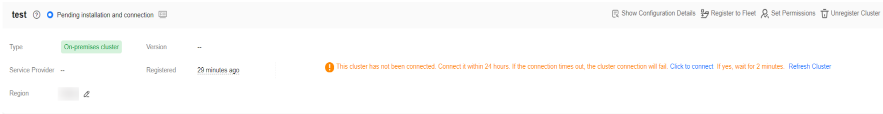
Table 1-11 Parameter description

Parameter	Description
* Cluster Name	Enter a cluster name. Only digits, lowercase letters, and hyphens (-) are allowed, and the name must start with a lowercase letter and cannot end with a hyphen (-).
* Region	Select a region where the cluster is deployed.
Cluster Label	Optional. You can add labels in the form of key-value pairs to classify clusters. A key or value can contain a maximum of 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.
Fleet	Select the fleet that the cluster belongs to. A cluster can be added to only one fleet. Fleets are used for fine-grained access management. If you do not select a fleet, the cluster will be displayed on the **Clusters Not in Fleet** tab upon registration. You can add it to a fleet later. When registering a cluster, you cannot select a fleet with cluster federation enabled. To add your cluster to the fleet with cluster federation enabled, register your cluster with UCS first. For details about cluster federation, see **Enabling Cluster Federation**. For details about how to create a fleet, see **Managing Fleets**.

Step 4 Click **OK**. After the cluster is added, its status is as shown in **Figure 1-3**. You need to connect the cluster to UCS within 24 hours. You can choose either the public or the private network access mode. For details about the network connection process, click in the upper right corner.

If the cluster is not connected to UCS within 24 hours, it will fail to be registered. In this case, click in the upper right corner to register it again. If the cluster has been connected to UCS but no data is displayed, wait for 2 minutes and refresh the cluster.

Figure 1-3 Cluster waiting for network connection



----End

1.3.4 Installing an On-Premises Cluster

1.3.4.1 Pre-Installation Check

Disabling Automatic Software Updates and Upgrades

Disable automatic software updates on nodes. Do not install Docker or upgrade containerd. For details about how to disable automatic software updates in Ubuntu, see **Ubuntu Enable Automatic Updates Unattended Upgrades**.

Checking the OS Language

Ensure the OS language is English.

Checking Apt Sources on Nodes (Ubuntu)

NOTICE

Apt sources can be checked only on nodes running Ubuntu. If your node runs Huawei Cloud EulerOS or Red Hat, check the apt source by referring to **Checking Yum Sources on Nodes (Huawei Cloud EulerOS and Red Hat)**.

Apt sources provide dependency packages required for installing components such as ntpdate on nodes added to on-premises clusters. Make sure the apt sources are available on nodes. If there are any apt sources unavailable, perform the following steps:

Step 1 Log in to the management node as the installation user (**root** by default).

Step 2 Edit **/etc/apt/sources.list**.

Use the actual IP address of the Apt server.

Step 3 Save the file and run the following command:

sudo apt-get update

Step 4 (Optional) If there are multiple management nodes for, for example, HA, log in to each node and perform the preceding operations.

----End

Checking Yum Sources on Nodes (Huawei Cloud EulerOS and Red Hat)

Yum sources provide dependency packages required for installing components such as ntpdate on nodes added to on-premises clusters. Make sure the yum sources are available on nodes. If there are any yum sources unavailable, perform the following steps:

Step 1 Log in to the management node as the installation user (**root** by default).

Step 2 Modify the software source configuration file in **/etc/yum.repos.d/**.

Use the actual IP address of the yum server.

Step 3 Save the file and run the following command:

sudo yum-get update

Step 4 (Optional) If there are multiple management nodes for, for example, HA, log in to each node and perform the preceding operations.

----End

Minimum Installation Requirements

- Do not install unnecessary software packages in the OS.

  To reduce system vulnerabilities and prevent system attacks, install only the necessary software packages and service components.
- Do not retain development and compilation tools in the production environment.

  For example:
  ```
  'cpp' (/usr/bin/cpp)
  'gcc' (/usr/bin/gcc)
  'ld' (/usr/bin/ld)
  'lex' (/usr/bin/lex)
  'rpcgen' (/usr/bin/rpcgen)
  ```
  If interpreters such as Lua and Python are required for product deployment and execution in the production environment, these interpreters can be kept.
  ```
  'python' (/usr/bin/python)
  'lua' (/usr/bin/lua)
  ```
  Some management programs in SUSE Linux rely on the Perl interpreter. In this case, the Perl interpreter can be kept.
  ```
  perl (/usr/bin/perl)
  ```
- Do not install security policy tools in the OS.

  To prevent security information disclosure, ensure that user **root** is the file owner of the preinstalled security hardening tools, and only **root** has the execution permission.
- Do not install network sniffing tools in the OS.

  To prevent malicious use, ensure there are no sniffing tools such as Tcpdump and Ethereal in the OS.

- Do not install modem software in the OS unless necessary.

  To adhere to the principle of minimal installation, do not install modem software unless necessary.

Pre-Installation Check Items

Before installing the on-premises cluster, you need to check the nodes.

The commands in the following table apply to Huawei Cloud EulerOS and Red Hat. If you use Ubuntu, change **yum** in the commands to **apt**.

Category	Item	Description	Criteria
Cluster check	Architecture check	Architecture check for all master nodes	The architectures of all master nodes must be the same.
	Host name check	Host name check for all master nodes	The host names of all master nodes must be unique.
	Time synchronization check	Time synchronization check for all master nodes	The time differences among all master nodes must be less than 10 seconds.
	VIP usage check	Whether the VIP is occupied by other nodes	The VIP must be idle. The method is to check whether port 22 can be accessed.
Node check	Language check	Whether the node language meets the criteria	The node language can be en_US.UTF-8 or en_GB.UTF-8.
	OS check	Whether the node OS meets the criteria	The node OS must be Ubuntu 22.04, Red Hat 8.6, or Huawei Cloud EulerOS 2.0.
	System command check	Whether basic command line tools are available	The OS must have the following command line tools: ifconfig, netstat, curl, systemctl, nohup, pidof, mount, uname, lsmod, swapoff, hwclock, ip, and ntpdate (for NTP servers).
	Idle port check	Whether the ports of mandatory services are idle	The following ports must be idle: 4001, 4002, 4003, 2380, 2381, 2382, 4011, 4012, 4013, 4005, 4006, 4007, 5444, 8080, 10257, 10259, 4133, 20100, 9444, 20102, 9443, 5443, 4134, 4194, 10255, 10248, 10250, 80, 443, 10256, 10249, and 20101

Category	Item	Description	Criteria
	Keepalived installation check	Whether Keepalived is installed	Keeplived must not be installed. You can run the **yum list --installed keepalived** command to check that.
	HAProxy installation check	Whether HAProxy is installed	HAProxy must not be installed. You can run the **yum list --installed haproxy** command to check that.
	Runit installation check	Whether runit is installed	Runit must not be installed. You can run the **yum list --installed runit** command to check that.
	paas user check	Whether the **paas** user can be created on the node	The **paas** user whose ID is **10000** can be created.
	NTP service check	Whether the NTP service is available	The NTP service must be available. You can run the **ntpdate -u ${***ntp_server***}** command to check that NTP is available.

1.3.4.2 Preparing for Installation (Private Network Access)

You need to prepare for installation only when you connect an on-premises cluster to UCS over a private network. If you select **Public access**, you can directly perform operations in Installation and Verification.

Before installing an on-premises cluster, you need to create a VPC, connect the VPC to the on-premises network, create a VPC endpoint, and configure the VPC endpoint on the DNS server in the VPC.

Deploying the Network Environment

Create a VPC in the region where UCS provides services to install the VPC endpoint, and ensure that the VPC can communicate with your on-premises network.

For details about how to create a VPC, see **Creating a VPC**. Currently, only AP-Singapore is supported.

NOTE

The subnet CIDR block of the VPC cannot overlap with the subnet CIDR block of your on-premises data center. If the CIDR blocks overlap, the cluster cannot be connected to UCS. For example, if the subnet of an on-premises data center is 192.168.1.0/24, the subnet of the Huawei Cloud VPC cannot be 192.168.1.0/24.

Connect the on-premises network to the cloud network using either of the following solutions:

- VPN: See **Connecting an On-Premises Data Center to a VPC Through a VPN**.
- NOTICE

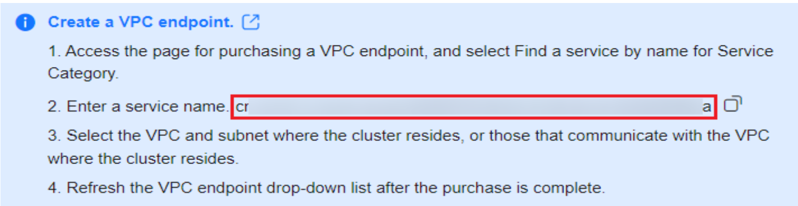
  After the on-premises network or the private network of the third party cloud and the cloud network are connected, you are advised to ping the private IP address of a server in the VPC from an on-premises server or a server of the third-party cloud to check network connectivity.

Buying a VPC Endpoint

Step 1 Log in to the UCS console and click **Click to connect** in the card view of the cluster. In the window that slides out from the right, select **Private access**.

Step 2 In **Create a VPC Endpoint.**, click to record the service name.

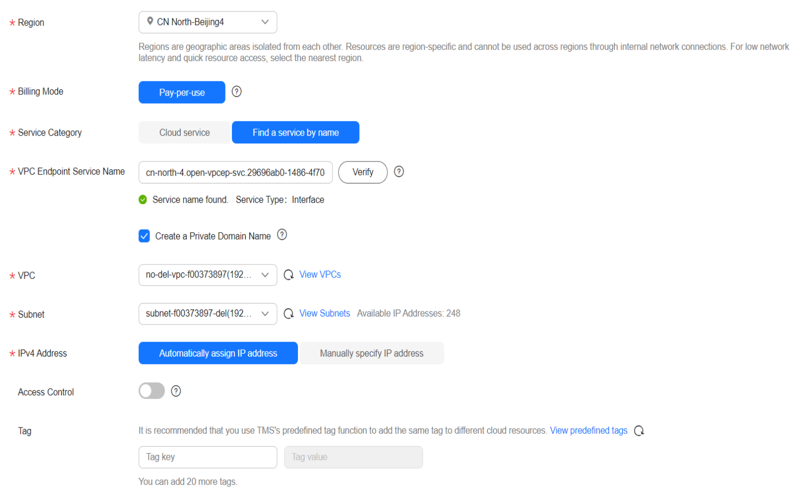
Figure 1-4 Creating a VPC endpoint



Step 3 Log in to the VPC Endpoint console and click **Buy VPC Endpoint** to create a VPC endpoint for connecting to different services.

Step 4 Select the region that the VPC endpoint belongs to, click **Find a service by name**, enter the service name recorded in **Step 2**, and click **Verify**.

Figure 1-5 Searching for a service by name



Step 5 Create VPC endpoints for DNS, SWR, and OBS.

Step 6 Select the VPC and subnet created in **Deploying the Network Environment**.

Step 7 Select **Automatically assign IP address** or **Manually specify IP address** for assigning the private IP address of the VPC endpoint.

Step 8 Click **Next**, confirm the specifications, and click **Submit**.

Step 9 Configure the created VPC endpoint on the DNS server. Click the name of the created VPC endpoint and record the IP address so that the Huawei Cloud DNS forwarder can be added to the DNS server in the on-premises data center.

----End

Configuring a DNS Server

Step 1 **Configure DNS forwarding**: Configure a DNS forwarding rule on the DNS server to forward the request for resolving the Huawei Cloud internal domain name to the endpoint for accessing DNS. Take DNS Bind as an example. In **/etc/named.conf**, add the DNS forwarder configuration and set **forwarders** to the IP address of the endpoint for accessing DNS.

The following code *xx.xx.xx.xx* represents the endpoint IP address of DNS.

```
options {
        forward only;
        forwarders{ xx.xx.xx.xx;};
};
```

Step 2 **Configure static DNS resolution**: Configure static DNS resolution and add the IP addresses of SWR and CIE instances. Take CN North-Beijing4 as an example. If **dnsmasq** is used, add the following two static resolutions to **/etc/dnsmasq.conf**:

The following shows the first static resolution, where *xx.xx.xx.xx* represents the IP address of the SWR endpoint. Replace **region** with the URL of the region that the service belongs to.

```
address=/swr.region.myhuaweicloud.com/xx.xx.xx.xx
```

The following shows the second static resolution, where *xx.xx.xx.xx* represents the IP address that is specific to the domain name and is generated after cluster monitoring is enabled. Replace **region** with the URL of the region that the service belongs to.

```
address=/cia-{First eight digits of the selected VPC ID}{First eight digits of the selected subnet ID}.region.myhuaweicloud.com/xx.xx.xx
```

Example: **address=/cia-9992be3cf3eace24.cn-north-4.myhuaweicloud.com/172.16.0.81**

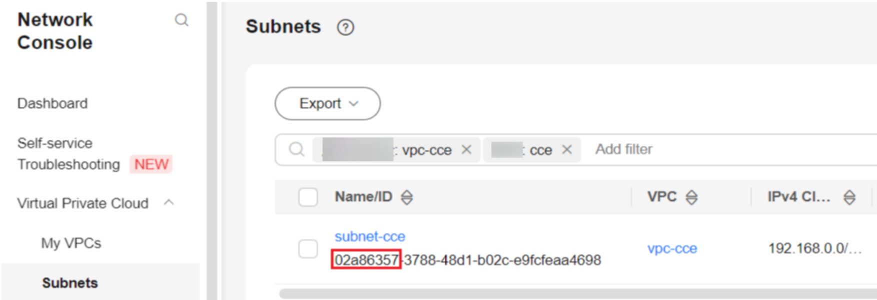
Step 3 **Generate a domain name.**

SWR: address=/swr.cn-north-4.myhuaweicloud.com/*{SWR VPC endpoint}*

CIA: Obtain the domain name. The following figure shows the selected VPC (**vpc-cce** as an example) and subnet.

Figure 1-6 First eight digits of the VPC ID

Figure 1-7 First eight digits of the subnet ID



The final domain name is cia-e52a5d7e02a86357.cn-north-4.myhuaweicloud.com.

----End

1.3.4.3 Installation and Verification

After an on-premises cluster is registered with UCS, its status is **Pending installation and connection**. This means UCS does not install Kubernetes for the cluster, and there is no network connection established between the cluster and UCS. In this case, you need to configure a network agent in the cluster for network connectivity and cluster installation.

NOTICE

Connect the cluster to UCS within 24 hours after the cluster is registered. You can click in the upper right corner to view the detailed network connection process. If the cluster is not connected to UCS within 24 hours, it will fail to be registered. In this case, click in the upper right corner to register it again. If the cluster is connected to UCS but its status is not updated, wait for 2 minutes and refresh the cluster.

Uploading the Configuration File

Step 1 Log in to the UCS console and click **Click to connect** in the card view of the cluster.

Step 2 Select an access mode and download the agent configuration file.

If you select **Public access**, click **Download** to download **agent-***{Cluster name}***.yaml**.

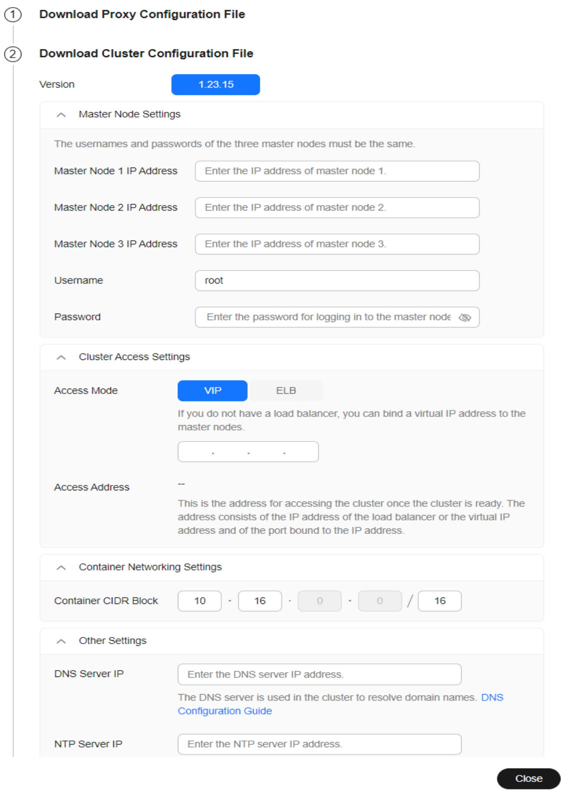
If you select **Private access**, select a project and then the VPC endpoint created in **Preparing for Installation (Private Network Access)** and click **Download** to download **agent-***{Cluster name}***.yaml**.

NOTE

The agent configuration file contains keys and can be downloaded only once for each cluster. Keep the file secure.

Step 3 Set the parameters required for cluster installation and download the cluster configuration file **cluster-{*Cluster name*}.yaml**.

Figure 1-8 Downloading the cluster configuration file



Step 4 Use the remote file transfer tool to upload the **agent-{*Cluster name*}.yaml** and **cluster-{*Cluster name*}.yaml** files to the **/root/** directory on the executor as the root user.

NOTE

- If you want to use load balancing at Layer 4 or Layer 7, set the cluster network type to BGP. For details, see **Cilium**.
- If the SSH connection times out on the executor, rectify the fault by referring to **How Do I Do If VM SSH Connection Times Out?**

----End

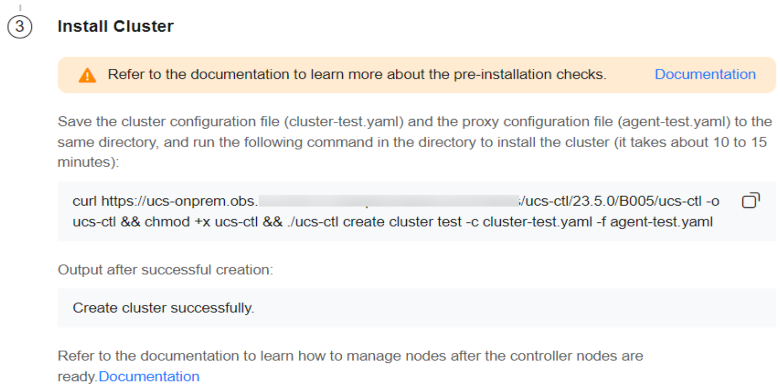
(Optional) Verifying the Integrity of ucs-ctl

ucs-ctl is a command-line tool for managing UCS on-premises clusters. Before installing an on-premises cluster and using ucs-ctl, verify the integrity of ucs-ctl to prevent it from being tampered with. For details about ucs-ctl, see **Using ucs-ctl to Manage On-Premises Clusters**.

In an on-premises cluster, you can use the SHA256 verification file to verify the integrity of the ucsctl file.

Step 1 Click **Install Cluster**, copy the the installation address of ucs-ctl shown in **Figure 1-9**.

Figure 1-9 ucs-ctl installation address



Step 2 Replace the download address in the following command with the address recorded in **Step 1** and run the command to download the SHA256 verification file:

```
curl {download_address}.sha256 -o ucs-ctl.sha256 #
```

Step 3 Save the verification file to the **ucs-ctl** directory and run the following command to verify the integrity of ucs-ctl:

```
sha256sum -c <(grep ucs-ctl ucs-ctl.sha256)
```

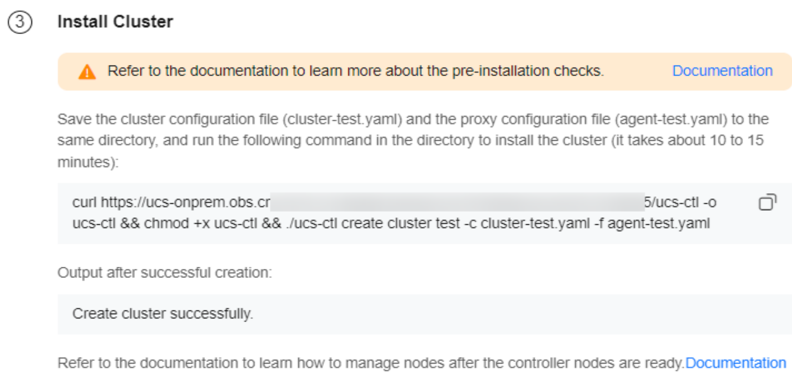
Step 4 If "OK" is displayed in the command output, the verification is successful. If "FAILED" is displayed in the command output, the verification fails. In this case, submit a service ticket and contact technical support personnel.

----End

Installing an On-Premises Cluster

Step 1 Click **Install Cluster**, copy the installation command, and run the command in the **/root** directory (or another available directory).

Figure 1-10 Installing an on-premises cluster



Step 2 Go to the UCS console and refresh the cluster status. The cluster is in the **Running** state.

Step 3 Click the name of the on-premises cluster to access its details page. Perform operations on resources such as cluster nodes and workloads. If the operations can be performed without errors, the on-premises cluster has been successfully connected.

----End

1.3.5 Managing an On-Premises Cluster

1.3.5.1 kubeconfig of an On-Premises Cluster

Obtaining the kubeconfig of an On-Premises Cluster

A kubeconfig file can be used to organize information about clusters, users, namespaces, and authentication mechanisms. The kubectl command-line tool uses the kubeconfig file to find the information it needs to choose a cluster and communicate with the API server of the cluster.

You need to use ucs-ctl to obtain the kubeconfig file of an on-premises cluster.

Step 1 Use ucs-ctl to obtain the name of the on-premises cluster.

```
$ ./ucs-ctl get cluster
```

```
[root@local-cluster-0001 ~]# ./ucs-ctl get cluster
+--------------+--------+---------------+--------------+---------------+---------------+
| CLUSTER NAME | USE ELB |    VIP/ELB    |   MASTER-1   |   MASTER-2    |   MASTER-3    |
+--------------+--------+---------------+--------------+---------------+---------------+
| test-redhat86 | false  | 192.168.0.165 | 192.168.0.68 | 192.168.0.225 | 192.168.0.145 |
+--------------+--------+---------------+--------------+---------------+---------------+
```

Step 2 Use ucs-ctl to export the kubeconfig file of the on-premises cluster.

```
$ ./ucs-ctl get kubeconfig -c test-redhat86 -o kubeconfig
```

NOTE

You can run the **ucs-ctl get kubeconfig -h** command to view the following parameters in a kubeconfig file:

- **-c, --cluster**: specifies the name of the cluster whose kubeconfig file is to be exported.
- **-e, --eip**: specifies the EIP of the API server.
- **-o, --output**: specifies the name of the kubeconfig file.

----End

Using the kubeconfig of an On-Premises Cluster

After obtaining the kubeconfig file generated by ucs-ctl, take the following steps to make this file take effect on the node:

Step 1 Copy the kubeconfig file to the node.

```
$ scp /local/path/to/kubeconfig user@remote:/remote/path/to/kubeconfig
```

Step 2 If environment variable **EnableSecretEncrypt** has been added, delete it first.

```
$ unset EnableSecretEncrypt
```

Step 3 Make the kubeconfig file take effect by using one of the following methods:

- Method 1: Copy the kubeconfig file to the default path.
```
$ mv /remote/path/to/kubeconfig $HOME/.kube/config
```
- Method 2: Specify KUBECONFIG as the environment variable.
```
$ export KUBECONFIG=/remote/path/to/kubeconfig
```

- Method 3: Specify kubeconfig in command lines.
```
$ kubectl --kubeconfig=/remote/path/to/kubeconfig
```

----End

After the preceding operations are performed, kubectl can communicate with the API server of the on-premises cluster. For details about how to use the kubeconfig file, see **Organizing Cluster Access Using kubeconfig Files**.

1.3.5.2 On-Premises Cluster Configuration File

The on-premises cluster configuration file is a **Cluster.yaml** file, which is automatically generated on the UCS console and is used to initialize the master node of the on-premises cluster. **Table 1-12** lists the fields in the configuration file.

Table 1-12 Commands

Configuration Item	Configuration Command
# User for logging in to the master node in SSH mode	USERNAME: root
# Password for logging in to the master node in SSH mode	PASSWORD:
# IP address of the master1 node in the cluster	MASTER-1:
# IP address of the master2 node in the cluster	MASTER-2:
# IP address of the master3 node in the cluster	MASTER-3:
#Whether to use ELB# Whether to use ELB	ACCESS_EXTERNAL_LOAD_BALANCE: false
# IP address of the available ELB	EXTERNAL_LOAD_BALANCE_IP:
# Virtual IP address of the cluster	VIRTUAL_IP:
# Container network service	NETWORK_PROVIDER: cilium
# Container CIDR block	CILIUM_IPV4POOL_CIDR: 172.16.0.0/16
# Cilium BGP switch	CILIUM_BGP_ENABLE: false
# Cilium BGP peer IP address	CILIUM_BGP_PEER_ADDRESS: 127.0.0.1
# Cilium BGP ASN	CILIUM_BGP_PEER_ASN: 65010
# CIDR block for Cilium load balancer	LOAD_BALANCER_CIDR:
# Cilium container network mode	CILIUM_NETWORK_MODE: overlay
# Time zone	TIMEZONE: Asia/Shanghai
# Whether to add taints to the management node	TAINT_MANAGE: yes

# Whether to use NTP	INSTALL_NTP: true
#IP address of the external NTP server	NTP_SERVER_IP:
# Proxy forwarding mode	PROXY_MODE: ebpf
# IP address of the external DNS server	DNS_SERVER_IP:
# External access address of the cluster	CUSTOM_IP:
# Address for downloading the cluster installation package	PACKAGE_PATH:
# Address for downloading the cluster image package	IMAGES_PACKAGE_PATH:
# IAM domain ID	IAM_DOMAIN_ID:
# IAM service address	IAM_ENDPOINT:

1.3.5.3 Managing Nodes in an On-Premises Cluster

This section describes how to use ucs-ctl to manage nodes in an on-premises.

NOTE

ucs-ctl is a command-line tool for managing UCS on-premises clusters. For details about ucs-ctl, see **Using ucs-ctl to Manage On-Premises Clusters**.

Adding a Node to an On-Premises Cluster

Step 1 Run the **./ucs-ctl config generator -t node -o node.csv** command on the executor to generate the configuration file used for managing nodes.

Step 2 Write the parameters of the required node to the configuration file and use commas (,) to separate the parameters. **Table 1-13** describes the parameters.

Table 1-13 Parameters in the configuration file

Parameter	Description
Node IP	Node IP
User	Username for SSH connection
Password	Password for SSH connection

Example:

```
Node IP,User,Password
123.45.6.789,root,********
123.45.6.890,root,********
```

Step 3 Run the **./ucs-ctl create node -c [*Cluster name*] -m node.csv** command on the executor to manage the node.

----End

CAUTION

The **node.csv** file contains keys. Keep the file secure.

Deleting a Node from an On-Premises Cluster

- Method 1:

  Run the following command on the executor:

  ```
  ./ucs-ctl delete node -c [Cluster name] -n [node ip1],[node ip2],...
  ```

  -n specifies IP addresses. Use commas (,) to separate the IP addresses.

- Method 2:

  Run the following command on the executor:

  ```
  ./ucs-ctl delete node -c [Cluster name] -m node.csv
  ```

  -m specifies the configuration file used for managing nodes. You can delete all nodes at a time.

NOTE

If nodes fail to be deleted from an on-premises cluster, perform operations in **How Do I Manually Clear Nodes of an On-premises Cluster?**

1.3.5.4 Managing On-Premises Cluster Networks

1.3.5.4.1 Cilium Overview

Why Cilium?

Cilium is a high-performance and high-reliability solution for securing network connectivity between containers. At the foundation of Cilium lies a technology rooted in the Linux kernel, namely the extended Berkley Packet Filter (eBPF). Cilium supports multiple transport layer protocols, such as TCP, UDP, and HTTP, and provides multiple security features, such as access control at the application layer and support for service mesh. Cilium also supports Kubernetes network policies and provides global networking and service discovery to help administrators better manage and deploy cloud-native applications.

Cilium uses eBPF to monitor network traffic inside the kernel in real time, which enables efficient, secure packet exchange. eBPF shines in many scenarios such as network functions virtualization, container networks, and edge computing. It helps enterprises improve network performance and security and provides better infrastructure support for cloud-native applications.

Basic Functions

- Network connectivity: Cilium allocates a unique IP address to each container for communications between containers. Cilium also supports multiple network protocols.

- Network intrusion detection: Cilium can integrate third-party network intrusion detection services, such as Snort, to detect network traffic.
- Automatic security policy management: Cilium automatically creates security policies for each container using the Kubernetes custom resource definition (CRD) mechanism to ensure container security.
- Load balancing: Cilium provides multiple load balancing algorithms to route traffic across containers.
- Service discovery: Cilium uses the Kubernetes service detection mechanism to automatically detect services in containers and register the services with Kubernetes APIs for other containers to access.

Constraints

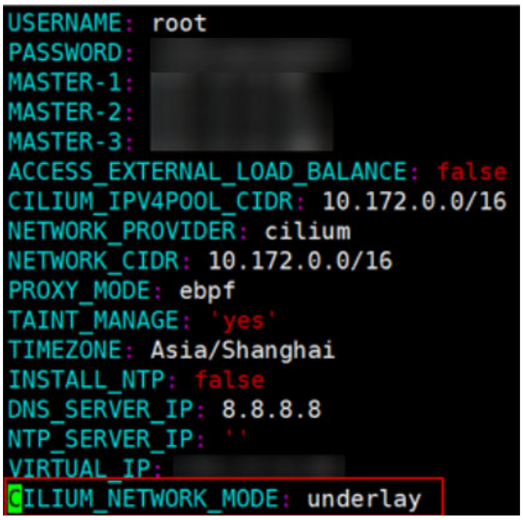
Only new on-premises clusters support Cilium. Existing on-premises clusters do not support Cilium even after they are upgraded.

Cilium Underlay

Add the following settings to the on-premises cluster configuration file **cluster-[*Cluster name*].yaml**.

```
CILIUM_NETWORK_MODE: underlay
```

Example:



Advantages

- If Cilium works with underlay, Cilium sends all packets that are not sent to other containers to the routing system of the Linux kernel. This means that the packets will be forwarded by a route, as if the local process sends the data packets, which reduces the encapsulation and conversion of the packets. This method is better used when the traffic is heavy.
- **ipv4-native-routing-cidr** is automatically configured so that Cilium automatically enables IP forwarding in the Linux kernel.

Dependency

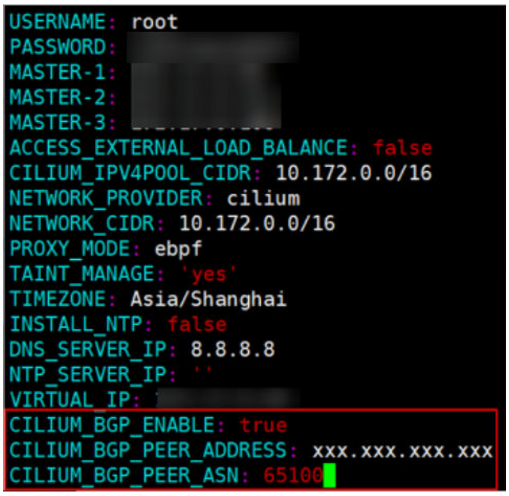
The network of the host running Cilium can use the IP address allocated to the pod or other workloads to forward traffic. The source and destination address check of the node must be disabled, and the security group of the node must allow traffic from and to the container CIDR block over the port and using the protocol of the node.

Enabling BGP for Cilium

Add the following settings to the on-premises cluster configuration file **cluster-[*Cluster name*].yaml**.

```
CILIUM_BGP_ENABLE: true
CILIUM_BGP_PEER_ADDRESS: IP address of the switch to be interconnected
CILIUM_BGP_PEER_ASN: BGP ASN of the switch to be interconnected (64512-65535)
LOAD_BALANCER_CIDR: Load balancer CIDR block that needs to be broadcast and can be used by open-source add-ons such as MetalLB
```

Example:



Configure the IP address of the node that needs to be exposed as the neighbor address on the BGP network where the host is located. The default ASN of the container is **65010**.

The BGP capability of Cilium is to advertise node container routes at the node granularity so that services out of the cluster can directly access the pods in the cluster.

1.3.5.4.2 MetalLB for Load Balancing at Layer 4

Kubernetes does not offer an implementation of network load balancers (Services of type LoadBalancer) for bare-metal clusters. Bare-metal cluster operators are left with two types of Services, NodePort and externalIPs, to bring user traffic into their clusters. MetalLB aims to redress this imbalance by offering a network load balancer implementation, so that external services on bare-metal clusters can work better. For details about MetalLB, see the **official projects of the community** and the **MetalLB official website**.

This section describes how to create and use MetalLB in on-premises clusters.

Constraints

Currently, MetalLB can be installed only in on-premises clusters.

Prerequisites

You have enabled BGP for your on-premises cluster and configured **LOAD_BALANCER_CIDR** to broadcast the load balancer CIDR block to the underlying network.

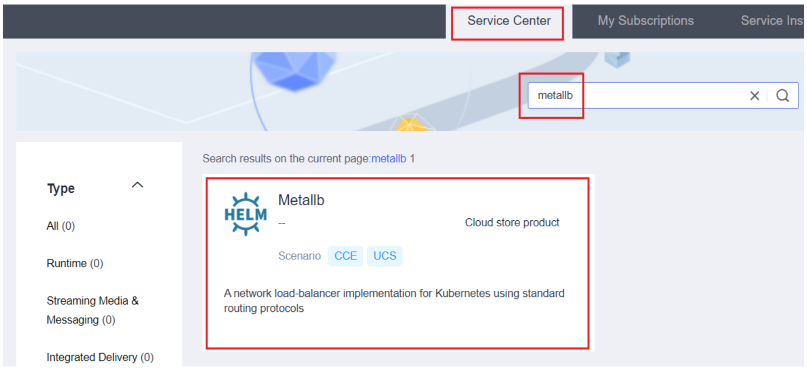
Installing MetalLB

Step 1 Log in to the UCS console.

Step 2 In the navigation pane on the left, choose **Operator Service Center**.

Step 3 Search for Metallb in the service center and click the name of Metallb to access its details page.

Figure 1-11 Searching for Metallb



Step 4 Subscribe to MetalLB, click **Create Instance**, and select the target cluster. Complete the installation as prompted.

----End

Function Verification

Step 1 Access the cluster details page.

- If the cluster is not added to any fleet, click the cluster name.
- If the cluster has been added to a fleet, click the fleet name. In the navigation pane, choose **Clusters** > **Container Clusters**.

Step 2 In the navigation pane on the left, choose **Workloads**. On the displayed page, click **Create from Image**.

Step 3 Select an available image to create a workload and add a LoadBalancer Service. For details, see **Workload Service Configuration**.

Step 4 Click the service name, copy the IP address of the load balancer for access from a node outside the cluster to verify that the access is successful.

Figure 1-12 Load balancer IP address

----End

1.3.5.4.3 Ingress-NGINX for Load Balancing at Layer 7

The Ingress-NGINX controller is used to store NGINX configuration and implement unified traffic forwarding. For details about Ingress-NGINX, see **Ingress-NGINX Controller** and **official community projects**.

This section describes how you can install and use Ingress-NGINX for an on-premises cluster.

Constraints

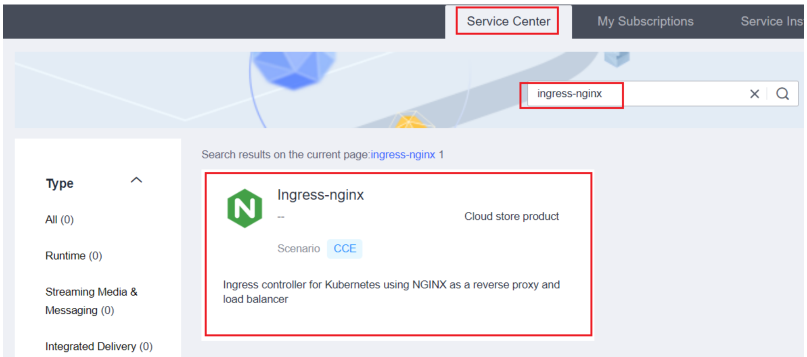
Currently, Ingress-NGINX can be installed only in on-premises clusters.

Installing Ingress-NGINX

Step 1 Log in to the UCS console.

Step 2 In the navigation pane on the left, choose **Operator Service Center**. On the **Service Center** tab, search for Ingress-NGINX and click this add-on to go to its details page.

Figure 1-13 Searching for Ingress-NGINX



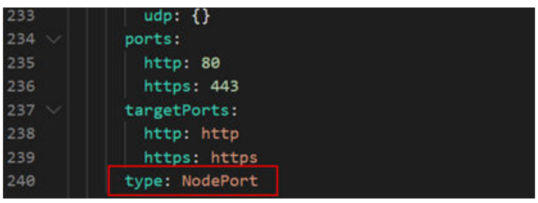
Step 3 Subscribe to Ingress-NGINX, click **Create Instance**, and select the target cluster.

- If MetalLB has been installed in the cluster, you can use the load balancing capability of MetalLB to expose Ingress-NGINX to external networks and install Ingress-NGINX as prompted.
- If MetalLB is not installed in the cluster, Ingress-NGINX can be exposed only with the help of NodePort.

NOTE

You need to change the value of **.values.controller.service.type** from **LoadBalancer** to **NodePort** before installing Ingress-NGINX, as shown in **Figure 1-14**.

Figure 1-14 Changing the parameter value



----End

Function Verification

Step 1 Log in to the UCS console.

- If the cluster is not added to any fleet, click the cluster name.
- If the cluster has been added to a fleet, click the fleet name. In the navigation pane on the left, choose **Clusters** > **Container Clusters**. Then, click the cluster name to access the cluster console.

Step 2 In the navigation pane on the left, choose **Workloads**. Then, click **Create from Image**.

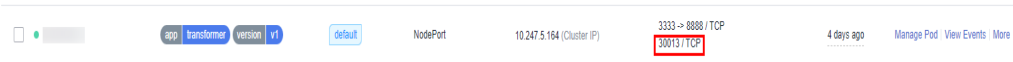
Step 3 Select an available image to create a workload. In **Service Settings**, click + to add a Service of type ClusterIP. For details, see **Workload Service Configuration**.

Step 4 In the navigation pane on the left, choose **Services & Ingresses**. Then, click the **Ingresses** tab, click **Create Ingress**, and select the Service of type ClusterIP. For details about how to configure an ingress, see **Ingresses**.

Step 5 Access this Service to verify that the forwarding rule is successfully configured.

- If this Service is exposed by LoadBalancer, select the Service of type LoadBalancer to make the Service reachable outside the cluster.

- If this Service is exposed by NodePort, select any node and use the Service port to make the Service reachable outside the cluster.



----End

1.3.5.5 Upgrading an On-Premises Cluster

Cluster upgrades improve the lifecycle management of on-premises clusters. You can use the command line tool on the on-premises cluster details page to upgrade the on-premises cluster. You can refer to the prompts and guides provided on the UCS console when you upgrade a cluster.

Constraints

- To upgrade an on-premises cluster, upgrade the master nodes and components first, and then upgrade the worker nodes.
- The upgrade prompts on the cluster list page depend on the status of the master nodes. The upgrade needs to be completed at a time. The worker nodes that are not upgraded are not displayed in the cluster list.
- The version to be upgraded cannot be selected. By default, the cluster is upgraded to the latest version.
- During the master node upgrade, the cluster on the cluster details page may be unavailable for a short period of time. After the upgrade is complete, the cluster will be connected again.

Upgrade Operations

Step 1 Log in to the UCS console, select a running cluster to be upgrade on the **Fleets** or **Clusters Not in Fleet** tab, and click **Cluster Upgrade** in the lower right corner.

Step 2 Use a node that can connect to the cluster as the executor. Run the following command to download the cluster management tool of the new version:

```
curl https://ucs-onprem.obs.XXXX.huawei.com/toolkits/ucs-ctl/ucs-ctl -o ucs-ctl && chmod +x ucs-ctl
```

Step 3 Upgrade the master nodes. You can run the **-y** command to answer yes for all questions. For details about other configurable flags, see **Commands for Upgrading Master Nodes and Components**.

```
./ucs-ctl upgrade cluster [cluster name]
```

CAUTION

The cluster name must be the same as that specified when the on-premises cluster is created. If you are not sure about the cluster name, run the following command to view the cluster name:

```
./ucs-ctl get cluster
```

Step 4 Upgrade the worker nodes in either of the following methods:

- Upgrade all worker nodes in the cluster by running the following command:
  ```
  ./ucs-ctl upgrade node -a -c [cluster name]
  ```
- Upgrade the worker nodes in batches to prevent service interruption during the upgrade. In this case, you need to manually select the worker nodes.
  ```
  ./ucs-ctl upgrade node -n [node ip] -c [cluster name]
  ```

NOTE

If there are only master nodes in an on-premises cluster, only the upgrade command of the master nodes is required.

For details about other configurable flags, see **Commands for Upgrading Worker Nodes**.

----End

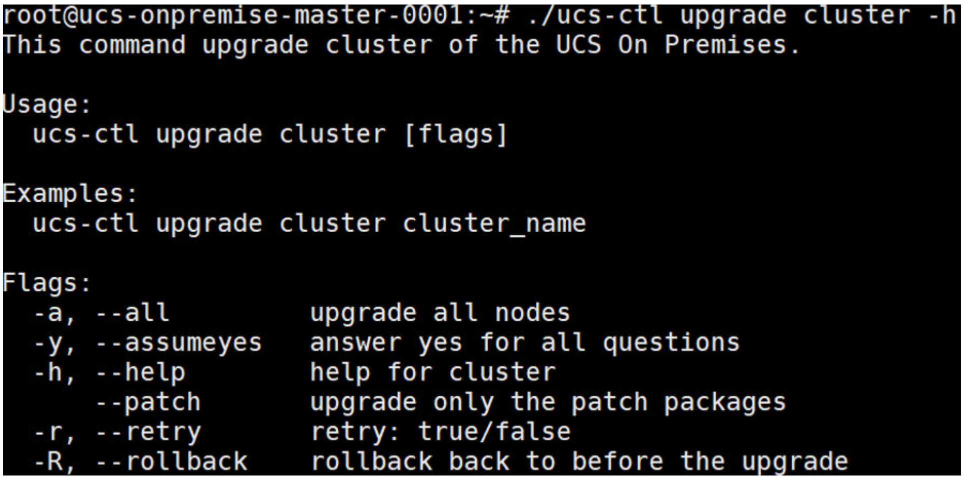
Commands for Upgrading Master Nodes and Components

You can use ucs-ctl of the latest version to upgrade your on-premises cluster. To upgrade the master nodes and components, run commands in the following format:

```
./ucs-ctl upgrade cluster [cluster_name] [flags]
```

The following flags can be configured:

- **-a**: upgrades all master and worker nodes. By default, only the master nodes and components are upgraded.
- **-y**: answers yes for all questions.
- **-patch**: upgrades only the patch packages.
- **-R**: rolls back the upgrade.



```
root@ucs-onpremise-master-0001:~# ./ucs-ctl upgrade cluster -h
This command upgrade cluster of the UCS On Premises.

Usage:
  ucs-ctl upgrade cluster [flags]

Examples:
  ucs-ctl upgrade cluster cluster_name

Flags:
  -a, --all          upgrade all nodes
  -y, --assumeyes    answer yes for all questions
  -h, --help         help for cluster
      --patch        upgrade only the patch packages
  -r, --retry        retry: true/false
  -R, --rollback     rollback back to before the upgrade
```

Commands for Upgrading Worker Nodes

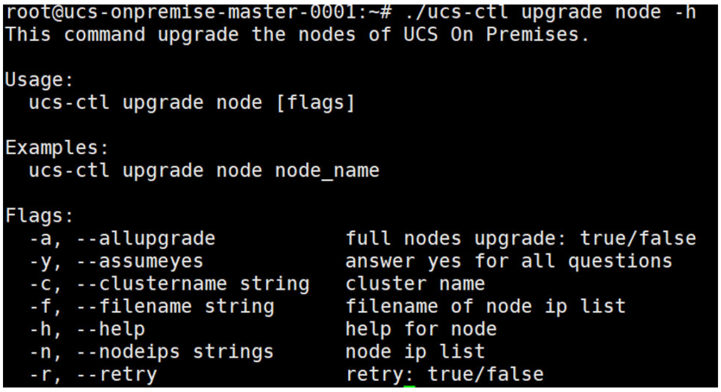
To upgrade worker nodes, run commands in the following format:

```
./ucs-ctl upgrade node [flags] -c [cluster_name]
```

The cluster name must be specified, so the **-c** *[cluster_name]* flag must be added.

The following flags can be configured:

- **-a**: upgrades all worker nodes.
- **-y**: answers yes for all questions.
- **-c**: specifies the cluster name.
- **-R**: rolls back the upgrade.
- **-n**: specifies the node IP addresses.
- **-f**: specifies the node configuration file.



```
root@ucs-onpremise-master-0001:~# ./ucs-ctl upgrade node -h
This command upgrade the nodes of UCS On Premises.

Usage:
  ucs-ctl upgrade node [flags]

Examples:
  ucs-ctl upgrade node node_name

Flags:
  -a, --allupgrade              full nodes upgrade: true/false
  -y, --assumeyes               answer yes for all questions
  -c, --clustername string      cluster name
  -f, --filename string         filename of node ip list
  -h, --help                    help for node
  -n, --nodeips strings         node ip list
  -r, --retry                   retry: true/false
```

1.3.5.6 Unregistering an On-Premises Cluster

Unregistering an On-Premises Cluster on the Console

CAUTION

If you unregister an on-premises cluster on the console, the on-premises cluster will not be deleted.

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 Locate the on-premises cluster to be unregistered.

- If the on-premises cluster has been added to a fleet, click the fleet name to access the fleet console. In the navigation pane on the left, choose **Clusters** > **Container Clusters**.
- If the on-premises cluster is not added to any fleet, click **Clusters Not in Fleet** on the top of the fleet list.

Step 3 Click the unregistration button in the upper right corner of the on-premises cluster.

Step 4 Confirm the information such as the cluster name, select **I have read and understood the preceding information**, and click **OK**.

----End

Deleting an On-Premises Cluster

CAUTION

Deleting an on-premises cluster may make cluster-specific resources (such as workloads scheduled to this cluster) unavailable.

Step 1 Manually delete the on-premises cluster.

Step 2 Copy the uninstallation command returned by the console.

Step 3 Run the uninstallation command on the node in the on-premises cluster.

```
./ucs-ctl delete cluster cluster_name
```

NOTE

Replace *cluster_name* with the actual cluster name.

----End

1.3.5.7 Using ucs-ctl to Manage On-Premises Clusters

ucs-ctl is a command line tool and can only be used by UCS on-premises clusters.

Before using ucs-ctl, verify its integrity to prevent it from being tampered with. For details, see "Verifying the Integrity of ucs-ctl" in **Installation and Verification**.

Table 1-14 Common commands

Command	Description
config generator	Provides templates for creating clusters and nodes.
create	Creates clusters or adds nodes.
delete	Deletes clusters or removes nodes.
get	Obtains on-premises cluster information.

Command	Description
help	Obtains help information.
version	Obtains ucs-ctl version information.

Parameters

ucs-ctl config generator

```
Flags:
        -o Path and name of the file to be exported
        -t Type of the template to be exported, which can be cluster or node.
```

Example:

```
./ucs-ctl config generator -t clustername
```

ucs-ctl create

- Creating a cluster: **ucs-ctl create cluster**

Object:

Clustername: cluster name

```
Flags:
        -f, --agent string       Cluster connection configuration file
        -c, --config string      Cluster configuration file
        -h, --help               Help information
        -r, --retry              Installation retry
```

Example:

```
./ucs-ctl create cluster clustername -c cluster.yaml -f agent.yaml
```

- Adding a node: **ucs-ctl create node**

```
Flags:
        -c, --cluster string     Name of the cluster that the node is to be added to
        -h, --help               Help information
        -m, --machine string     Information about the node to be added to the cluster
        -r, --retry              Node management retry
```

Example:

```
./ucs-ctl create node -c cluster_name -m machine.csv
```

ucs-ctl delete

- Deleting a cluster: **ucs-ctl delete cluster**

```
Flags:
        -y, --default-yes        Operation for confirming the cluster deletion
        -h, --help               Help information
```

Example:

```
./ucs-ctl delete cluster clustername
```

- Deleting a node: **ucs-ctl delete node**

```
Flags:
        -y, --assumeyes          Operation for confirming the node deletion
        -c, --cluster string     Name of the cluster that the node to be deleted from
        -h, --help               Help information
```



```
    -m, --machine string     Information about the node to be deleted from the cluster
    -n, --node-ip string     IP address of the node to be deleted
```

Example:

```
./ucs-ctl delete node -c clustername -m machine.csv
```

ucs-ctl get

- Obtaining on-premises cluster information: **ucs-ctl get cluster**

Example:

```
./ucs-ctl get cluster
```

- Obtaining kubeconfig information: **ucs-ctl get kubeconfig**

```
Flags:
    -c, --cluster string    Cluster name
    -e, --eip string        EIP used as the API access point
    -h, --help              Help information
    -o, --output string     Path of the file to be exported
```

Example:

```
./ucs-ctl get kubeconfig -c clustername -o kubeconfig
```

1.3.5.8 GPU Virtualization

1.3.5.8.1 Overview

On-premises clusters use xGPU for GPU virtualization to allow you to dynamically allocate the GPU memory and compute. A GPU can be virtualized into a maximum of 20 vGPUs. Dynamic allocation provides more flexibility than static allocation. You can assign the right amount of GPU for service stability, which improves the GPU utilization.

Advantages

GPU virtualization of on-premises clusters has the following advantages:

- **Flexible:** The compute and GPU memory are configured in a refined manner. The compute can be allocated at a granularity of 5% of the GPU, and the GPU memory at a granularity of 1 MiB.
- **Isolated:** There are two isolation modes: GPU memory isolation and isolation of GPU memory and compute.
- **Compatible:** There is no need to recompile the services or replace the CUDA library.

1.3.5.8.2 Preparing GPU Virtualization Resources

This section describes how you can plan and prepare basic software and hardware before using GPU virtualization.

Basic planning

Resource	Version
Cluster	v1.25.15-r7 or later

Resource	Version
OS	Huawei Cloud EulerOS 2.0
GPU	T4 and V100
GPU driver	470.57.02, 470.103.01, 470.141.03, 510.39.01, and 510.47.03
Container runtime	containerd
Add-ons	The following add-ons must be installed in a cluster: ● **volcano**: 1.10.1 or later ● **gpu-device-plugin**: 2.0.0 or later

Procedure 1: Adding GPU Nodes to a Cluster and Labeling the Nodes

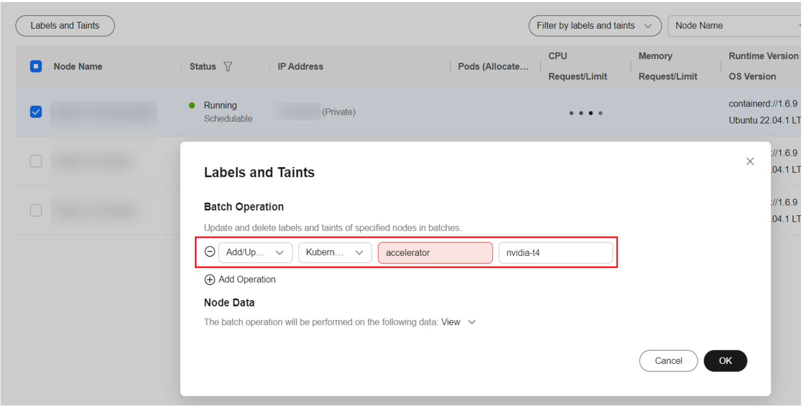
NOTE

If there are GPU nodes that comply with the **basic planning** in your cluster, skip this procedure.

Step 1 Add nodes that support GPU virtualization to your cluster. For details, see **Adding Nodes to On-Premises Clusters**.

Step 2 Label the nodes with **accelerator: nvidia-***{GPU model}*. For details, see Adding Labels/Taints to Nodes.

Figure 1-15 Labeling nodes that support GPU virtualization



----End

Procedure 2: Installing the Add-ons

NOTE

If the add-ons that comply with **the basic planning** have been installed in your cluster, you can skip this procedure.

If the driver version is changed, restart the node to apply the change.

Step 1 Log in to the UCS console and click the cluster name to access the cluster console. In the navigation pane on the left, choose **Add-ons**. In the **Add-ons Installed** area, check whether volcano and gpu-device-plugin have been installed.

Step 2 If volcano is not installed, install it by referring to **volcano**.

If gpu-device-plugin is not installed, install it by referring to **gpu-device-plugin**.

----End

1.3.5.8.3 Creating a Workload That Will Receive vGPU Support

This section describes how to use GPU virtualization to isolate the compute and GPU memory and efficiently use GPU resources.

Prerequisites

- You have **prepared GPU virtualization resources**.
- If you want to create a cluster by running commands, use kubectl to connect to the cluster. For details, see Connecting to a Cluster Using kubectl.

Constraints

- The init container does not support GPU virtualization.
- For a single GPU:
  - A maximum of 20 vGPUs can be created.
  - A maximum of 20 pods that use the isolation capability can be scheduled.
  - Only workloads in the same isolation mode can be scheduled. (GPU virtualization supports two isolation modes: GPU memory isolation and isolation of GPU memory and compute.)
- For different containers of the same workload:
  - You can configure one GPU model and cannot configure two or more GPU models concurrently.
  - You can configure the same GPU usage mode and cannot configure virtualization and non-virtualization modes concurrently.
- After a GPU is virtualized, the GPU cannot be used by workloads that use **shared GPU resources**.

Creating a Workload That Will Receive vGPU Support on the Console

Step 1 Log in to the UCS console.

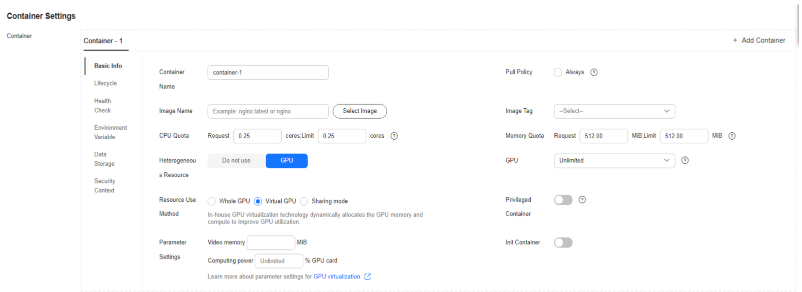
Step 2 Click the on-premises cluster name to access its details page, choose **Workloads** in the navigation pane, and click **Create Workload** in the upper right corner.

Step 3 Configure workload parameters. In **Container Settings**, choose **Basic Info** and set the GPU quota.

Video memory: The value must be a positive integer, in MiB. If the configured GPU memory exceeds that of a single GPU, GPU scheduling cannot be performed.

Computing power: The value must be a multiple of 5, in %, and cannot exceed 100.

Figure 1-16 Configuring workload information



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

Step 5 Verify the isolation capability of GPU virtualization.

- Log in to the target container and check its GPU memory.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Wed Apr 12 07:54:59 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4792MiB /  5000MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

5,000 MiB of GPU memory is allocated to the container, and 4,792 MiB is used.

- Run the following command on the node to check the isolation of the GPU memory:

```
export PATH=$PATH:/usr/local/nvidia/bin;nvidia-smi
```

Expected output:

```
Wed Apr 12 09:31:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4837MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                          4835MiB |
+-----------------------------------------------------------------------------+
```


16,160 MiB of GPU memory is allocated to the GPU node, and 4,837 MiB is used by the pod.

----End

Creating a Workload That Will Receive vGPU Support Using kubectl

Step 1 Log in to the master node and use kubectl to connect to the cluster.

Step 2 Create a workload that will support vGPUs. Create a **gpu-app.yaml** file.

NOTE

There are two isolation modes: GPU memory isolation and isolation of both GPU memory and compute. **volcano.sh/gpu-core.percentage** cannot be set separately for GPU compute isolation.

- Isolate the GPU memory only:

```
apiVersion: apps/v1
 kind: Deployment
 metadata:
   name: gpu-app
   labels:
     app: gpu-app
 spec:
   replicas: 1
   selector:
     matchLabels:
       app: gpu-app
   template:
      metadata:
       labels:
         app: gpu-app
     spec:
       containers:
       - name: container-1
         image: <your_image_address>     # Replace it with your image address.
         resources:
           limits:
            volcano.sh/gpu-mem: 5000    # GPU memory allocated to the pod
       imagePullSecrets:
         - name: default-secret
```

- Isolate both the GPU memory and compute:

```
apiVersion: apps/v1
 kind: Deployment
 metadata:
   name: gpu-app
   labels:
     app: gpu-app
 spec:
   replicas: 1
   selector:
     matchLabels:
       app: gpu-app
   template:
      metadata:
       labels:
         app: gpu-app
     spec:
       containers:
       - name: container-1
         image: <your_image_address>     # Replace it with your image address.
         resources:
           limits:
             volcano.sh/gpu-mem: 5000    # GPU memory allocated to the pod
            volcano.sh/gpu-core.percentage: 25    # Compute allocated to the pod
```

```
    imagePullSecrets:
      - name: default-secret
```

Table 1-15 Key parameters

Parameter	Required	Description
volcano.sh/gpu-mem	No	The value must be a positive integer, in MiB. If the configured GPU memory exceeds that of a single GPU, GPU scheduling cannot be performed.
volcano.sh/gpu-core.percentage	No	The value must be a multiple of 5, in %, and cannot exceed 100.

Step 3 Run the following command to create a workload:

```
kubectl apply -f gpu-app.yaml
```

Step 4 Verify the isolation.

- Log in to a container and check its GPU memory.

```
kubectl exec -it gpu-app -- nvidia-smi
```

Expected output:

```
Wed Apr 12 07:54:59 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4792MiB /  5000MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

5,000 MiB of GPU memory is allocated to the container, and 4,792 MiB is used.

- Run the following command on the node to check GPU memory isolation:

```
/usr/local/nvidia/bin/nvidia-smi
```

Expected output:

```
Wed Apr 12 09:31:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4837MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
```

```
|      ID   ID                                         Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                        4835MiB |
+-----------------------------------------------------------------------------+
```

16,160 MiB of GPU memory is allocated to the node, and 4,837 MiB is used by the pod in this example.

----End

1.3.5.8.4 Monitoring GPU Virtualization Resources

This section describes how to view global monitoring metrics of GPU virtualization resources on the UCS console.

Prerequisites

- You have prepared GPU virtualization resources.
- You have enabled GPU virtualization on some nodes in the on-premises cluster.
- You have enabled monitoring for the on-premises cluster.

GPU Virtualization Monitoring

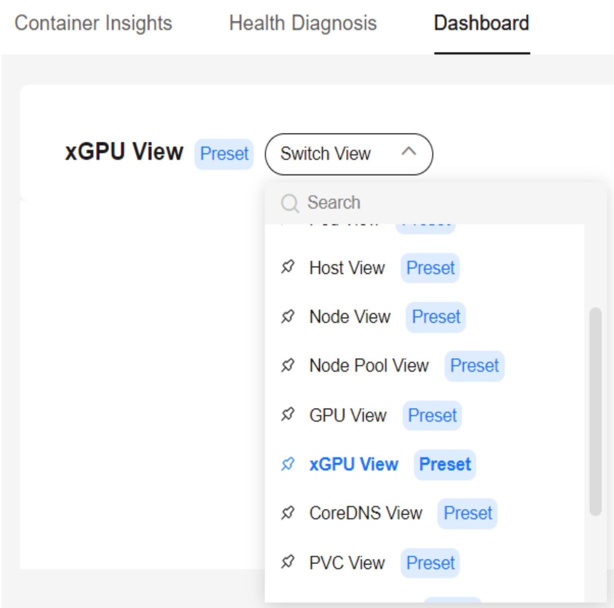
Step 1 Log in to the UCS console. In the navigation pane, choose **Container Intelligent Analysis**.

Step 2 Locate the target cluster and enable monitoring. For details, see **Enabling Monitoring for a Cluster**.

Step 3 Click the cluster name to go to the **Container Insights** tab.

Step 4 Click **Dashboard** and click **Switch View** next to the cluster view to switch to **xGPU View**.

Figure 1-17 Dashboard



Step 5 View the xGPU view.

----End

1.3.5.9 Backup and Restoration

Context

After an on-premises cluster is registered with UCS, you can back up the certificates, encryption and decryption materials, and etcd data on the three master nodes to ensure cluster high availability and prevent data loss when the cluster is faulty. These backups can be used to restore data.

Constraints

The node IP addresses must remain unchanged regardless of whether a single master node or master nodes are faulty.

Cluster Backup

Local backup

1. Create a path for storing the backup file package.
2. Run the following backup command:

```
./ucs-ctl backup {Cluster name}  --path {Backup path} --type local
```

   Example:

```
./ucs-ctl backup gpu-test --path /home/ggz/gpu-test --type local
```

   After the command is executed successfully, a backup file package in the format of *{Cluster name}*-**backup-***{Timestamp}***.tar.gz** is generated in the specified backup path.

   The package stores the **ha.yaml** file and the **etcd-snapshot** and **crt** directories. The **etcd-snapshot** directory contains etcd data, and the **crt** directory contains certificates and encryption and decryption materials.

Remote backup

1. Create a path for storing the backup file package on the remote host over SFTP.
2. Run the following backup command:

```
./ucs-ctl backup {Cluster name} --path {Backup path} --type sftp --ip {IP address of the remote host}
--user {Username of the remote host}
```

   Example:

```
./ucs-ctl backup gpu-test --path /home/ggz/gpu-test --type sftp --ip 100.95.142.93 --user root
```

NOTICE

- If you perform remote backup for the first time, enter the password of the remote host after "please input sftp password:" is displayed.
- The backup path in the backup command must be valid. Otherwise, the backup file generated on the remote host after the command is executed may be incorrect.

After the command is executed successfully, a backup file package in the format of *{Cluster name}*-**backup-***{Timestamp}***.tar.gz** is generated in the specified backup path on the remote host. The package stores the **ha.yaml** file and the **etcd-snapshot** and **crt** directories. The **etcd-snapshot** directory contains etcd data, and the **crt** directory contains certificates and encryption and decryption materials.

Periodic backup

Run the **crontab -e** command to compile a crontab expression so that the backup command is executed periodically.

In the following example, the local backup task of the cluster is executed at 16:40 every day.

```
40 16 * * * /root/cluster/ucs-ctl backup cluster-redhat --path /root/cluster/backup --type local
```

During periodic remote backup, you do not need to specify the password in the crontab expression after entering the password of the remote host for the first time.

To prevent the number of backup files from increasing, compile the crontab expression on the remote host to periodically execute the backup file aging script. The following is an example of the backup file aging script:

```
#!/bin/bash
backup_dir=${1}                                   # Path for storing backup files
keep_days=${2}                                    # Retention days of backup files
if [ ! -d ""$backup_dir"" ]; then                   # Check whether the log path exists.
  echo "There is no backup file path."
  exit 1
fi
find "$backup_dir" -type f -mtime +$keep_days -exec rm {} \;       # Delete legacy logs.
echo "Expired backup files have been deleted!"
```

Data Restoration

Restoring etcd Data

1. Prepare an etcd data backup package in the format of *{Cluster name}*-**backup-***{Timestamp}***.tar.gz**.

   Upload the package to each master/etcd node of the cluster.

2. Stop the etcd service.

   Run the following command on the node:

   ```
   mv /var/paas/kubernetes/manifests/etcd*.manifest /var/paas/kubernetes/
   ```

   Wait until the service is stopped.

   ```
   crictl ps | grep etcd
   ```

   If no etcd container is found, the service has been stopped.

   ```
   root[redacted]01:~# crictl ps | grep etcd
   root[redacted]01:~#
   ```

3. (Optional) Back up the etcd data on a node.

   ```
   mv /var/paas/run/etcd/data /var/paas/run/etcd/data-bak
   mv /var/paas/run/etcd-event/data /var/paas/run/etcd-event/data-bak
   ```

4. Run the restoration command on the node where the etcd database resides.

   ```
   ./ucs-ctl restore etcd {Path of the etcd data backup package}
   ```

   Example:

```
./ucs-ctl restore etcd /home/ggz/gpu-test/backup-file-20230625164904.tar.gz
```

The etcd data is restored if the following command output is displayed:

```
Restore the etcd snapshot successfully.
```

5. Restart the etcd service on the node. The restart takes several minutes.

```
mv /var/paas/kubernetes/etcd*.manifest /var/paas/kubernetes/manifests
```

Wait for the service to restart.

```
crictl ps | grep etcd
```

If the etcd containers are found, the service has been restarted and the etcd data is restored on the node.

NOTE

To restore etcd data, perform steps **1** to **5** on each node where the etcd database resides.

Restoring a single master node

Run the single-node fault recovery command on the executor:

```
./ucs-ctl restore node {IP address of the node} --name {Cluster name}
```

{IP address of the node} indicates the IP address of the faulty node. The following is an example:

```
./ucs-ctl restore node 192.168.0.87 --name gpu-test
```

The fault on the single master node is rectified if the following command output is displayed:

```
restore node 192.168.0.87 successfully.
```

Restore the etcd data on the node. For details, see **4**.

Restoring master nodes

Run the cluster fault recovery command on the executor:

```
./ucs-ctl restore cluster {Cluster name} -b {Path of the backup file package}
```

Example:

```
./ucs-ctl restore cluster gpu-test -b /home/ggz/gpu-test/backup-file-20230625164904.tar.gz
```

The faults on the master nodes are rectified if the following command output is displayed:

```
restore cluster successfully.
```

Restore the etcd data on each node. For details, see **4**.

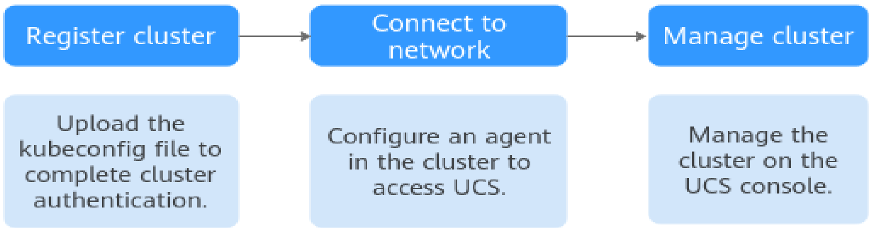
1.4 Attached Clusters

1.4.1 Overview

Attached clusters refer to third-party Kubernetes clusters that comply with the Cloud Native Computing Foundation (CNCF) standard, such as AWS EKS clusters, Google Cloud GKE clusters, and Kubernetes clusters that are deployed and run by third parties.

Figure 1-18 shows the attached cluster management process.

Figure 1-18 Attached cluster management process



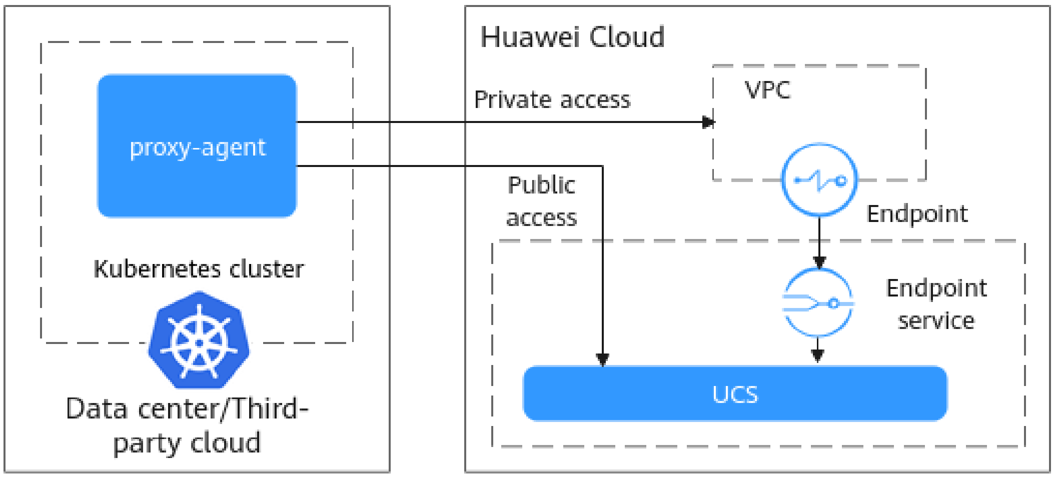
Access Mode

Cluster providers or on-premises data centers have different inbound port rules for attached clusters to prevent inbound traffic from ports other than the specific ones. UCS uses the cluster network agent to connect to clusters, as shown in **Figure 1-19**. You do not need to enable any inbound port on the firewall. Instead, only the cluster agent program is required to establish sessions with UCS in the outbound direction.

There are two methods with different advantages for attached clusters to connect to UCS:

- **Over a public network**: flexibility, cost-effectiveness, and easy access
- **Over a private network**: high speed, low latency, stability, and security

Figure 1-19 How clusters are connected to UCS



1.4.2 Registering an Attached Cluster (Public Network Access)

This section describes how to register an attached cluster and connect it to UCS over a public network.

Constraints

- A Huawei Cloud account must have the **UCS FullAccess** and **VPCEndpoint Administrator** permissions.
- If you are connecting a cluster outside the Chinese mainland to UCS, the connection and the subsequent actions you will take must comply with local laws and regulations.
- Registered Kubernetes clusters must pass the CNCF Certified Kubernetes Conformance Program and be between v1.19 and v1.28.

Prerequisites

- A cluster has been created and is running properly.
- The node where the proxy-agent component is deployed must be accessible from the public network through an EIP or a NAT gateway.
- You have obtained the kubeconfig file of the cluster. For guides of obtaining the kubeconfig file, see **kubeconfig**. For details about the kubeconfig file, see **Organizing Cluster Access Using kubeconfig Files**.

Registering a Cluster

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Fleets**. In the card view of **Attached cluster**, click **Register Cluster**.

Step 3 Enter the basic information of the cluster as listed in **Table 1-16**. The parameters marked with an asterisk (*) are mandatory.

Table 1-16 Basic information for registering a cluster

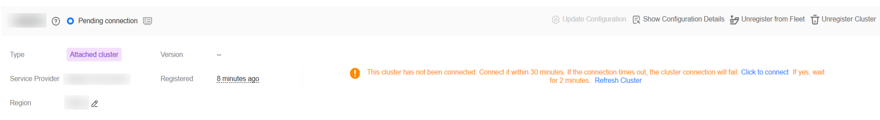
Parameter	Description
* Cluster Name	Enter a cluster name. Only digits, lowercase letters, and hyphens (-) are allowed, and the name must start with a lowercase letter and cannot end with a hyphen (-).
* Service Provider	Select a cluster service provider.
* Region	Select a region where the cluster is deployed.
Cluster Label	Optional. You can add labels in the form of key-value pairs to classify clusters. A key or value can contain a maximum of 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.
* kubeconfig	Upload the kubectl configuration file to complete cluster authentication. The file can be in JSON or YAML format. The procedure for obtaining the kubeconfig file varies according to vendors. For details, see **kubeconfig**.

Parameter	Description
* Context	Select the corresponding context. After the kubeconfig file is uploaded, the option list automatically obtains the **contexts** field from the file. The default value is the context specified by the **current-context** field in the kubeconfig file. If the file does not contain this field, you need to manually select a context from the list.
Fleets	Select the fleet that the cluster belongs to. A cluster can be added to only one fleet. Fleets are used for fine-grained access management. If you do not select a fleet, the cluster will be displayed on the **Clusters Not in Fleet** tab upon registration. You can add it to a fleet later. When registering a cluster, you cannot select a fleet with cluster federation enabled. To add your cluster to the fleet with cluster federation enabled, register your cluster with UCS first. For details about cluster federation, see **Enabling Cluster Federation**. For details about how to create a fleet, see **Managing Fleets**.

Step 4 Click **OK**. After the registration is complete, **Figure 1-20** is displayed. Connect the cluster to the network within 30 minutes. You can choose either the public or the private network access mode. For details about the network connection process, click in the upper right corner.

If the cluster is not connected to UCS within 30 minutes, it will fail to be registered. In this case, click in the upper right corner to register it again. If the cluster has been connected to UCS but no data is displayed, wait for 2 minutes and refresh the cluster.

Figure 1-20 Cluster waiting for network connection



----End

Connecting the Cluster to UCS

After the cluster is registered with UCS, its status is **Pending connection**. In this case, the network connection between UCS and the cluster is not established. You need to configure a network agent in the cluster.

Step 1 Log in to the UCS console.

Step 2 Click **Public access** in the row of the target cluster to download the configuration file of the cluster agent.

NOTE

The configuration file contains keys and can be downloaded only once. Keep the file secure.

Step 3 Use kubectl to connect to the cluster, run the following command to create a YAML file named **agent.yaml** (which can be changed as needed) in the cluster, and copy the agent configuration in **Step 2** and paste it to the YAML file:

vim agent.yaml

Step 4 Run the following command in the cluster to deploy the agent:

```
kubectl apply -f agent.yaml
```

Step 5 Check the deployment of the cluster agent.

```
kubectl -n kube-system get pod | grep proxy-agent
```

Expected output for successful deployment:

```
proxy-agent-5f7d568f6-6fc4k 1/1 Running 0 9s
```

Step 6 Check the running status of the cluster agent.

```
kubectl -n kube-system logs <Agent Pod Name> | grep "Start serving"
```

Expected log output for normal running:

```
Start serving
```

Step 7 Go to the UCS console and refresh the cluster status. The cluster is in the **Running** state.

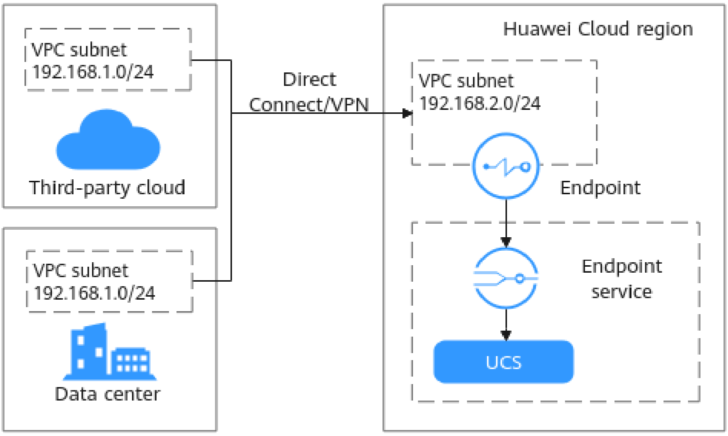
----End

1.4.3 Registering an Attached Cluster (Private Network Access)

Connecting attached clusters located in on-premises data centers or third-party clouds to UCS over public networks may cause security risks. To ensure stability and security, you can use private networks to connect the clusters to UCS for management.

The private network features high speed, low latency, and security. After you connect the on-premises network or the private network of a third-party cloud to the cloud network over Direct Connect or VPN, you can use a VPC endpoint to access UCS over the private network.

Figure 1-21 How clusters are connected to UCS over private networks



Constraints

- A Huawei Cloud account must have the **UCS FullAccess** and **VPCEndpoint Administrator** permissions.
- If you are connecting a cluster outside the Chinese mainland to UCS, the connection and the subsequent actions you will take must comply with local laws and regulations.
- Registered Kubernetes clusters must pass the CNCF Certified Kubernetes Conformance Program and be between v1.19 and v1.28.
- For attached clusters connected to UCS over private networks, the image repository may be restricted due to network restrictions.

  For clusters that are connected to UCS over a private network, images cannot be downloaded from SWR. Ensure that your nodes where your workloads run can access the public network.

Prerequisites

- A cluster has been created and is running properly.
- A VPC has been created in the region where UCS provides services. For details, see **Creating a VPC**.

  NOTE

  The subnet CIDR block of the VPC cannot overlap with the subnet CIDR blocks of on-premises data centers or third-party clouds. If the CIDR blocks overlap, the cluster cannot be connected to UCS. For example, if the subnet of an on-premises data center is 192.168.1.0/24, the subnet of the Huawei Cloud VPC cannot be 192.168.1.0/24.

- You have obtained the kubeconfig file of the cluster. For guides of obtaining the kubeconfig file, see **kubeconfig**. For details about the kubeconfig file, see **Organizing Cluster Access Using kubeconfig Files**.

Preparing the Network Environment

NOTICE

After the on-premises network or the private network of the third party cloud and the cloud network are connected, you are advised to ping the private IP address of a server in the VPC from an on-premises server or a server of the third-party cloud to check network connectivity.

Connect the on-premises network or the private network of the third party cloud to the cloud network.

- VPN: See **Connecting an On-Premises Data Center to a VPC Through a VPN**.
- Direct Connect: See **Accessing a VPC over a Single Connection Through Static Routes** or **Accessing a VPC over a Single Connection Through BGP Routes**.

Registering a Cluster

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Fleets**. In the card view of **Attached cluster**, click **Register Cluster**.

Step 3 Enter the basic information of the cluster as listed in **Table 1-17**. The parameters marked with an asterisk (*) are mandatory.

Table 1-17 Basic information for registering a cluster

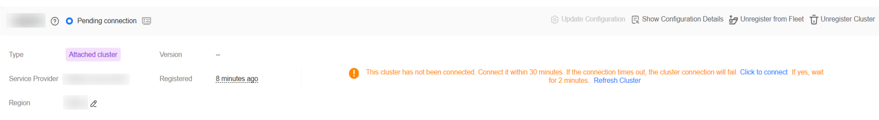
Parameter	Description
* Cluster Name	Enter a cluster name. Only digits, lowercase letters, and hyphens (-) are allowed, and the name must start with a lowercase letter and cannot end with a hyphen (-).
* Service Provider	Select a cluster service provider.
* Region	Select a region where the cluster is deployed.
Cluster Label	Optional. You can add labels in the form of key-value pairs to classify clusters. A key or value can contain a maximum of 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.
* kubeconfig	Upload the kubectl configuration file to complete cluster authentication. The file can be in JSON or YAML format. The procedure for obtaining the kubeconfig file varies according to vendors. For details, see **kubeconfig**.

Parameter	Description
* Context	Select the corresponding context. After the kubeconfig file is uploaded, the option list automatically obtains the **contexts** field from the file. The default value is the context specified by the **current-context** field in the kubeconfig file. If the file does not contain this field, you need to manually select a context from the list.
Fleets	Select the fleet that the cluster belongs to. A cluster can be added to only one fleet. Fleets are used for fine-grained access management. If you do not select a fleet, the cluster will be displayed on the **Clusters Not in Fleet** tab upon registration. You can add it to a fleet later. When registering a cluster, you cannot select a fleet with cluster federation enabled. To add your cluster to the fleet with cluster federation enabled, register your cluster with UCS first. For details about cluster federation, see **Enabling Cluster Federation**. For details about how to create a fleet, see **Managing Fleets**.

Step 4 Click **OK**. After the registration is complete, **Figure 1-22** is displayed. Connect the cluster to the network within 30 minutes. You can choose either the public or the private network access mode. For details about the network connection process, click in the upper right corner.

If the cluster is not connected to UCS within 30 minutes, it will fail to be registered. In this case, click in the upper right corner to register it again. If the cluster has been connected to UCS but no data is displayed, wait for 2 minutes and refresh the cluster.

Figure 1-22 Cluster waiting for network connection



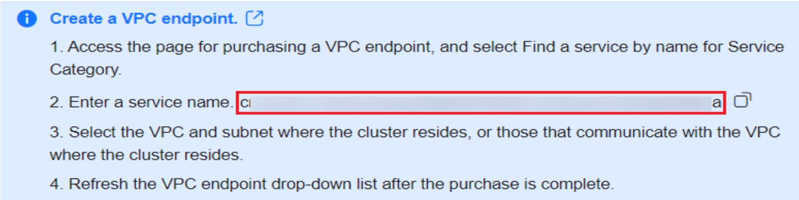
----End

Buying a VPC Endpoint

Step 1 Log in to the UCS console and click **Click to connect** in the card view of the cluster. In the window that slides out from the right, select **Private access**.

Step 2 In **Create a VPC Endpoint.**, click to record the service name.

Figure 1-23 Creating a VPC endpoint

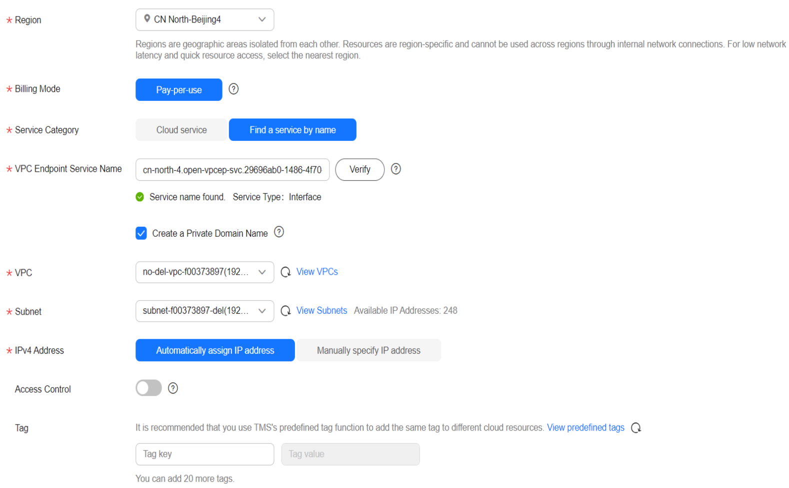


Step 3 Log in to the VPC Endpoint console and click **Create VPC Endpoint** to create a VPC endpoint for each service.

Step 4 Select the region that the VPC endpoint belongs to.

Step 5 Select **Find a service by name**, enter the service name recorded in **Step 2**, and click **Verify**.

Figure 1-24 Buying a VPC endpoint



Step 6 Select the VPC and subnet connected to the cluster network in **Preparing the Network Environment**.

Step 7 Select **Automatically assign IP address** or **Manually specify IP address** for assigning the private IP address of the VPC endpoint.

Step 8 After configuring other parameters, click **Next** and confirm the specifications.

- If the configuration is correct, click **Submit**.
- If any of the configurations is incorrect, click **Previous** to modify the parameters as needed, and click **Next** > **Submit**.

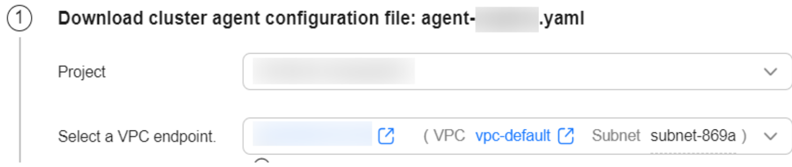
----End

Connecting to a Cluster

Step 1 Log in to the UCS console. In the card view of the target cluster in the **Pending connection** status, click **Private access**.

Step 2 Select a project. Select the VPC endpoint created in **Buying a VPC Endpoint**.

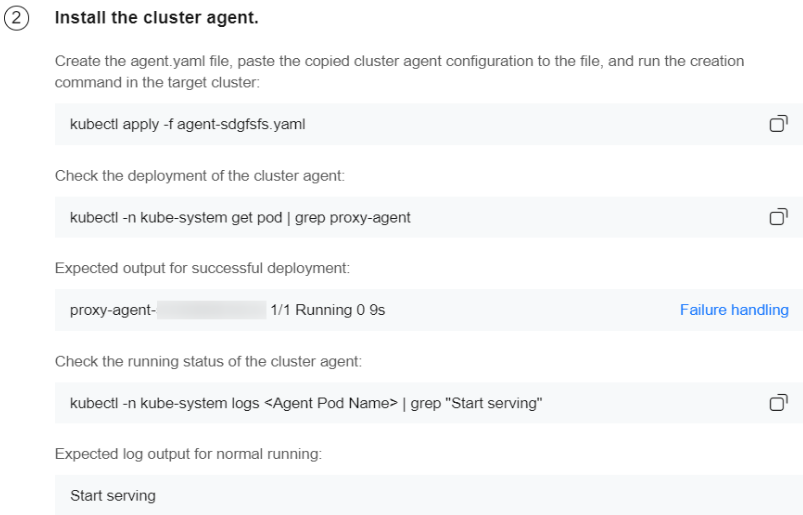
Figure 1-25 Selecting the VPC endpoint



Step 3 Upload the agent configuration file in **Step 2** to the node.

Step 4 Click **Configure Cluster Access** and run commands in the cluster. You can click on the right to copy each command.

Figure 1-26 Cluster agent installation configuration



NOTICE

- For clusters that are connected to UCS over a private network, images cannot be downloaded from SWR. Ensure that your nodes where your workloads run can access the public network.
- To pull the proxy-agent container image, the cluster must be able to access the public network, or the image can be uploaded to an image repository that can be accessed by the cluster. Otherwise, the image will fail to be deployed.

Step 5 Go to the UCS console and refresh the cluster status. The cluster is in the **Running** state.

----End

1.5 Multi-Cloud Clusters

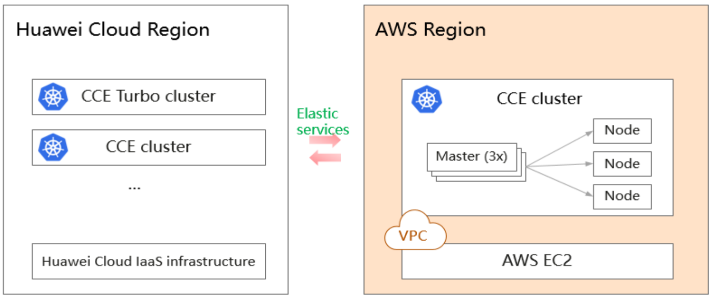
1.5.1 Overview

A multi-cloud cluster is a Kubernetes cluster provisioned by UCS but running on a third-party cloud (such as AWS). With this feature, you can create CCE clusters on

AWS or Azure. With multi-cloud clusters, you can deploy workloads to multiple clouds using unified APIs, tools, and configurations for a consistent management experience in a multi-cloud environment.

NOTE

Currently, multi-cloud clusters can be created only on AWS.



Access Mode

Multi-cloud clusters can be connected to UCS only over the public network. The public network features flexibility, cost-effectiveness, and easy access.

1.5.2 Service Planning for Multi-Cloud Cluster Installation

1.5.2.1 Basic Software Planning

Basic software, such as the OS and kernel, of the nodes must meet the version requirements listed in **Basic Software Planning**.

Table 1-18 Basic software planning

System Architecture	OS Type	Network Model	OS Version	Kernel Version
x86	Ubuntu 20.04	Cilium	Run **cat /etc/lsb-release** to check the version. `DISTRIB_DESCRIPTION="Ubuntu 20.04.4 LTS"`	Run **uname -r** to check the version. `5.15.0-1017-aws`

NOTE

- Cilium is a networking solution that supports network protocols such as BGP and eBPF. For details, see the **Cilium official documentation**.
- The multi-cloud cluster uses containerd as the container engine. If containerd and runC have been installed on the node running the OS, UCS directly uses them.

1.5.2.2 Data Planning

When you build a multi-cloud cluster on the AWS infrastructure, the following resources are automatically created on the AWS console. Ensure that the resource quota is sufficient.

Table 1-19 Resources quantity

Resource Type	EC2	NAT	VPC	Subnet	Route Table	Internet Gateway	EIP	Security Group	Network ACL	ELB	Network Port	Volume
Quantity	3	3	1	6	7	1	3	5	1	1	4	6

Table 1-20 EC2 resource specifications

Node Type	Quantity	CPU (vCPUs)	Mem (GiB)	Root Disk	Non-root Disk	Remarks
Cluster management nodes	3	8	32	100	200	t3.2xlarge
Cluster compute nodes	As required	8	32	100	200	You can increase the number of nodes as required.

Table 1-21 IAM permissions

Permission Type	Permission Name
IAMRole	AWSIAMRoleNodes, AWSIAMRoleControlPlane, and AWSIAMRoleControllers
IAMInstanceProfile	AWSIAMInstanceProfileNodes, AWSIAMInstanceProfileControl-Plane, and AWSIAMInstanceProfileControllers
IAMManagedPolicy	AWSIAMManagedPolicyCloudProviderNodes, AWSIAMManagedPolicyCloudProviderControlPlane, and AWSIAMManagedPolicyControllers

1.5.3 Registering a Multi-cloud Cluster

Register a UCS on AWS cluster. After the registration is complete, the cluster is automatically connected to UCS through the public network.

Constraints

Only **Huawei Cloud accounts** or users with AWS account permissions can register multi-cloud clusters.

Prerequisites

- You have applied for a multi-cloud cluster trial on the UCS console.
- The UCS cluster quota and AWS resource quota are sufficient.
- An access key has been created on the AWS console. See **How Do I Obtain an Access Key (AK/SK)?**

Procedure

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 Click **Register Cluster** in the card view of the multi-cloud cluster.

Step 3 Enter the basic information of the cluster to be registered as listed in the following table. The parameters marked with an asterisk (*) are mandatory.

Table 1-22 Parameter settings of registering a cluster

Parameter	Description
Cluster Type	Select **Multi-cloud cluster**.
Cloud Resource Provider	Select **AWS**.
Cluster Name	Enter a cluster name. Only digits, lowercase letters, and hyphens (-) are allowed, and the name must start with a lowercase letter and cannot end with a hyphen (-).
Region	Select the region where the cluster is located, that is, the AWS region. Ensure that the resource quota in the selected region is sufficient.
Version	Select **1.23**.
High Availability	Select **Yes**. Three EC2s will be automatically created as the master nodes of the cluster.
Cluster Label	Optional. You can add labels in the form of key-value pairs to classify clusters. A key or value can contain a maximum of 63 characters starting and ending with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.

Parameter	Description
Fleet	Select the fleet that the cluster belongs to. A cluster can be added to only one fleet. Fleets are used for fine-grained access management. If you do not select a fleet, the cluster will be displayed on the **Clusters Not in Fleet** tab upon registration. You can add it to a fleet later. When registering a cluster, you cannot select a fleet with cluster federation enabled. To add your cluster to the fleet with cluster federation enabled, register your cluster with UCS first. For details about cluster federation, see **Enabling Cluster Federation**. For details about how to create a fleet, see **Managing Fleets**.
Access Key ID*	Access key ID obtained from AWS IAM, that is, AccessKeyID.
Secret Access Key*	Secret access key obtained from AWS IAM, that is, SecretAccessKey.
Container CIDR Block*	Container CIDR block of the created Kubernetes cluster.
Service CIDR Block	Service CIDR block of the created Kubernetes cluster.

Step 4 Click **OK**. After the cluster is registered, wait for automatic connection.

----End

1.6 Single-Cluster Management

1.6.1 Overview

The UCS console allows you to manage each cluster on each cluster console.

- For Huawei Cloud clusters (CCE standard and CCE Turbo clusters), the operations on the cluster console are the same as those on the CCE console. For details about how to manage CCE clusters, see **CCE User Guide**.
- For attached clusters, on-premises clusters, and multi-cloud clusters, the cluster console enables you to manage basic Kubernetes resources, such as nodes, workloads, Services and Ingresses, container storage, ConfigMaps and secrets, and namespaces.

NOTICE

For attached clusters and on-premises clusters, you need to log in to the UCS console with a **Huawei Cloud account** or as a user who has the **UCS FullAccess** permission to configure permission policies on the **Permissions** page.

Accessing the Cluster Details Page

The method of accessing the cluster details page varies according to whether a cluster has been added to a fleet. The details are as follows:

- Cluster in fleet: On the **Fleets** page, click the **Fleets** tab and click the name of the target fleet. On the displayed page, choose **Container Clusters** in the navigation pane on the left and then click the cluster name to access the cluster console.
- Clusters not in fleet: Switch to the **Clusters Not in Fleet** tab, locate the cluster that is not added to the fleet, and click the cluster name to access the cluster console.

1.6.2 Nodes

1.6.2.1 Viewing Nodes in a Cluster

After a cluster is connected to UCS, you can access the cluster console from UCS to view the nodes in a cluster.

Procedure

Step 1 Access the cluster console.

Step 2 In the navigation pane on the left, choose **Nodes** to view the nodes in a cluster.

Step 3 Choose **More** > **View Pods** in the **Operation** column of the target node to view pods running on the current node.

Step 4 Click **View Events** to view node events.

Step 5 Choose **More** > **Disable Scheduling** in the **Operation** column of the target node to set the node as non-schedulable so that new pods cannot be scheduled to this node. For details about node taints, see **Adding Labels/Taints to Nodes**.

----End

1.6.2.2 Adding Labels/Taints to Nodes

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

Taints enable a node to repel specific pods to prevent these pods from being scheduled to the node, achieving reasonable allocation of workloads on nodes.

Node Label Usage Scenarios

Node labels are mainly used in the following scenarios:

- Node classification: 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 workloads in each module can be deployed on the corresponding node. In this way, workloads in modules do not interfere with each other and can be easily maintained on their nodes.

Inherent Node Labels

After a node is created, UCS adds labels to the node. These inherent labels cannot be edited or deleted. **Table 1-23** lists the inherent labels of a node.

Table 1-23 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.

Taint

Taints are in the format of **Key=Value:Effect**. **Key** and **Value** are the labels of a taint. **Value** can be empty. **Effect** is used to describe the effect of taints. The following options are supported for **Effect**:

- **NoSchedule**: No pod will be able to schedule onto the node unless it has a matching toleration, but existing pods will not be evicted from the node.
- **NoExecute**: Pods that cannot tolerate this taint cannot be scheduled onto the node, and existing pods will be evicted from the node.

Toleration

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node. This marks that the node should not accept any pods that do not tolerate the taints.

Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

In the preceding toleration label, **key** is **key1**, **value** is **value1**, and **effect** is **NoSchedule**. Therefore, the pod can be scheduled to the corresponding node.

The tolerance can also be set as follows, indicating that when a taint whose **key** is **key1** and **effect** is **NoSchedule** exists on a node, the pod can also be scheduled to the corresponding node.

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

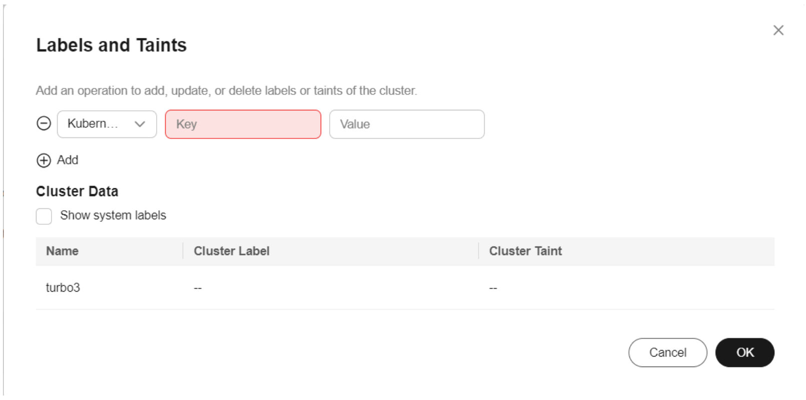
Managing Node Labels/Taints

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Nodes**, select the target node, and click **Manage Labels and Taints**.

Step 3 Click ⊕ to add a node label or taint. You can add a maximum of 10 operations at a time.

Figure 1-27 Adding labels or taints



- Choose **Add/Update** or **Delete**.
- Set the operation object to **Kubernetes Label** or **Taint**.
- Specify **Key** and **Value**.
- If you choose **Taint**, select a taint effect. For details, see **Taint**.

Step 4 Click **OK**.

----End

1.6.2.3 Creating and Deleting Nodes (Only for Multi-Cloud Clusters)

Viewing Nodes in a Cluster

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

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

Step 2 In the navigation pane on the left, choose **Nodes** to view the node information in the cluster.

Step 3 Choose **More** > **View Pods** in the **Operation** column of the target node to view pods running on the current node.

Step 4 Click **View Events** to view node events.

Step 5 Choose **More** > **Disable Scheduling** in the **Operation** column of the target node to set the node as non-schedulable so that new pods cannot be scheduled to this node.

----End

Creating a Node

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

Step 2 In the navigation pane on the left, choose **Nodes**. On the **Nodes** tab, click **Create Node** in the upper right corner.

Step 3 Enter a node name and select the required specifications. You can select the disk size and number of data disks as required.

Step 4 Click **Next: Confirm**.

Step 5 Confirm the specifications and click **Submit**. If you have any questions, click **Previous** to modify the specifications.

NOTE

- During node creation, the AZ, node type, container engine, OS, system disk, and data disk type cannot be selected.
- The number of data disks can be increased as required. A maximum of four data disks can be added.
- The default minimum size of a data disk or system disk is 100 GB.

----End

Deleting a Node

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

Step 2 In the navigation pane on the left, choose **Nodes**.

Step 3 Select the nodes to be deleted, click **More** above **Node Name**, and select **Delete** to delete nodes in batches.

Step 4 To delete a single node, click **More** of the target node and select **Delete**.

Step 5 Click **Yes**.

----End

1.6.3 Workload Management

1.6.3.1 Deployments

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

Basic Concepts

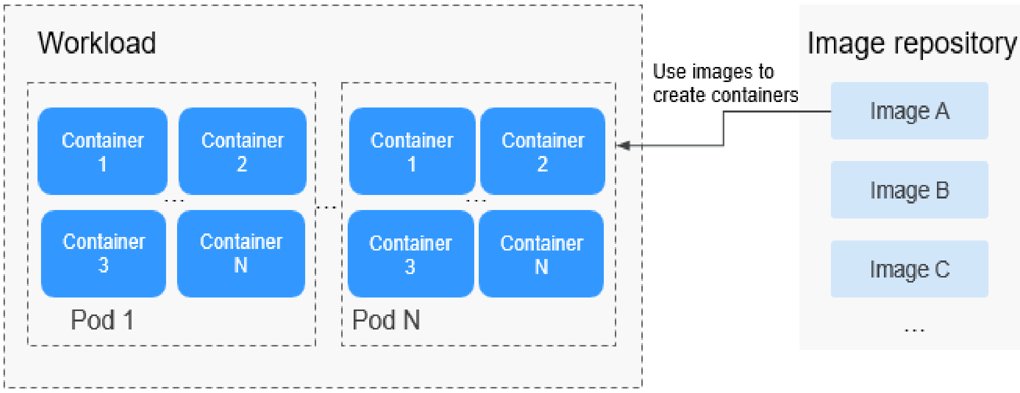
- Deployments: Pods are completely independent of each other and functionally identical. They feature auto scaling and rolling upgrade. Typical examples include Nginx and WordPress. For details on how to create a Deployment, see **Creating a Deployment**.

- 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**.
- DaemonSets: A DaemonSet runs a pod on each node in a cluster and ensures that there is only one pod. This works well for certain system-level applications, such as log collection and resource monitoring. For details on how to create a DeamonSet, see **Creating a DaemonSet**.

Relationship Between Workloads and Containers

As shown in **Figure 1-28**, 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 1-28 Relationship between workloads and containers



Workload Lifecycle

Table 1-24 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.
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 the image pulled from SWR, first upload your image to SWR. For details about how to upload an image, see **Image Management**. If you create a workload using an open source image, you do not need to upload the image to SWR.

Step 2 On the cluster details page, choose **Workloads** > **Deployments** and click **Create from Image**.

Step 3 Set basic workload parameters as described in **Table 1-25**. The parameters marked with an asterisk (*) are mandatory.

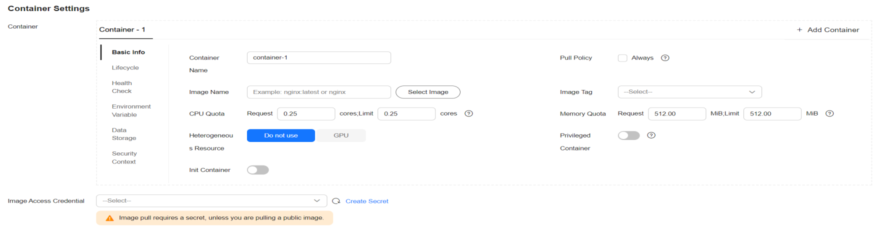
Table 1-25 Basic workload parameters

Parameter	Description
*Workload Name	Name of a workload, which must be unique.
Cluster Name	Cluster to which the workload belongs. You do not need to set this parameter.
*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.
Time Zone Synchronization	If this parameter is enabled, the containers and the node use the same time zone, and disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volumes** area. Do not modify or delete the disks.

Step 4 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 1-29 Container settings



- **Container Information**: Click **Add Container** on the right to configure multiple containers for the pod.
  - **Basic Info**: See **Table 1-26**.

Table 1-26 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. ▪ **My Images**: images in the image repository of the current region. If no image is available, click **Upload Image** to upload an image. ▪ **Open Source Images**: official images in the open source image repository. ▪ **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	▪ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ▪ **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.

Parameter	Description
Memory Quota	▪ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ▪ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

   - **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
   - **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
   - **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
   - **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Container Storage**.
   - **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 5 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Service Name**: Name of the Service to be added. It is customizable and must be unique.
- **Service Type**
  - **ClusterIP**: The Service is only reachable from within the cluster.
  - **NodePort**: The Service can be accessed from any node in the cluster.
  - **LoadBalancer**: The workload is accessed from the public network using a load balancer.
- **Service Affinity** (for NodePort and LoadBalancer only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.
- **Annotation**: The key-value pair format is supported. Configure annotations based on your service and vendor requirements and then click **Add**.

Step 6 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade**: upgrade mode of the Deployment, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
- **Replace upgrade**: Old pods are deleted before new pods are created. Services will be interrupted during a replace upgrade.

- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.
- **Toleration**: When the node where the workload pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes. By default, the toleration time window is 300s.
  - Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Example Tutorial**.
  - Click + under **Taints and Tolerations** to add a policy. For details about related parameters, see **Tolerance Policies**.

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

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

----End

Related Operations

On the cluster details page, you can also perform the operations described in **Table 1-27**.

Table 1-27 Related operations

Operation	Description
Creating a workload from a YAML file	Click **Create from YAML** in the upper right corner to create a workload from an existing YAML file.
Viewing pod details	Click the name of a workload. You can view pod details on the **Pods** tab page. • **View Events**: You can set search criteria, such as the time segment during which an event is generated or the event name, to view related events. • **View Container**: You can view the container name, status, image, and restarts of the pod. • **View YAML**: You can view the YAML file of the pod.

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target workload resides to edit its YAML file.
Upgrade	1. Click **Upgrade** in the row where the target workload resides. 2. Modify information about the workload. 3. Click **Upgrade Workload** to submit the modified information.
Rollback	Choose **More** > **Roll Back** in the row where the target workload resides, and select the target version for rollback.
Redeploy	Choose **More** > **Redeploy** in the row where the target workload resides, and click **Yes** in the dialog box displayed. Redeployment will restart all pods in the workload.
Disabling upgrade	Choose **More** > **Disable Upgrade** in the row where the workload resides, and click **Yes** in the dialog box displayed. ● After a workload is marked "Upgrade disabled", its upgrade will not be applied to the pods. ● Any ongoing rolling upgrade will be suspended.
Delete	Choose **More** > **Delete** in the row where the workload resides, and click **Yes** in the dialog box displayed.
Deleting workloads in batches	1. Select the target workloads to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

1.6.3.2 StatefulSets

Creating a StatefulSet

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

Step 2 On the cluster details page, choose **Workloads** > **StatefulSets** and click **Create from Image**.

Step 3 Set basic workload parameters as described in **Table 1-28**. The parameters marked with an asterisk (*) are mandatory.

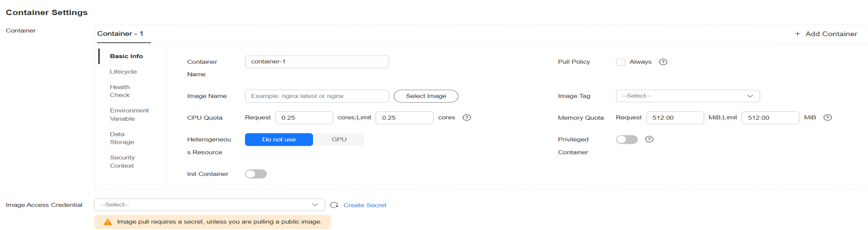
Table 1-28 Basic workload parameters

Parameter	Description
*Workload Name	Name of a workload, which must be unique.
Cluster Name	Cluster to which the workload belongs. You do not need to set this parameter.
*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.
Time Zone Synchronization	If this parameter is enabled, the containers and the node use the same time zone, and disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volumes** area. Do not modify or delete the disks.

Step 4 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 1-30 Container settings



- **Container Information**: Click **Add Container** on the right to configure multiple containers for the pod.
  - **Basic Info**: See **Table 1-29**.

Table 1-29 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. ■ **My Images**: images in the image repository of the current region. If no image is available, click **Upload Image** to upload an image. ■ **Open Source Images**: official images in the open source image repository. ■ **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

Parameter	Description
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

- **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
- **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
- **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
- **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Container Storage**.
- **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.

- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 5 Configure the headless Service parameters for the workload.

StatefulSet pods discover each other through headless Services. No cluster IP is allocated for a headless Service, and the DNS records of all pods are returned during query. In this way, the IP addresses of all pods can be queried.

- **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**
  - **Port**: Name of the container port. You are advised to enter a name that indicates the function of the port.
  - **Service Port**: Port of the Service.
  - **Container Port**: Listening port of the container.

Step 6 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Service Name**: Name of the Service to be added. It is customizable and must be unique.
- **Service Type**
  - **ClusterIP**: The Service is only reachable from within the cluster.
  - **NodePort**: The Service can be accessed from any node in the cluster.
  - **LoadBalancer**: The workload is accessed from the public network using a load balancer.
- **Service Affinity** (for NodePort and LoadBalancer only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.
- **Annotation**: The key-value pair format is supported. Configure annotations based on your service and vendor requirements and then click **Add**.

Step 7 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade**: Upgrade mode of the StatefulSet, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
- **Replace upgrade**: You need to delete old pods manually before new pods are created. Services will be interrupted during a replace upgrade.

- **Pod Management Policies**
  - **OrderedReady**: The StatefulSet will launch, terminate, or scale pods sequentially. It will wait for the state of the pods to change to Running and Ready or completely terminated before it launches or terminates another pod.
  - **Parallel**: The StatefulSet will launch or terminate all pods in parallel. It will not wait for the state of the pods to change to Running and Ready or completely terminated before it launches or terminates another pod.
- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.
- **Toleration**: When the node where the workload pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes. By default, the toleration time window is 300s.
  - Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Example Tutorial**.
  - Click + to add a policy. For details about related parameters, see **Tolerance Policies**.

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

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

----End

Related Operations

On the cluster details page, you can also perform the operations described in **Table 1-30**.

Table 1-30 Related operations

Operation	Description
Creating a workload from a YAML file	Click **Create from YAML** in the upper right corner to create a workload from an existing YAML file.

Operation	Description
Viewing pod details	Click the name of a workload. You can view pod details on the **Pods** tab page. ● **View Events**: You can set search criteria, such as the time segment during which an event is generated or the event name, to view related events. ● **View Container**: You can view the container name, status, image, and restarts of the pod. ● **View YAML**: You can view the YAML file of the pod.
Editing a YAML file	Click **Edit YAML** in the row where the target workload resides to edit its YAML file.
Upgrade	1. Click **Upgrade** in the row where the target workload resides. 2. Modify information about the workload. 3. Click **Upgrade Workload** to submit the modified information.
Rollback	Choose **More** > **Roll Back** in the row where the target workload resides, and select the target version for rollback.
Redeploy	Choose **More** > **Redeploy** in the row where the target workload resides, and click **Yes** in the dialog box displayed. Redeployment will restart all pods in the workload.
Disabling upgrade	Choose **More** > **Disable Upgrade** in the row where the workload resides, and click **Yes** in the dialog box displayed. ● After a workload is marked "Upgrade disabled", its upgrade will not be applied to the pods. ● Any ongoing rolling upgrade will be suspended.
Delete	Choose **More** > **Delete** in the row where the workload resides, and click **Yes** in the dialog box displayed.
Deleting workloads in batches	1. Select the target workloads to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

1.6.3.3 DaemonSets

Creating a DaemonSet

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

Step 2 On the cluster details page, choose **Workloads** > **DaemonSets** and click **Create from Image**.

Step 3 Set basic workload parameters as described in **Table 1-31**. The parameters marked with an asterisk (*) are mandatory.

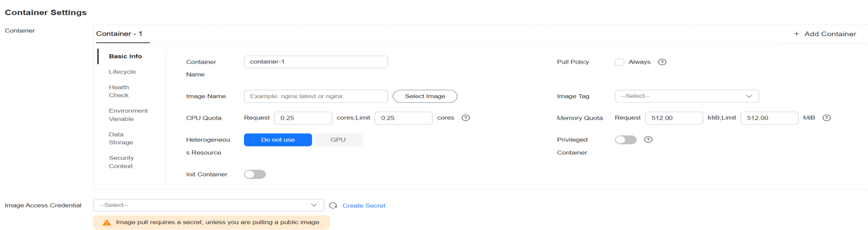
Table 1-31 Basic workload parameters

Parameter	Description
*Workload Name	Name of a workload, which must be unique.
Cluster Name	Cluster to which the workload belongs. You do not need to set this parameter.
*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.
Description	Description of the workload.
Time Zone Synchronization	If this parameter is enabled, the containers and the node use the same time zone, and disks of the hostPath type will be automatically added and listed in the **Data Storage** > **Local Volumes** area. Do not modify or delete the disks.

Step 4 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 1-31 Container settings



- **Container Information**: Click **Add Container** on the right to configure multiple containers for the pod.

– **Basic Info**: See **Table 1-32**.

Table 1-32 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. ■ **My Images**: images in the image repository of the current region. If no image is available, click **Upload Image** to upload an image. ■ **Open Source Images**: official images in the open source image repository. ■ **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

Parameter	Description
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

   - **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
   - **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
   - **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
   - **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Container Storage**.
   - **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 5 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Service Name**: Name of the Service to be added. It is customizable and must be unique.
- **Service Type**
   - **ClusterIP**: The Service is only reachable from within the cluster.

  - **NodePort**: The Service can be accessed from any node in the cluster.
  - **LoadBalancer**: The workload is accessed from the public network using a load balancer.
- **Service Affinity** (for NodePort and LoadBalancer only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.
- **Annotation**: The key-value pair format is supported. Configure annotations based on your service and vendor requirements and then click **Add**.

Step 6 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade**: upgrade mode of the DaemonSet, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
  - **Replace upgrade**: You need to delete old pods manually before new pods are created. Services will be interrupted during a replace upgrade.
- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.
- **Toleration**: When the node where the workload pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes. By default, the toleration time window is 300s.

– Using both taints and tolerations allows (not forcibly) the pod to be scheduled to a node with the matching taints, and controls the pod eviction policies after the node where the pod is located is tainted. For details, see **Example Tutorial**.

– Click + to add a policy. For details about related parameters, see **Tolerance Policies**.

Step 7 After the configuration is complete, click **Create Workload**. You can view the DaemonSet status in the DaemonSet List.

If the DaemonSet is in the **Running** status, the DaemonSet is successfully created.

----End

Related Operations

On the cluster details page, you can also perform the operations described in **Table 1-33**.

Table 1-33 Related operations

Operation	Description
Creating a workload from a YAML file	Click **Create from YAML** in the upper right corner to create a workload from an existing YAML file.
Viewing pod details	Click the name of a workload. You can view pod details on the **Pods** tab page. ● **View Events**: You can set search criteria, such as the time segment during which an event is generated or the event name, to view related events. ● **View Container**: You can view the container name, status, image, and restarts of the pod. ● **View YAML**: You can view the YAML file of the pod.
Editing a YAML file	Click **Edit YAML** in the row where the target workload resides to edit its YAML file.
Upgrade	1. Click **Upgrade** in the row where the target workload resides. 2. Modify information about the workload. 3. Click **Upgrade Workload** to submit the modified information.
Rollback	Choose **More** > **Roll Back** in the row where the target workload resides, and select the target version for rollback.

Operation	Description
Redeploy	Choose **More** > **Redeploy** in the row where the target workload resides, and click **Yes** in the dialog box displayed. Redeployment will restart all pods in the workload.
Disabling upgrade	Choose **More** > **Disable Upgrade** in the row where the workload resides, and click **Yes** in the dialog box displayed. ● After a workload is marked "Upgrade disabled", its upgrade will not be applied to the pods. ● Any ongoing rolling upgrade will be suspended.
Delete	Choose **More** > **Delete** in the row where the workload resides, and click **Yes** in the dialog box displayed.
Deleting workloads in batches	1. Select the target workloads to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

1.6.3.4 Jobs and Cron Jobs

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 a crontab 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 On the cluster details page, choose **Workloads** > **Jobs** and click **Create from Image**.

Step 3 Set basic workload parameters.

Basic Info

- **Workload Type**: Select **Job**.
- **Workload Name**: Enter a workload name.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.
- **Pods**: Enter the number of pods.

Container Settings

- **Container Information**: Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.
  - **Basic Info**: See **Table 1-34**.

Table 1-34 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. ▪ **My Images**: images in the image repository of the current region. If no image is available, click **Upload Image** to upload an image. ▪ **Open Source Images**: official images in the open source image repository. ▪ **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.

Parameter	Description
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

- **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
- **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.

- **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Container Storage**.

- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Advanced Settings

- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.
- **Job Settings**
  - **Parallel Pods**: Maximum number of pods that can run in parallel during job execution. The value cannot be greater than the total number of pods in the job.
  - **Timeout (s)**: Once a job reaches this time, the job status becomes failed and all pods in this job will be deleted. If you leave this parameter blank, the job will never time out.

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

If the status of the job is **Processing**, 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 On the cluster details page, choose **Workloads** > **Cron Jobs**, and click **Create Workload**.

Step 3 Configure workload parameters.

Basic Info

- **Workload Type**: Select **Cron Job**.
- **Workload Name**: Enter a workload name.
- **Namespace**: Select the namespace of the workload. The default value is **default**. You can also click **Create Namespace** to create one. For details, see **Creating a Namespace**.

Container Settings

- **Container Information**: Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

– **Basic Info**: See **Table 1-35**.

Table 1-35 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. ■ **My Images**: images in the image repository of the current region. If no image is available, click **Upload Image** to upload an image. ■ **Open Source Images**: official images in the open source image repository. ■ **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	■ **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ■ **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	■ **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ■ **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

Parameter	Description
Privileged Container	Programs in a privileged container have certain privileges. If **Privileged Container** is enabled, the container is assigned privileges. For example, privileged containers can manipulate network devices on the host machine and modify kernel parameters.

  - **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
  - **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Execution Settings

- **Concurrency Policy**: The following three modes are supported:
  - **Forbid**: A new job cannot be created before the previous job is completed.
  - **Allow**: The cron job allows concurrently running jobs, which preempt cluster resources.
  - **Replace**: If it is time for a new job run and the previous job run has not finished yet, the cron job replaces the currently running job run with a new job run.
- **Policy Settings**: Time when a new cron job is executed. Scheduled rules in YAML are implemented using the cron expression.
  - A cron job is executed at a fixed interval. The unit can be minute, hour, day, or month. For example, if a cron job is executed every 30 minutes and the corresponding cron expression is ***/30 * * * ***, the execution time starts from 0 in the unit range, for example, **00:00:00**, **00:30:00**, **01:00:00**, and **...**.
  - The cron job is executed by month. For example, if a cron job is executed at 00:00 on the first day of each month, the corresponding cron expression is **0 0 1 */1 ***, and the execution time is ******-01-01 00:00:00**, ******-02-01 00:00:00**, and **...**.
  - The cron job is executed by week. For example, if a cron job is executed at 00:00 every Monday, the corresponding cron expression is **0 0 * * 1**, and the execution time is ******-**-01 00:00:00 on Monday**, ******-**-08 00:00:00 on Monday**, and **...**.

– **Custom Cron Expression**: For details about how to use cron expressions, see **cron**.

NOTE

– If a cron job is executed at a fixed time (by month) and the number of days in a month does not exist, the job will not be executed that month. For example, the execution will skip February if the date is set to 30.

– Due to the definition of cron, the fixed period is not a strict period. The time is divided starting from 0 by period. For example, if the unit is minute, the value ranges from 0 to 59. If the value cannot be exactly divided, the last period will be reset. Therefore, an accurate period can be represented only when the period can be evenly divided.

Take a cron job that is executed by hour as an example. As **/2, /3, /4, /6, /8, and /12** can exactly divide 24 hours, an accurate period can be represented. If another period is used, the last period will be reset at the beginning of a new day. For example, if the cron expression is ** */12 * * ***, the execution time is **00:00:00** and **12:00:00** every day. If the cron expression is ** */13 * * ***, the execution time is **00:00:00** and **13:00:00** every day. At 00:00 on the next day, the execution time is updated even if the period does not reach 13 hours.

- **Job Records**: You can set the number of jobs that are successfully executed or fail. Setting a limit to **0** corresponds to keeping none of the jobs after they are completed.

Advanced Settings

- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.

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

Related Operations

- **View Events**: You can set search criteria, such as the time segment during which an event is generated or the event name, to view related events.
- **Pods/Jobs**: View the information about the target pod/job.
  – **View Events**: Event information generated by the pod, which is stored for one hour.
  – **Pods**: View the pod name, status, and restart times.
  – **View YAML**: View the YAML file of the pod.
  – **Delete**: Delete the pod.
- **View/Edit YAML**: View or edit the YAML file of the workload.
- **Delete**: Delete the workload.
- **Stop** (for cron jobs only): Stop a cron job.

1.6.3.5 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 from YAML**.

Step 2 On the displayed **Create from YAML** page, edit the YAML file.

Step 3 Click **OK**.

----End

Related Operations

- **View Events**: You can set search criteria, such as the time segment during which an event is generated or the event name, to view related events.
- **View Container**: You can view the container name, status, image, and restarts of the pod.
- **View YAML**: You can view the YAML file of the pod.

1.6.3.6 Setting Container Specifications

Scenario

UCS allows you to set resource limits for added containers during workload creation. You can apply for and limit the CPU and memory quotas used by each pod in the workload.

Meanings

The meanings of requests and limits for CPU and memory are as follows:

- Requests are the minimum guaranteed amount of a resource that is reserved for containers in a pod. If the node where the pod is running does not have enough of that resource, the containers fail to be created.
- Limits are the maximum amount of a resource to be used by containers. You can specify the resource limit for a container to prevent the container from using more of that resource than the limit you set or being evicted due to node resource exhaustion.

NOTE

When creating a workload, you are advised to set the upper and lower limits of CPU and memory resources. If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

Configuration Description

- CPU quotas:

Table 1-36 Description of CPU quotas

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

Recommended configuration

Actual available CPU of a node ≥ Sum of CPU limits of all containers on the current node ≥ Sum of CPU requests of all containers on the current node. You can view the actual available CPUs of a node on the CCE console (**Resource Management** > **Nodes** > **Allocatable**).

- Memory quotas:

Table 1-37 Description of memory quotas

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

Recommended configuration

Actual available memory of a node ≥ Sum of memory limits of all containers on the current node ≥ Sum of memory requests of all containers on the current node. You can view the actual available memory of a node on the CCE console (**Resource Management** > **Nodes** > **Allocatable**).

NOTE

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node. The calculation formula is as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

Example

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

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

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

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

1.6.3.7 Setting Container Lifecycle Parameters

Scenario

UCS provides callback functions (hooks) for the lifecycle management of containerized applications. For example, if you want a container to perform a certain operation before stopping, you can register a hook.

UCS provides the following lifecycle callback functions:

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

Startup Commands

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

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, CCE runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

Table 1-38 Commands and arguments used to run a container

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

Step 1 When creating a workload, select **Lifecycle** under **Container Settings**.

Step 2 Enter a command and arguments on the **Startup Command** tab page.

Table 1-39 Container startup commands

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

----End

Post-Start Processing

Step 1 When creating a workload, select **Lifecycle** under **Container Settings**.

Step 2 Set the post-start processing parameters on the **Post-Start** tab page.

Table 1-40 Post-start processing parameters

Parameter	Description

CLI	Run a specified command in the container using either the bash or binary mode. You can configure the command by referring to the example.
	The command format is **Command Args[1] Args[2]...**. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution. **Commands that are executed in the background or asynchronously are not supported.**
	Example command:
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	Enter **/install install_agent** in the script. This command indicates that **install.sh** will be executed after the container is created successfully.

HTTP request	Send an HTTP request for post-start processing. The related parameters are described as follows: ● **Path**: (optional) request URL. ● **Port**: (mandatory) request port. ● **Host**: (optional) IP address of the request. The default value is the IP address of the node where the container resides.

----End

Pre-Stop Processing

Step 1 When creating a workload, select **Lifecycle** under **Container Settings**.

Step 2 Set the pre-start processing parameters on the **Pre-Stop** tab page.

Table 1-41 Pre-stop processing parameters

Parameter	Description

CLI	Run a specified command in the container using either the bash or binary mode. You can configure the command by referring to the example.
	The command format is **Command Args[1] Args[2]...**. **Command** is a system command or a user-defined executable program. If no path is specified, an executable program in the default path will be selected. If multiple commands need to be executed, write the commands into a script for execution.
	Example command:
	`exec:`
	`  command:`
	`  - /uninstall.sh`
	`  - uninstall_agent`
	Enter **/uninstall uninstall_agent** in the script. This command indicates that the **uninstall.sh** script will be executed before the container completes its execution and stops running.

HTTP request	Send an HTTP request for pre-stop processing. The related parameters are described as follows: ● **Path**: (optional) request URL. ● **Port**: (mandatory) request port. ● **Host**: (optional) IP address of the request. The default value is the IP address of the node where the container resides.

----End

YAML Example

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
```

```
- image: nginx
  command:
  - sleep 3600                 #Startup command
  imagePullPolicy: Always
  lifecycle:
    postStart:
      exec:
        command:
        - /bin/bash
        - install.sh              #Post-start command
    preStop:
      exec:
        command:
        - /bin/bash
        - uninstall.sh              #Pre-stop command
  name: nginx
imagePullSecrets:
- name: default-secret
```

1.6.3.8 Setting Health Check for a Container

Scenarios

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

Kubernetes provides the following health check probes:

- **Liveness probe** (livenessProbe): checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness check of a container fails, the cluster restarts the container. If the liveness check is successful, no operation is executed.
- **Readiness probe** (readinessProbe): checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some applications to start up before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, the application process is running, but the application cannot provide services. To address this issue, this health check probe is used. If the container readiness check fails, the cluster masks all requests sent to the container. If the container readiness check is successful, the container can be accessed.
- **Startup probe** (startupProbe): checks when a containerized application has started. If such a probe is configured, it disables liveness and readiness checks until it succeeds, ensuring that those probes do not interfere with the application startup. This can be used to perform liveness checks on slow starting containers to prevent them from getting terminated by the kubelet before they are started.

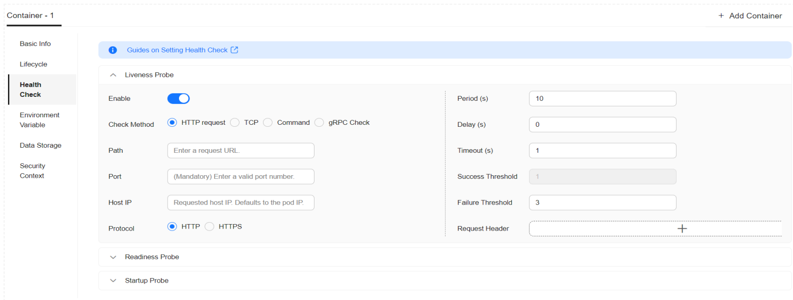
Check Methods

- **HTTP request**

  This health check mode can be used for containers that provide HTTP/HTTPS services. The cluster periodically initiates an HTTP/HTTPS GET request to such containers. If the return code of the HTTP/HTTPS response is within 200–399, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port and an HTTP/HTTPS request path.

For example, for a container that provides HTTP services, the HTTP check path is **/health-check**, the port is 80, and the host address is optional (which defaults to the container IP address). Here, 172.16.0.186 is used as an example, and we can get such a request: GET http://172.16.0.186:80/health-check. The cluster periodically initiates this request to the container.

Figure 1-32 HTTP request–based check

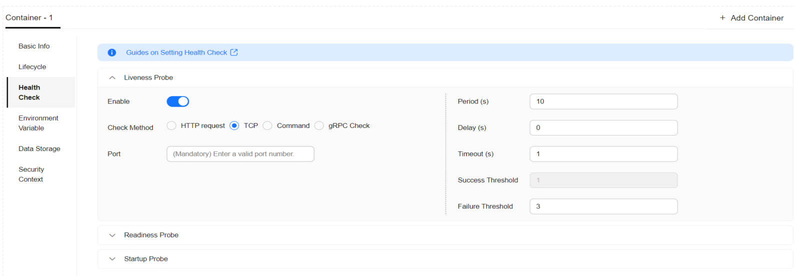


- **TCP port**

  For a container that provides TCP communication services, the cluster periodically establishes a TCP connection to the container. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port.

  For example, if you have a Nginx container with service port 80, after you specify TCP port 80 for container listening, the cluster will periodically initiate a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

Figure 1-33 TCP port–based check



- **CLI**

  CLI is an efficient tool for health check. When using the CLI, you must specify an executable command in a container. The cluster periodically runs the command in the container. If the command output is 0, the health check is successful. Otherwise, the health check fails.

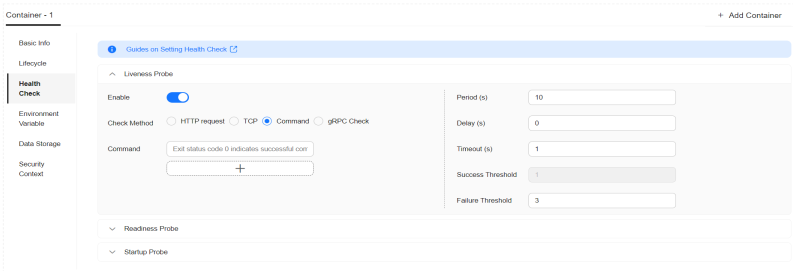
  The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

  - For a TCP port, you can use a script to connect to a container port. If the connection is successful, the script returns **0**. Otherwise, the script returns **–1**.
  - For an HTTP request, you can use a script to run the **wget** command for a container.

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

Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **–1**.

Figure 1-34 Command–based check



NOTICE

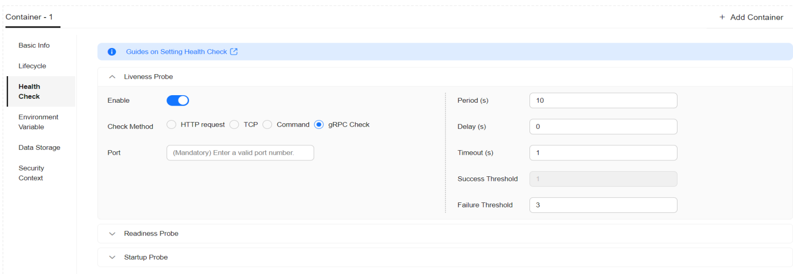
- Put the program to be executed in the container image so that the program can be executed.
- If the command to be executed is a shell script, do not directly specify the script as the command, but add a script parser. For example, if the script is **/data/scripts/health_check.sh**, the program is as follows:

```
sh
/data/scripts/health_check.sh
```

- **gRPC check**

  This health check mode allows you to configure startup, liveness, and readiness probes for your gRPC application without exposing any HTTP endpoint or using an executable. Kubernetes can connect to your workload via gRPC and obtain its status.

Figure 1-35 gRPC–based check



NOTICE

- To use the gRPC check, your application must support the **gRPC health checking protocol**.
- Similar to HTTP and TCP probes, if the port is incorrect, the application does not support the health checking protocol, or there are another configuration error, the check will fail.

Common Parameters

Table 1-42 Common parameter description

Parameter	Description
Period (periodSeconds)	Indicates the probe detection period, in seconds. For example, if this parameter is set to **30**, the detection is performed every 30 seconds.
Delay (initialDelaySec-onds)	Check delay time in seconds. Set this parameter according to the normal startup time of services. For example, if this parameter is set to **30**, the health check will be started 30 seconds after the container is started. The time is reserved for containerized services to start.
Timeout (timeoutSeconds)	Number of seconds after which the probe times out. Unit: second. For example, if this parameter is set to **10**, the timeout wait time for performing a health check is 10s. If the wait time elapses, the health check is regarded as a failure. If the parameter is left blank or set to **0**, the default timeout time is 1s.
Success Threshold (successThreshold)	Minimum consecutive successes for the probe to be considered successful after having failed. The default value is **1**, which is also the minimum value. The value of this parameter is fixed to **1** in **Liveness Probe** and **Startup Probe**.
Failure Threshold (failureThreshold)	Number of retry times when the detection fails. Giving up in case of liveness probe means to restart the container. In case of readiness probe the pod will be marked **Unready**. The default value is **3**, and the minimum value is **1**.

YAML Example

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
```

```
      httpHeaders:
      - name: Custom-Header
        value: Awesome
    initialDelaySeconds: 3
    periodSeconds: 3
  readinessProbe:
    exec:
      command:
        - cat
        - /tmp/healthy
    initialDelaySeconds: 5
    periodSeconds: 5
  startupProbe:
    httpGet:
      path: /healthz
      port: 80
    failureThreshold: 30
    periodSeconds: 10
```

1.6.3.9 Setting Environment Variables

Scenario

An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

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

NOTICE

After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

Environment variables can be set in the following modes:

- **Custom**: Enter a variable name and value.
- **Added from ConfigMap**: Import all keys in a ConfigMap as environment variables.
- **Added from ConfigMap key**: Import a key in a ConfigMap as the value of an environment variable. For example, if you import **configmap_value** of **configmap_key** in ConfigMap **configmap-example** as the value of environment variable **key1**, an environment variable named **key1** with its value **is configmap_value** exists in the container.
- **Added from secret**: Import all keys in a secret as environment variables.
- **Added from secret key**: Import the value of a key in a secret as the value of an environment variable. For example, if you import **secret_value** of **secret_key** in secret **secret-example** as the value of environment variable

key2, an environment variable named **key2** with its value **secret_value** exists in the container.

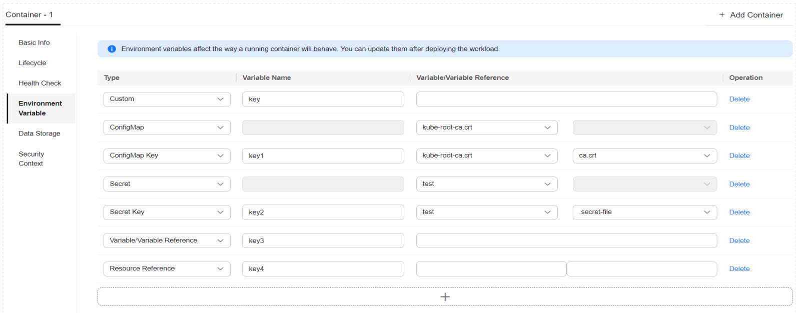
- **Variable Value/Reference**: Use the field defined by a pod as the value of the environment variable, for example, the pod name.
- **Resource Reference**: Use the field defined by a container as the value of the environment variable, for example, the CPU limit of the container.

Adding Environment Variables

Step 1 When creating a workload, select **Environment Variables** under **Container Settings**.

Step 2 Set environment variables.

Figure 1-36 Adding environment variables



----End

YAML Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          env:
            - name: key                     # Custom
              value: value
            - name: key1                     # Added from ConfigMap key
```

```
            valueFrom:
              configMapKeyRef:
                name: configmap-example
                key: key1
          - name: key2                    # Added from secret key
            valueFrom:
              secretKeyRef:
                name: secret-example
                key: key2
          - name: key3                    # Variable reference, which uses the field defined by a pod as the value
of the environment variable.
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: key4                    # Resource reference, which uses the field defined by a container as the
value of the environment variable.
            valueFrom:
              resourceFieldRef:
                containerName: container1
                resource: limits.cpu
                divisor: 1
        envFrom:
          - configMapRef:                 # Added from ConfigMap
              name: configmap-example
          - secretRef:                    # Added from secret
              name: secret-example
      imagePullSecrets:
        - name: default-secret
```

Viewing Environment Variables

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

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

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

The environment variables in the pod are as follows:

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

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

1.6.3.10 Configuring a Workload Upgrade Policy

In actual applications, upgrade is a common operation. A Deployment, StatefulSet, or DaemonSet can easily support application upgrade.

Configuring the Workload Upgrade Policy on the Console

Step 1 When creating a workload, click **Expand**.

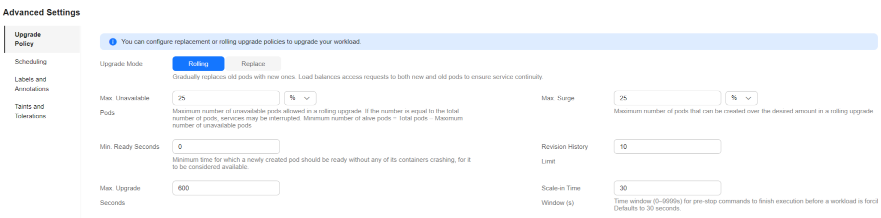
Step 2 Configure the workload upgrade policy based on **Table 1-43**.

Table 1-43 Parameters for configuring the workload upgrade policy

Parameter	Description
Upgrade Mode	You can set different upgrade policies: ● **Rolling upgrade**: New pods are created gradually and then old pods are deleted. This is the default policy. ● **Replace upgrade**: The current pods are deleted and then new pods are created.
Max. Unavailable Pods (maxUnavailable)	Specifies the maximum number of pods that can be unavailable during the upgrade process. The default value is 25%. For example, if **spec.replicas** is set to **4**, at least 3 pods exist during the upgrade process. The deletion step is 1. The value can also be set to an absolute number. This parameter is only available for Deployments.
Max. Surge (maxSurge)	Specifies the maximum number of pods that can exist over **spec.replicas**. The default value is 25%. For example, if **spec.replicas** is set to **4**, no more than 5 pods can exist during the upgrade process, that is, the upgrade step is 1. The absolute number is calculated from the percentage by rounding up. The value can also be set to an absolute number. This parameter is only available for Deployments.
Min. Ready Seconds (minReadySeconds)	A pod is considered available only when the minimum readiness time is exceeded without any of its containers crashing. The default value is **0** (the pod is considered available immediately after it is ready).

Parameter	Description
Revision History Limit (revisionHistoryLimit)	Specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of **kubectl get rs**. The configuration of each Deployment revision is stored in its ReplicaSets. Therefore, once the old ReplicaSet is deleted, you lose the ability to roll back to that revision of the Deployment. By default, 10 old ReplicaSets will be kept, but the ideal value depends on the frequency and stability of the new Deployments.
Max. Upgrade Duration (progressDeadlineSeconds)	Specifies the number of seconds that the system waits for a Deployment to make progress before reporting a Deployment progress failure. It is surfaced as a condition with Type=Progressing, Status=False, and Reason=ProgressDeadlineExceeded in the status of the resource. The Deployment controller will keep retrying the Deployment. In the future, once automatic rollback will be implemented, the Deployment controller will roll back a Deployment as soon as it observes such a condition. If this parameter is specified, the value of this parameter must be greater than that of **.spec.minReadySeconds**.
Scale-In Time Window (terminationGracePeriodSeconds)	Graceful deletion time. The default value is 30 seconds. When a pod is deleted, a SIGTERM signal is sent and the system waits for the applications in the container to terminate. If the application is not terminated within the time specified by **terminationGracePeriodSeconds**, a SIGKILL signal is sent to forcibly terminate the pod.

Figure 1-37 Upgrade policies



----End

Rolling Back the Workload Version on the Console

Rollback is to roll an application back to the source version when a fault occurs during the upgrade. A Deployment can be easily rolled back to the source version.

Step 1 On the cluster details page, choose **Workloads** and click the name of the workload to be rolled back.

Step 2 Click the **Change History** tab, locate the target version, click **Roll Back to This Version**, and click **OK**. Wait until the workload version is rolled back.

----End

Configuring the Workload Upgrade Policy Using the CLI

The Deployment can be upgraded in a declarative mode. That is, you only need to modify the YAML definition of the Deployment. For example, you can run the **kubectl edit** command to change the Deployment image to **nginx:alpine**. After the modification, query the ReplicaSet and pod. The query result shows that a new ReplicaSet is created and the pod is re-created.

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME               DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd   2         2         2         1m
nginx-7f98958cdf   0         0         0         48m

$ kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

The Deployment can use the **maxSurge** and **maxUnavailable** parameters to control the proportion of pods to be re-created during the upgrade, which is useful in many scenarios. The configuration is as follows:

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

In the preceding example, the value of **spec.replicas** is **2**. If both **maxSurge** and **maxUnavailable** are the default value 25%, **maxSurge** allows a maximum of three pods to exist (2 x 1.25 = 2.5, rounded up to 3), and **maxUnavailable** does not allow a maximum of two pods to be unavailable (2 x 0.75 = 1.5, rounded up to 2). That is, during the upgrade process, there will always be two pods running. Each time a new pod is created, an old pod is deleted, until all pods are new.

Rolling Back the Workload Version Using the CLI

For example, if the upgraded image is faulty, you can run the **kubectl rollout undo** command to roll back the Deployment.

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

A Deployment can be easily rolled back because it uses a ReplicaSet to control a pod. After the upgrade, the previous ReplicaSet still exists. The Deployment is rolled back by using the previous ReplicaSet to re-create the pod. The number of ReplicaSets stored in a Deployment can be restricted by the **revisionHistoryLimit** parameter. The default value is **10**.

1.6.3.11 Scheduling Policy (Affinity/Anti-affinity)

When creating a workload, you can use a nodeSelector to constrain pods to nodes with particular labels. The affinity and anti-affinity features greatly increase the types of constraints you can express.

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

Node Affinity (nodeAffinity)

You can use a nodeSelector to constrain pods to nodes with specific labels. The following example shows how to use a nodeSelector to deploy pods only on the nodes with the **gpu=true** label.

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

Node affinity rules can achieve the same results, as shown in the following example.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: gpu
                operator: In
```

```
            values:
            - "true"
```

Even though the node affinity rule requires more lines, it is more expressive, which will be further described later.

requiredDuringSchedulingIgnoredDuringExecution seems to be complex, but it can be easily understood as a combination of two parts.

- requiredDuringScheduling indicates that pods can be scheduled to the node only when all the defined rules are met (required).
- IgnoredDuringExecution indicates that pods already running on the node do not need to meet the defined rules. That is, a label on the node is ignored, and pods that require the node to contain that label will not be re-scheduled.

In addition, the value of **operator** is **In**, indicating that the label value must be in the values list. Other available operator values are as follows:

- **NotIn**: The label value is not in a list.
- **Exists**: A specific label exists.
- **DoesNotExist**: A specific label does not exist.
- **Gt**: The label value is greater than a specified value (string comparison).
- **Lt**: The label value is less than a specified value (string comparison).

Note that there is no such thing as nodeAntiAffinity because operators **NotIn** and **DoesNotExist** provide the same function.

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

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

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

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

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

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

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

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

Node Preference Rule

The preceding **requiredDuringSchedulingIgnoredDuringExecution** rule is a hard selection rule. There is another type of selection rule, that is,

preferredDuringSchedulingIgnoredDuringExecution. It is used to specify which nodes are preferred during scheduling.

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

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: DISK
                operator: In
                values:
                - SSD
          - weight: 20
            preference:
              matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

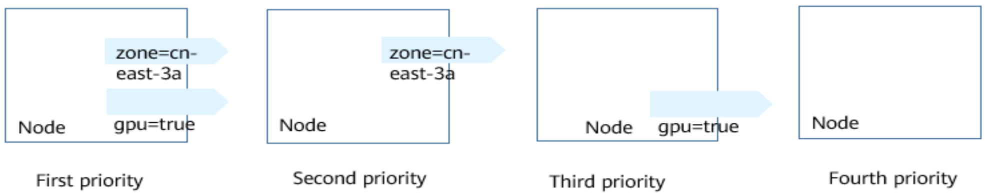
After the deployment, you can find that five pods are deployed on the **192.168.0.212** node, and two pods are deployed on the **192.168.0.100** node.

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

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

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

Figure 1-38 Scheduling priority



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

Workload Affinity (podAffinity)

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

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

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```

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

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

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

The **topologyKey** field specifies the selection range. The scheduler selects nodes within the range based on the affinity rule defined. The effect of **topologyKey** is not fully demonstrated in the preceding example because all the nodes have the **kubernetes.io/hostname** label, that is, all the nodes are within the range.

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

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

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

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

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

Define **topologyKey** in the **podAffinity** section as **prefer**.

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

The scheduler recognizes the nodes with the **prefer** label, that is, **192.168.0.97** and **192.168.0.94**, and then finds the pods with the **app=backend** label. In this way, all frontend pods are deployed onto **192.168.0.97**.

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

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

Workload Anti-Affinity (podAntiAffinity)

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

The following example defines an inter-pod anti-affinity rule, which specifies that pods must not be scheduled to nodes that already have pods with the **app=frontend** label, that is, to deploy the pods of the frontend to different nodes with each node has only one replica.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
```

```
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - frontend
```

Deploy the frontend and query the deployment results. You can find that each node has only one frontend pod and one pod of the Deployment is **Pending**. This is because when the scheduler is deploying the fifth pod, all nodes already have one pod with the **app=frontend** label on them. There is no available node. Therefore, the fifth pod will remain in the **Pending** status.

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

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

Configuring Scheduling Policies

Step 1 When creating a workload, click **Scheduling** in the **Advanced Settings** area.

Table 1-44 Node affinity settings

Parameter	Description
Required	This is a hard rule that must be met for scheduling. It corresponds to **requiredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Multiple required rules can be set, and scheduling will be performed if only one of them is met.
Preferred	This is a soft rule specifying preferences that the scheduler will try to enforce but will not guarantee. It corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in Kubernetes. Scheduling is performed when one rule is met or none of the rules are met.

Step 2 Under **Node Affinity**, **Workload Affinity**, and **Workload Anti-Affinity**, click + to add scheduling policies. In the dialog box displayed, add a policy directly or by specifying a node or an AZ.

Specifying a node or an AZ is essentially implemented through labels. The **kubernetes.io/hostname** label is used when you specify a node, and the **failure-domain.beta.kubernetes.io/zone** label is used when you specify an AZ.

Table 1-45 Scheduling policy configuration

Parameter	Description
Label	Node label. You can use the default label or customize a label.
Operator	The following relations are supported: **In**, **NotIn**, **Exists**, **DoesNotExist**, **Gt**, and **Lt** ● **In**: A label exists in the label list. ● **NotIn**: A label does not exist in the label list. ● **Exists**: A specific label exists. ● **DoesNotExist**: A specific label does not exist. ● **Gt**: The label value is greater than a specified value (string comparison). ● **Lt**: The label value is less than a specified value (string comparison).
Label Value	Label value.
Namespace	This parameter is available only in a workload affinity or anti-affinity scheduling policy. Namespace for which the scheduling policy takes effect.
Topology Key	This parameter can be used only in a workload affinity or anti-affinity scheduling policy. Select the scope specified by **topologyKey** and then select the content defined by the policy.
Weight	This parameter can be set only in a **Preferred** scheduling policy.

----End

1.6.3.12 Tolerance Policies

A tolerance policy allows the scheduler to schedule pods to nodes with corresponding taints. This policy must be used together with node taints. One or more taints can be added to each node. For pods without node tolerance policy, the scheduler performs selective scheduling based on the taint effect to prevent pods from being allocated to inappropriate nodes.

Configuring a Tolerance Policy on the Console

Step 1 Log in to the UCS console.

Step 2 When creating a workload, click **Next: Scheduling and Differentiation**.

Step 3 Add a tolerance policy.

Parameter	Description
Taint Key	Key of a node taint.
Operator	• **Equal**: matches the nodes with the specified taint key (mandatory) and value. If the taint value is left blank, all taints with the key the same as the specified taint key will be matched. • **Exists**: matches the nodes with the specified taint key. In this case, the taint value cannot be specified. If the taint key is left blank, all taints will be tolerated.
Taint Value	• If the value of **Operator** is **Exists**, the value attribute can be omitted. • If the value of **Operator** is **Equal**, the relationship between the key and value is **Equal**. • If **Operator** is not specified, the default value is **Equal**.
Taint Policy	• **All**: All taint policies are matched. • **NoSchedule**: Only the **NoSchedule** taint is matched. • **NoExecute**: Only the **NoExecute** taint is matched.
Toleration Time Window	**tolerationSeconds**, which is configurable only when **Taint Policy** is set to **NoExecute**. Within the tolerance time window, pods still run on the node with taints. After the time expires, the pods will be evicted.

----End

1.6.4 Networking

1.6.4.1 Services

Services provide fixed modes for accessing workloads in a cluster. You can create the following Services on the cluster console:

- **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 from outside the cluster. A NodePort Service is exposed on each node's IP address at a static port. If a node in the cluster is bound to an elastic IP address (EIP), you can use <*EIP*>:<*NodePort*> to access the workload from a public network.

- **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**.

ClusterIP

Step 1 Access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. On the displayed page, click the **Services** tab and select the namespace that the Service belongs to. For details about how to create a namespace, see **Creating a Namespace**.

Step 3 Click **Create Service** in the upper right corner and configure the parameters.

- **Service Name**: Can be the same as the workload name.
- **Service Type**: Select **ClusterIP**.
- **Namespace**: Set it to the namespace that the workload belongs to.
- **Selector**: Add a label and click **Add**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Port**
  - **Protocol**: Select a protocol used by the Service.
  - **Service 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.
  - **Container Port**: Port on which the workload listens. For example, the Nginx application listens on port 80 (container port).

Step 4 Click **OK**.

----End

NodePort

Step 1 Access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. On the displayed page, click the **Services** tab and select the namespace that the Service belongs to. For details about how to create a namespace, see **Creating a Namespace**.

Step 3 Click **Create Service** in the upper right corner and configure the parameters.

- **Service Name**: Can be the same as the workload name.
- **Service Type**: Select **NodePort**.
- **Service Affinity**
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.

  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Namespace**: Set it to the namespace that the workload belongs to.
- **Selector**: Add a label and click **Add**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Port**
  - **Protocol**: Select a protocol used by the Service.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port**: 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

Step 4 Click **OK**.

----End

LoadBalancer

Step 1 Access the cluster console.

Step 2 In the navigation pane, choose **Services & Ingresses**. On the displayed page, click the **Services** tab and select the namespace that the Service belongs to. For details about how to create a namespace, see **Creating a Namespace**.

Step 3 Click **Create Service** in the upper right corner and configure the parameters.

- **Service Name**: Can be the same as the workload name.
- **Service Type**: Select **LoadBalancer**.
- **Service Affinity**
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.

- **Namespace**: Set it to the namespace that the workload belongs to.
- **Selector**: Add a label and click **Add**. A Service selects a pod based on the added label. You can also click **Reference Workload Label** to reference the label of an existing workload. In the dialog box that is displayed, select a workload and click **OK**.
- **Port**
  - **Protocol**: Select a protocol used by the Service.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
- **Annotation**: The key-value pair format is supported. Configure annotations based on your service and vendor requirements and then click **Add**.

Step 4 Click **OK**.

----End

1.6.4.2 Ingresses

An Ingress uses load balancers as the entry for external traffic. Compared with Layer-4 load balancing, it supports Uniform Resource Identifier (URI) configurations and distributes access traffic to the corresponding Services based on the URIs. You can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic. 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**.

Procedure

Step 1 Access the cluster console.

Step 2 In the navigation pane on the left, choose **Services & Ingresses**. On the displayed page, click the **Ingresses** tab and select the namespace that the Ingress belongs to. For details about how to create a namespace, see **Creating a Namespace**.

Step 3 Click **Create Ingress** in the upper right corner and configure the parameters.

- **Name**: name of the Ingress to be created, which can be self-defined.
- **Namespace**: namespace that the Ingress belongs to.
- **TLS**:
  - **Server Certificate**: Select the IngressTLS server certificate. If no desired certificate is available, click **Create IngressTLS Secret** to create an IngressTLS secret. For details, see **Creating a Secret**.
  - **SNI**: Enter the domain name and select the corresponding certificate. Server Name Indication (SNI) is an extended protocol of TLS. It allows multiple TLS-based access domain names to be provided for external systems using the same IP address and port number. Different domain names can use different security certificates.

- **Forwarding Policy**: When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL, for example, 10.117.117.117:80/helloworld), the request is forwarded to the corresponding target Service for processing. You can add multiple forwarding policies.
  - **Domain Name**: (Optional) actual domain name. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
  - **URL**: access path to be registered, for example, **/healthz**. The access path must be the same as the URL exposed by the backend application. Otherwise, a 404 error will be returned.
  - **Destination Service**: Select a Service name. You need to create the NodePort Service first. For details, see **NodePort**.
  - **Destination Service Port**: After you select the destination Service, the corresponding container port is automatically filled in.
- **Ingress Class**: You can select an existing Ingress class or manually enter an Ingress class name.
- **Annotation**: The key-value pair format is supported. Configure annotations based on your service and vendor requirements and then click **Add**.

Step 4 Click **OK**.

----End

1.6.5 Container Storage

To mount a PVC to a cluster, the cluster provider must support the StorageClass resource to dynamically create storage volumes. You can choose **Storage** on the cluster details page and click the **Storage Classes** tab to view the storage classes supported by the cluster. For more information about StorageClass, see **Storage Classes**.

Creating a PVC

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Storage**. On the displayed page, click the **PVCs** tab. Then click **Create from YAML** in the upper right corner.

Step 3 Write a YAML file for the PVC.

Step 4 Click **OK**.

----End

Creating a PV

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Storage**. On the displayed page, click the **PVs** tab. Then click **Create from YAML** in the upper right corner.

Step 3 Write a YAML file for the PV.

Step 4 Click **OK**.

----End

1.6.6 ConfigMaps and Secrets

1.6.6.1 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 Access the cluster details page. In the navigation pane, choose **ConfigMaps and Secrets**. Then click the **ConfigMaps** tab. You can create a ConfigMap directly or using YAML. If you want to create a ConfigMap using YAML, go to **Step 4**.

Step 2 Select the namespace to which the ConfigMap will belong.

Step 3 Create a ConfigMap directly by clicking **Create ConfigMap**.

Set the parameters listed in **Table 1-46**.

Table 1-46 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. The current namespace is used by default.
Description	Description of the ConfigMap.
Data	The workload configuration data can be used in a container or used to store the configuration data. Click + and enter the key and value. **Key** indicates the configuration name, and **Value** indicates the configuration content.

Parameter	Description
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. Enter the label key and value. 2. Click **Add**.

Step 4 Create a ConfigMap from a YAML file by clicking **Create from YAML**.

NOTE

To create a resource by uploading a file, ensure that the resource description file has been created. UCS 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 **Import** 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 5 When the configuration is complete, click **OK**.

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
  namespace: default
data:
  data-1: "value-1"
  data-2: "value-2"
```

Related Operations

On the cluster details page, you can also perform the operations described in **Table 1-47**.

Table 1-47 Related operations

Operation	Description
Viewing details	Click the ConfigMap name to view its details.
Editing a YAML file	Click **Edit YAML** in the row where the target ConfigMap resides to edit its YAML file.
Updating a ConfigMap	1. Click **Update** in the row where the target ConfigMap resides. 2. Modify the ConfigMap data according to **Table 1-46**. 3. Click **OK** to submit the modified information.
Deleting a ConfigMap	Click **Delete** in the row where the target ConfigMap resides, and click **Yes**.
Deleting ConfigMaps in batches	1. Select the ConfigMap to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

1.6.6.2 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 navigation pane, choose **ConfigMaps and Secrets**, and click the **Secrets** tab. You can create a secret directly or using YAML. If you want to create a secret using YAML, go to **Step 4**.

Step 2 Select the namespace to which the secret will belong.

Step 3 Click **Create Secret**.

Set the parameters listed in **Table 1-48**.

Table 1-48 Basic information parameters

Parameter	Description
Name	Name of the secret you create, which must be unique.
Namespace	Namespace to which the secret belongs. The current namespace is used by default.
Description	Description of the secret.
Secret Type	Type of the secret. ● **Opaque**: general secret type. 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. If you select this secret type, enter the image repository address. ● **IngressTLS**: a secret that stores the certificate required by ingresses. If you select this secret type, upload the certificate file and private key file. ● **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 select **Auto Base64 Encoding** to Base64-encode the entered value. For details about manual Base64 encoding, see **Base64 Encoding**. ● If the secret type is **kubernetes.io/dockerconfigjson**, enter the username 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. Set **Key** and **Value**. 2. Click **Confirm**.

Step 4 Create a secret from a YAML file by clicking **Create from YAML**.

NOTE

To create a resource by uploading a file, ensure that the resource description file has been created. UCS 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 **Import** 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 5 When the configuration is complete, click **OK**.

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 1-49**.

NOTE

The secrets in the **kube-system** namespace can only be viewed.

Table 1-49 Other operations

Operation	Description
Editing a YAML file	Click **Edit YAML** in the row where the target secret resides to edit its YAML file.
Updating a secret	1. Click **Update** in the row where the target secret resides. 2. Modify the secret data according to **Table 1-48**. 3. Click **OK**.
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** in the upper left corner. 3. Delete the secret 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
******
```

1.6.7 kubeconfig

1.6.7.1 Obtaining a kubeconfig File

A kubeconfig file contains the authentication credentials and endpoint (access address) required for accessing a Kubernetes cluster when used in conjunction with kubectl or other clients. For details, see the **Kubernetes documentation**.

This section describes how to obtain the kubeconfig file of a cluster. Different cluster providers have different kubeconfig file formats. Perform operations based on your cluster.

NOTICE

The kubeconfig file contains cluster authentication information. If this file is leaked, your clusters may be attacked. Keep it secure.

Huawei Cloud Clusters

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

Step 2 In the **Connection Information** area, click **Configure** next to kubectl.

Step 3 Download the kubectl configuration file as prompted. (If the public IP address is changed, you need to download it again.)

Step 4 Use the configuration file downloaded in **Step 3** to connect to the cluster. For details, see **Registering an Attached Cluster (Public Network Access)** or **Registering an Attached Cluster (Private Network Access)**.

----End

Third-Party Cloud Clusters

Different third-party cloud vendors have different kubeconfig file formats. You need to create a ServiceAccount that has the permission of all cluster resources and obtain the token of the ServiceAccount to configure the kubeconfig file supported by UCS.

Step 1 Use kubectl to connect to the cluster.

Step 2 Create the **ucs-service-account.yaml** file.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ucs-user
---
apiVersion: v1
kind: Secret
metadata:
  name: ucs-user-token
  annotations:
    kubernetes.io/service-account.name: "ucs-user"
type: kubernetes.io/service-account-token
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ucs-user-role
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - get
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: ucs-user-role-binding
subjects:
  - kind: ServiceAccount
    name: ucs-user
    namespace: default
roleRef:
  kind: ClusterRole
  name: ucs-user-role
  apiGroup: rbac.authorization.k8s.io
```

Step 3 Run the following command in the cluster to create a ServiceAccount:

kubectl apply -f ucs-service-account.yaml

Step 4 Run the following command to obtain the token:

kubectl get secret ucs-user-token -n *default* **-oyaml | grep token: | awk '{print $2}' | base64 -d ;echo**

Step 5 Configure the kubeconfig file.

Create a **kubeconfig.yaml** file by referring to the following example and replace the token with the value obtained in **Step 4**.

kubeconfig.yaml:

```
kind: Config
apiVersion: v1
preferences: {}
clusters:
  - name: internalCluster
    cluster:
      server: 'https://kubernetes.default.svc.cluster.local:443'
      insecure-skip-tls-verify: true
users:
  - name: ucs-user
    user:
      token: 'MIIFbAYJKo*****'
contexts:
  - name: internal
    context:
      cluster: internalCluster
      user: ucs-user
current-context: internal
```

The parameters in the kubeconfig file are described as follows:

Parameter	Value	Description	Mandatory
server	'https://kubernetes.default.svc.cluster.local:443'	Intra-cluster access address of the API server. Some vendors restrict cluster external access to the API server, so UCS may fail to connect to the cluster. You are advised to use the intra-cluster access address.	Yes
insecure-skip-tls-verify	true	If this parameter is used, certificate authentication is skipped. The value must be **true**.	1 out of 2 **NOTE** If the value of **server** is an intra-cluster access address, certificate authentication is preferentially skipped.

Parameter	Value	Description	Mandatory
certificate-authority-data	Base64-encrypted string	If this parameter is used, two-way authentication is enabled for the cluster. The value is the server certificate encrypted using Base64. The default path of the server certificate of a native Kubernetes cluster is **/etc/kubernetes/pki/ca.crt** on the master node.	
token	Base64-encrypted string	Token-based authentication. The value is the token obtained in **Step 4**.	1 out of 3 **NOTE** Token-based authentication is recommended. UCS supports only the three authentication modes.
• client-certificate-data • client-key-data	Base64-encrypted string	Certificate- and private key–based authentication. • **client-certificate-data**: client certificate encrypted using Base64. • **client-key-data**: client private key encrypted using Base64.	
• username • password	String	Username- and password-based authentication. • **username**: username for accessing the cluster. • **password**: password of the username.	

Step 6 Use the kubeconfig file configured in **Step 5** to connect to the cluster. For details, see **Registering an Attached Cluster (Public Network Access)** or **Registering an Attached Cluster (Private Network Access)**.

NOTE

When using UCS, you cannot delete the ServiceAccount, ClusterRole, and ClusterRoleBinding. Otherwise, the token will be invalid.

If the cluster is no longer connected to UCS, you can run the **kubectl delete -f ucs-service-account.yaml** command to delete the ServiceAccount.

----End

Self-Managed Clusters

If your cluster is a standard cluster built using an official Kubernetes binary file or a deployment tool such as Kubeadm, you can perform the following steps to obtain the kubeconfig file.

The procedure does not apply to commercial clusters provided by cloud service vendors. For details about how to obtain the kubeconfig file of a commercial cluster, see **Third-Party Cloud Clusters**.

Step 1 Log in to the master node of the cluster.

Step 2 View the cluster access credential. By default, the kubeconfig file of a self-managed cluster is stored in **$HOME/.kube/config** on the master node. If another kubeconfig file is specified for your cluster, change the directory.

cat $HOME/.kube/config

Step 3 Copy the credential content.

Step 4 Create a YAML file on your local PC, paste the credential content to the file, and save the file.

Step 5 Use the YAML file created in **Step 4** to connect to the cluster. For details, see **Registering an Attached Cluster (Public Network Access)** or **Registering an Attached Cluster (Private Network Access)**.

----End

1.6.7.2 Updating a kubeconfig File

This section describes how to update the kubeconfig file of a cluster to handle leakage or expiration of cluster certificates or perform routine security maintenance.

Only the kubeconfig files of attached clusters and partner cloud clusters can be updated.

Prerequisites

- The target cluster has not joined any fleet.
- The anp-agent add-on has been installed in the cluster to ensure that the new kubeconfig file can be used for connectivity detection with the cluster.

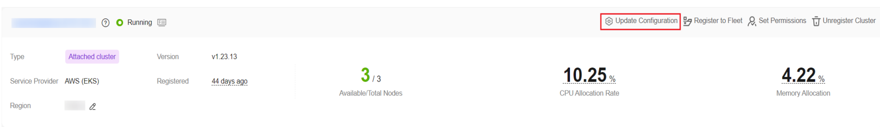
Procedure

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 Click **Clusters Not in Fleet** and locate the cluster whose kubeconfig file needs to be updated.

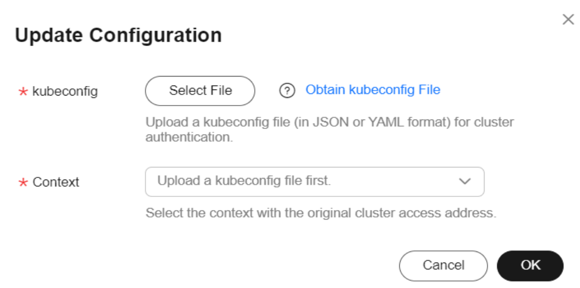
Step 3 Click  in the upper right corner.

Figure 1-39 Updating the kubeconfig file



Step 4 Upload the local kubeconfig file and select the context address that is the same as the original address.

Figure 1-40 Uploading the kubeconfig file



Step 5 Click **OK**.

----End

1.6.8 Custom Resource Definitions

Custom Resource Definitions (CRDs) are custom resource objects similar to Deployments or Services. You can run the kubectl commands to create and access CRDs for modular Kubernetes extension. For details, see **Extend the Kubernetes API with CustomResourceDefinitions**.

Procedure

Step 1 Access the cluster details page.

Step 2 In the navigation pane on the left, choose **Custom Resources**, and click **Create from YAML** in the upper right corner.

Step 3 Edit the YAML file online or import one, and click **OK**.

Step 4 Other operations:

- Click **View YAML** in the **Operation** column of the target CRD to view its YAML file.
- Click **View Details** in the **Operation** column of the target CRD to view its instances in the cluster.

----End

1.6.9 Namespaces

Namespaces that you create on the cluster console apply only to the current cluster. You can create Kubernetes objects and manage resource quotas in such namespaces, or delete these namespaces.

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

Creating a Namespace

Step 1 Access the cluster details page.

Step 2 Choose **Namespaces** in the navigation pane, click **Create Namespace** in the upper right corner, and configure parameters.

- **Namespace Name**: Name of the namespace, which must be unique in a cluster.
- **Description**: Description of the namespace.
- **Quota Management**: If this function is enabled, you can configure resource quotas. Resource quotas can limit the amount of resources available in namespaces, achieving resource allocation by namespace.

  If you do not enable this function, you can click **Manage Quota** in the namespace list to configure resource quotas after the namespace is created. For details, see **Configuring Resource Quotas in a Namespace**.

Step 3 Click **OK**.

----End

Deleting a Namespace

NOTICE

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

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Namespaces**, select the target namespace, and choose **More** > **Delete**.

----End

Configuring 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.

NOTICE

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

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Namespaces**, locate the target namespace, and click **Manage Quota** in the **Operation** column.

Step 3 Configure resource quotas.

NOTICE

- There is no limit on quotas by default. To specify a resource quota, enter an integer greater than or equal to 1. If you want to limit the CPU or memory quota, you must specify the CPU or memory request 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.
- **Memory (MiB)**: maximum amount of memory that can be allocated to workload pods in the namespace.
- **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.
- **Pods**: maximum number of pods, including those in terminated state, that can be created in the namespace.
- **Pods** (excluding terminated pods): maximum number of pods in a non-terminated state that can be created in the namespace.
- **Services**: maximum number of Services, including those in terminated state, that can be created in the namespace.
- **Services** (excluding terminated Services): maximum number of Services in a non-terminated state that can be created in the namespace.
- **PersistentVolumeClaims (PVCs)**: maximum number of PVCs that can be created in the namespace.
- **ConfigMaps**: maximum number of ConfigMaps that can be created in the namespace.
- **Secrets**: maximum number of secrets that can be created in the namespace.

Step 4 Click **OK**.

----End

1.6.10 Workload Auto Scaling (HPA)

Horizontal Pod Autoscaling (HPA) in Kubernetes implements horizontal scaling of pods. In a CCE HPA policy, you can configure different cooldown period windows and scaling thresholds for different applications.

Prerequisites

To use HPA, install either of the following add-ons that support metrics APIs: (For details, see **Support for metrics APIs**.)

- metrics-server: collects metrics from the Summary API exposed by kubelet and provides resource usage metrics such as the container CPU and memory usage
  - For details about how to install metrics-server for an on-premises cluster, see **metrics-server**.
  - For details about how to install metrics-server for other types of clusters, see the **official documentation**. For an attached cluster, you can also install metric-server provided by the corresponding vendor.
- Prometheus: an open source monitoring and alarming framework that collects metrics and provides basic resource metrics and custom metrics

Constraints

- At least one pod is available in the cluster. If no pod is available, pod scale-out will be performed.
- If no metric collection add-on has been installed in the cluster, the workload scaling policy cannot take effect.
- metrics-server can only be installed for on-premises clusters for calling the Metrics API. More add-ons will be available in the future.

Procedure

Step 1 Access the cluster details page.

- If the cluster is not added to any fleet, click the cluster name.
- If the cluster has been added to a fleet, click the fleet name. In the navigation pane, choose **Clusters** > **Container Clusters**.

Step 2 In the navigation pane, choose **Workload Scaling**. Then click **Create HPA Policy** in the upper right corner.

Step 3 Configure the parameters for the HPA policy.

Table 1-50 HPA policy parameters

Parameter	Description
Policy Name	Enter a name for the policy.
Namespace	Select the namespace that the workload belongs to.
Associated Workload	Select the workload that the HPA policy is associated with.
Pod Range	Enter minimum and maximum numbers of pods. When the policy is triggered, the workload pods are scaled within this range.

<table>
<tr><td>System Policy</td><td>

- **Metric**: Select **CPU usage** or **Memory usage**.

NOTE
Usage = CPU or memory used by pods/Requested CPU or memory

- **Desired Value**: Enter the desired average resource usage.

This parameter indicates the desired value of the selected metric.

Number of new pods required (rounded up) = Current metric value/Desired value x Number of current pods

NOTE
When calculating the number of pods to be added or reduced, the HPA policy uses the maximum number of pods in the last 5 minutes.

- **Tolerance Range**: The default tolerance is 0.1. Enter the scale-in and scale-out thresholds. The desired metric value must be within this tolerance range.
If the metric value is greater than the scale-in threshold and less than the scale-out threshold, no scaling operation will be triggered. **This parameter is available only in clusters of v1.15 or later.**

NOTICE
You can configure multiple system policies.
</td></tr>
</table>

----End

1.6.11 Add-ons

1.6.11.1 kube-prometheus-stack

Introduction

kube-prometheus-stack provides easy-to-use, end-to-end Kubernetes cluster monitoring capabilities by using Prometheus Operators and Prometheus. It also supports customized add-on specifications, interconnection with Grafana, high availability, and node affinity.

The core components of kube-prometheus-stack include prometheusOperator, prometheus, alertmanager, thanosSidecar, thanosQuery, adapter, kubeStateMetrics, nodeExporter, and grafana.

- prometheusOperator: deploys and manages the Prometheus Server based on Custom Resource Definition (CRDs), and monitors and processes the events related to these CRDs. It is the control center of the entire system.
- prometheus (Server): a Prometheus Server cluster deployed by the operator based on the Prometheus CRDs that can be regarded as StatefulSets.
- alertmanager: the alarm center of the add-on. It receives alarms sent by Prometheus and manages alarm information by deduplicating, grouping, and distributing.

- thanosSidecar: in HA scenarios, runs with Prometheus in the same pod to implement persistent storage of Prometheus metric data.
- thanosQuery: entry for PromQL query when Prometheus is in HA scenarios. It can delete duplicate data of the same metrics from Store or Prometheus.
- adapter (custom-metrics-apiserver): aggregates custom metrics to the native Kubernetes API Server.
- kube-state-metrics: converts the Prometheus metric data into a format that can be identified by Kubernetes APIs. By default, kube-state-metrics does not collect all labels and annotations of Kubernetes resources. If these labels and annotations need to be collected, see **How Do I Modify the Collection Configuration of the kube-state-metrics Component?**.
- nodeExporter: deployed on each node to collect node monitoring data.
- grafana: visualizes monitoring data. grafana creates a 5 GiB storage volume by default. Uninstalling the add-on will not delete this volume.
- clusterProblemDetector: monitors cluster exceptions.

Add-on Deployment Modes

The kube-prometheus-stack add-on can be deployed in **Agent** or **Server** mode.

- Deployed in **Agent** mode, the add-on occupies fewer cluster resources and provides the Prometheus metric collection capability for the cluster. However, the HPA and health diagnosis functions based on custom Prometheus statements are not supported.
- Deployed in **Server** mode, the add-on supports HPA and health diagnosis based on custom Prometheus statements. This mode depends on PVC and consumes a large amount of memory.

Precaution

kube-prometheus-stack is a system monitoring add-on. When cluster resources are insufficient, Kubernetes prioritizes resource scheduling to the pod where the add-on runs.

Permissions

nodeExporter monitors the disk space of Docker and reads the info data of Docker from the **/var/run/dockersock** directory of the host.

The following privilege is required by nodeExporter:

- cap_dac_override: reads the info data of Docker.

Upgrading the Add-on

Step 1 Select a fleet or a cluster that is not added to the fleet.

Figure 1-41 Selecting a fleet or a cluster not in the fleet

Step 2 Choose **Container Insights** > **Clusters** to view the clusters with monitoring enabled. Locate the cluster for which the add-on is to be upgraded and click **View Details** in the **Operation** column to access its overview page.

Step 3 The version of kube-prometheus-stack is displayed in the upper right corner. If the version is not the latest, upgrade the add-on to experience the latest functions.

----End

Resource Quota Requirements of Different Specifications

Before installing the kube-prometheus-stack add-on, ensure that the cluster has sufficient schedulable resources such as CPUs and memory. For details about the resource quota requirements of default specifications in Agent mode, see **Table 1-51**. For details about the resource quota requirements of different add-on specifications in Server mode, see **Table 1-52**.

Table 1-51 Resource quota requirements of default specifications in Agent mode

Add-on Specification	Container	CPU Quota		Memory Quota	
Default	prometheusOperator	Request: 100m	Limit: 500m	Request: 100 MiB	Limit: 500 MiB
	prometheus	Request: 500m	Limit: 4	Request: 1 GiB	Limit: 8 GiB
	kube-state-metrics	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 500 MiB
	nodeExporter	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 1 GiB
	grafana	Request: 100m	Limit: 500m	Request: 200 MiB	Limit: 2 GiB

Table 1-52 Resource quota requirements of different specifications in Server mode

Add-on Specification	Container	CPU Quota		Memory Quota	
Demo (≤ 100 containers)	prometheusOperator	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 500 MiB
	prometheus	Request: 500m	Limit: 2	Request: 2 GiB	Limit: 8 GiB
	alertmanager	Request: 200m	Limit: 1	Request: 200 MiB	Limit: 1 GiB
	thanosSidecar	Request: 100m	Limit: 1	Request: 100 MiB	Limit: 2 GiB
	thanosQuery	Request: 500m	Limit: 2	Request: 500 MiB	Limit: 4 GiB
	adapter	Request: 400m	Limit: 2	Request: 400 MiB	Limit: 1 GiB
	kube-state-metrics	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 500 MiB
	nodeExporter	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 1 GiB
	grafana	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 2 GiB
	clusterProblemDetector	Request: 100m	Limit: 200m	Request: 200 MiB	Limit: 400 MiB
Small (≤ 2,000 containers)	prometheusOperator	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 500 MiB
	prometheus	Request: 4	Limit: 8	Request: 16 GiB	Limit: 32 GiB
	alertmanager	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 1 GiB
	thanosSidecar	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 2 GiB
	thanosQuery	Request: 2	Limit: 4	Request: 2 GiB	Limit: 16 GiB
	adapter	Request: 2	Limit: 4	Request: 4 GiB	Limit: 16 GiB
	kube-state-metrics	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 1 GiB

Add-on Specification	Container	CPU Quota		Memory Quota	
	nodeExporter	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 1 GiB
	grafana	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 2 GiB
	clusterProblemDetector	Request: 200m	Limit: 500m	Request: 300 MiB	Limit: 1 GiB
Medium (≤ 5,000 containers)	prometheusOperator	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 1 GiB
	prometheus	Request: 8	Limit: 16	Request: 32 GiB	Limit: 64 GiB
	alertmanager	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 2 GiB
	thanosSidecar	Request: 1	Limit: 2	Request: 1 GiB	Limit: 4 GiB
	thanosQuery	Request: 2	Limit: 4	Request: 2 GiB	Limit: 16 GiB
	adapter	Request: 2	Limit: 4	Request: 16 GiB	Limit: 32 GiB
	kube-state-metrics	Request: 1	Limit: 2	Request: 1 GiB	Limit: 2 GiB
	nodeExporter	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 1 GiB
	grafana	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 2 GiB
	clusterProblemDetector	Request: 200m	Limit: 1	Request: 400 MiB	Limit: 2 GiB
Large (> 5,000 containers)	prometheusOperator	Request: 500m	Limit: 1	Request: 500 MiB	Limit: 2 GiB
	prometheus	Request: 8	Limit: 32	Request: 64 GiB	Limit: 128 GiB
	alertmanager	Request: 1	Limit: 2	Request: 1 GiB	Limit: 4 GiB
	thanosSidecar	Request: 2	Limit: 4	Request: 2 GiB	Limit: 8 GiB
	thanosQuery	Request: 2	Limit: 4	Request: 2 GiB	Limit: 32 GiB

Add-on Specification	Container	CPU Quota		Memory Quota	
	adapter	Request: 2	Limit: 4	Request: 32 GiB	Limit: 64 GiB
	kube-state-metrics	Request: 1	Limit: 3	Request: 1 GiB	Limit: 3 GiB
	nodeExporter	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 1 GiB
	grafana	Request: 200m	Limit: 500m	Request: 200 MiB	Limit: 2 GiB
	clusterProblemDetector	Request: 200m	Limit: 1	Request: 400 MiB	Limit: 2 GiB

1.6.11.2 log-agent

log-agent is a cloud native log collection add-on built based on open source fluent-bit and OpenTelemetry. It supports CRD-based log collection policies, collects and forwards standard output logs, container file logs, node logs, and Kubernetes events of containers in a cluster.

The core components of log-agent include fluent-bit, cop-logs, log-operator, and otel-collector.

- fluent-bit: indicates the log collector, which is installed on each node as a DaemonSet.
- cop-logs: generates and updates configuration files on the collection side.
- log-operator: parses and updates log rules.
- otel-collector: forwards logs collected by fluent-bit to LTS in a centralized manner.

Resource Quota Requirements of Different Specifications

Ensure that the cluster has sufficient CPU and memory resources for scheduling when installing log-agent. **Table 1-53** describes the resource quota requirements of different log-agent specifications.

Table 1-53 Resource quota requirements of different specifications

Add-on Specification	Container	CPU Quota		Memory Quota	
Small specifications (1 pod)	fluent-bit	Request: 100 m	Limit: 500 m	Request: 200 MiB	Limit: 500 MiB
	cop-logs	Request: 100 m	Limit: 1	Request: 100 MiB	Limit: 500 MiB
	log-operator	Request: 100 m	Limit: 500 m	Request: 100 MiB	Limit: 500 MiB
	otel-collector	Request: 200 m	Limit: 1	Request: 1 GiB	Limit: 2 GiB
Large specifications (2 pods)	fluent-bit	Request: 100 m	Limit: 500 m	Request: 200 MiB	Limit: 500 MiB
	cop-logs	Request: 100 m	Limit: 1	Request: 100 MiB	Limit: 500 MiB
	log-operator	Request: 100 m	Limit: 500 m	Request: 100 MiB	Limit: 500 MiB
	otel-collector	Request: 200 m	Limit: 1	Request: 1 GiB	Limit: 2 GiB

1.6.11.3 metrics-server

From version 1.8 onwards, Kubernetes provides resource usage metrics, such as the container CPU and memory usage, through the Metrics API. These metrics can be directly accessed by users (for example, by running **kubectl top**) or used by controllers (for example, Horizontal Pod Autoscaler) in a cluster for decision-making. Metrics Server fetches these metrics.

Metrics Server is a cluster-wide aggregator of resource usage data. It is used as the metrics-server add-on on UCS. You can quickly install this add-on on the cluster details page.

After metrics-server is installed, you can create an HPA policy on the **Workload Scaling** page. For details, see **Workload Auto Scaling (HPA)**.

Official community projects: **https://github.com/kubernetes-sigs/metrics-server**

Constraints

metrics-server can be installed only in on-premises clusters.

Installing the Add-on

Step 1 Access the cluster details page.

- If the cluster is not added to any fleet, click the cluster name.
- If the cluster has been added to a fleet, click the fleet name. In the navigation pane, choose **Clusters** > **Container Clusters**.

Step 2 In the navigation pane, choose **Add-ons**. Locate **metrics-server** in **Add-ons Available** and click **Install**.

Step 3 Set **Add-on Specifications** to **Standalone**, **HA**, or **Custom** and click **Install**.

NOTE

- In the on-premises cluster, the maximum number of metrics-server instances depends on the number of manage nodes. If you want to create more metrics-server instances using custom specifications, expand the number of manage nodes first.
- The manage nodes are managed using labels and taints in the on-premises cluster. To expand the number of the manage nodes, you only need to add labels and taints to non-manage nodes in the cluster. The procedure is as follows:
  1. Access the cluster details page and click **Nodes** in the navigation pane.
  2. Select the non-manage node and click **Labels and Taints**.
  3. Click **Add Operation** to add an update content: **Add/Update** > **Kubernetes** > **cop.manage** > **manage**.
  4. Click **Add Operation** to add an update content: **Add/Update** > **Taint** > **role** > **manage** > **NoSchedule**.
  5. Click **OK**.

Step 4 Click the name of metrics-server in the installed add-on list to view the deployment status of the add-on instance in the cluster.

----End

Upgrading the Add-on

Step 1 Access the cluster details page. In the navigation pane, choose **Add-ons**.

Step 2 In the installed add-on list, if there is "New version available" next to the version label of metrics-server, click **Upgrade**.

NOTE

- If the button is unavailable, the add-on is already up-to-date and no upgrade is required.
- During the upgrade, metrics-server of the old version will be discarded, and metrics-server of the latest version will be installed.

Step 3 Configure the parameters as prompted and click **OK**.

----End

Modifying the Add-on

Step 1 Access the cluster details page. In the navigation pane, choose **Add-ons**.

Step 2 Locate metrics-server in the installed add-ons and click **Edit**.

Step 3 Configure the parameters as prompted and click **OK**.

----End

Uninstalling the Add-on

Step 1 Access the cluster details page. In the navigation pane, choose **Add-ons**.

Step 2 Locate metrics-server in the installed add-ons and click **Uninstall**.

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

NOTE

After metrics-server is uninstalled, you need to install another add-on that provides the Metrics API. If no add-on is installed, existing workload scaling policies will become unavailable.

----End

1.6.11.4 volcano

Introduction

Volcano is a batch processing platform based on Kubernetes. It provides a series of features required by machine learning, deep learning, bioinformatics, genomics, and other big data applications, as a powerful supplement to Kubernetes capabilities.

Volcano provides general-purpose, high-performance computing capabilities, such as job scheduling engine, heterogeneous chip management, and job running management, serving end users through computing frameworks for different industries, such as AI, big data, gene sequencing, and rendering. (Volcano has been open-sourced in GitHub.)

Volcano provides job scheduling, job management, and queue management for computing applications. Its main features are as follows:

- Diverse computing frameworks, such as TensorFlow, MPI, and Spark, can run on Kubernetes in containers. Common APIs for batch computing jobs through CRD, various add-ons, and advanced job lifecycle management are provided.
- Advanced scheduling capabilities are provided for batch computing and high-performance computing scenarios, including group scheduling, preemptive priority scheduling, packing, resource reservation, and task topology.
- Queues can be effectively managed for scheduling jobs. Complex job scheduling capabilities such as queue priority and multi-level queues are supported.

Open source community: **https://github.com/volcano-sh/volcano**

Installing the Add-on

Step 1 Log in to the UCS console and click the cluster name to go to its details page. In the navigation pane, choose **Add-ons**. Locate **Volcano** and click **Install**.

Step 2 Select **Standalone**, **Custom**, or **HA** for **Add-on Specifications**.

If you select **Custom**, the following requests and limits are recommended for **volcano-controller** and **volcano-scheduler**:

- If the number of nodes is less than 100, retain the default configuration. The requested CPU is 500m, and the limit is 2000m. The requested memory is 500 Mi, and the limit is 2000 Mi.
- If the number of nodes is greater than 100, increase the requested CPU by 500m and the requested memory by 1000 Mi each time 100 nodes (10,000 pods) are added. Increase the CPU limit by 1500m and the memory limit by 1000 Mi.

NOTE

Formulas for calculating the requests and limits:

- CPU: Calculate the number of nodes multiplied by the number of pods, perform interpolation search using the product of the number of nodes in the cluster multiplied by the number of pods in **Table 1-54**, and round up the request and limit that are closest to the specifications.

  For example, for 2,000 nodes and 20,000 pods, Number of target nodes x Number of target pods = 40 million, which is close to 700/70000 in the specification (Number of nodes x Number of pods = 49 million). You are advised to set the CPU request to 4000m and the limit to 5500m.
- Memory: Allocate 2.4 GiB of memory to every 1,000 nodes and 1 GiB of memory to every 10,000 pods. The memory request is the sum of the two values. (The obtained value may be different from the recommended value in **Table 1-54**. You can use either of them.)

  Memory request = Number of nodes/1000 x 2.4 GiB + Number of pods/10000 x 1 GiB

  For example, for 2,000 nodes and 20,000 pods, the memory request value is 6.8 GiB (2000/1000 x 2.4 GiB + 20000/10000 x 1 GiB).

Table 1-54 Recommended requests and limits for volcano-controller and volcano-scheduler

Nodes/Pods in a Cluster	CPU Request (m)	CPU Limit (m)	Memory Request (Mi)	Memory Limit (Mi)
50/5,000	500	2,000	500	2,000
100/10,000	1,000	2,500	1,500	2,500
200/20,000	1,500	3,000	2,500	3,500
300/30,000	2,000	3,500	3,500	4,500
400/40,000	2,500	4,000	4,500	5,500
500/50,000	3,000	4,500	5,500	6,500
600/60,000	3,500	5,000	6,500	7,500
700/70,000	4,000	5,500	7,500	8,500

Step 3 Configure parameters of the default volcano scheduler. For details, see **Table 1-55**.

```
colocation_enable: ''
default_scheduler_conf:
  actions: 'allocate, backfill'
  tiers:
    - plugins:
```



```
    - name: 'priority'
    - name: 'gang'
    - name: 'conformance'
  - plugins:
    - name: 'drf'
    - name: 'predicates'
    - name: 'nodeorder'
  - plugins:
    - name: 'cce-gpu-topology-predicate'
    - name: 'cce-gpu-topology-priority'
    - name: 'cce-gpu'
  - plugins:
    - name: 'nodelocalvolume'
    - name: 'nodeemptydirvolume'
    - name: 'nodeCSIscheduling'
    - name: 'networkresource'
```

Table 1-55 Volcano add-ons

Add-on	Function	Description	Demonstration

binpack	Schedules pods to nodes with high resource utilization to reduce resource fragments.	• **binpack.weight**: weight of the binpack add-on.
		• **binpack.cpu**: percentage of CPU. The default value is **1**.
		• **binpack.memory**: percentage of memory. The default value is **1**.
		• **binpack.resources**: resource type.
			`- plugins:`
			`  - name: binpack`
			`    arguments:`
			`      binpack.weight: 10`
			`      binpack.cpu: 1`
			`      binpack.memory: 1`
			`      binpack.resources: nvidia.com/gpu, example.com/foo`
			`      binpack.resources.nvidia.com/gpu: 2`
			`      binpack.resources.example.com/foo: 3`

conformance	Prevent key pods, such as the pods in the **kube-system** namespace from being preempted.	-	-
gang	The gang add-on considers a group of pods as a whole to allocate resources.	-	-
priority	The priority add-on schedules pods based on the custom workload priority.	-	-

Add-on	Function	Description	Demonstration

overcommit	Resources in a cluster are scheduled after being accumulated in a certain multiple to improve the workload enqueuing efficiency. If all workloads are Deployments, remove this add-on or set the raising factor to **2.0**.	**overcommit-factor**: Raising factor. The default value is **1.2**.	`- plugins:`
			`  - name: overcommit`
			`    arguments:`
			`      overcommit-factor: 2.0`

drf	Schedules resources based on the container group dominant resources. The smallest dominant resources would be selected for priority scheduling.	-	-
predicates	Determines whether a task is bound to a node using a series of evaluation algorithms, such as node/pod affinity, taint tolerance, node port repetition, volume limits, and volume zone matching.	-	-

Add-on	Function	Description	Demonstration

nodeorder	The nodeorder add-on scores all nodes for a task by using a series of scoring algorithms.	• **nodeaffinity.weight**: Pods are scheduled based on the node affinity. The default value is **1**.
		• **podaffinity.weight**: Pods are scheduled based on the pod affinity. The default value is **1**.
		• **leastrequested.weight**: Pods are scheduled to the node with the least requested resources. The default value is **1**.
		• **balancedresource.weight**: Pods are scheduled to the node with balanced resource. The default value is **1**.
		• **mostrequested.weight**: Pods are scheduled to the node with the most requested resources. The default value is **0**.
		• **tainttoleration.weight**: Pods are scheduled to the node with a high taint tolerance. The default value is **1**.
		• **imagelocality.weight**: Pods are scheduled to the node where the required images exist. The default value is **1**.
		• **selectorspread.weight**: Pods are evenly scheduled to different nodes. The default value is **0**.
		• **volumebinding.weight**: Pods are scheduled to the node with the local PV
			- plugins:
			  - name: nodeorder
			    arguments:
			      leastrequested.weight: 1
			      mostrequested.weight: 0
			      nodeaffinity.weight: 1
			      podaffinity.weight: 1
			      balancedresource.weight: 1
			      tainttoleration.weight: 1
			      imagelocality.weight: 1
			      volumebinding.weight: 1
			      podtopologyspread.weight: 2

Add-on	Function	Description	Demonstration
		delayed binding policy. The default value is **1**. ● **podtopologyspread.weight**: Pods are scheduled based on the pod topology. The default value is **2**.	
cce-gpu-topology-predicate	GPU-topology scheduling preselection algorithm	-	-
cce-gpu-topology-priority	GPU-topology scheduling priority algorithm	-	-
cce-gpu	GPU resource allocation that supports decimal GPU configurations by working with the gpu add-on.	-	-
numaaware	NUMA topology scheduling	**weight**: Weight of the numa-aware add-on.	-
networkresource	The ENI requirement node can be preselected and filtered. The parameters are transferred by CCE and do not need to be manually configured.	**NetworkType**: network type (**eni** or **vpc-router**).	-
nodelocalvolume	Filters out nodes that do not meet local volume requirements.	-	-

Add-on	Function	Description	Demonstration
nodeemptydirvolume	Filters out nodes that do not meet the emptyDir requirements.	-	-
nodeCSIscheduling	Filters out nodes that have everest component exceptions.	-	-

Step 4 Click **Install**.

----End

Modifying the volcano-scheduler Configurations Using the Console

Volcano allows you to configure the scheduler during installation, upgrade, and editing. The configuration will be synchronized to volcano-scheduler-configmap.

This section describes how to configure volcano-scheduler.

NOTE

Only Volcano of v1.7.1 and later support this function. On the new add-on page, options such as **plugins.eas_service** and **resource_exporter_enable** are replaced by **default_scheduler_conf**.

Log in to the CCE console and access the cluster console. Choose **Add-ons** in the navigation pane. On the right of the page, locate **volcano** and click **Install** or **Upgrade**. In the **Parameters** area, configure the volcano-scheduler parameters.

- Using **resource_exporter**:

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
```

```
                "name": "predicates"
            },
            {
                "name": "nodeorder"
            }
        ]
    },
    {
        "plugins": [
            {
                "name": "cce-gpu-topology-predicate"
            },
            {
                "name": "cce-gpu-topology-priority"
            },
            {
                "name": "cce-gpu"
            },
            {
                "name": "numa-aware" # add this also enable resource_exporter
            }
        ]
    },
    {
        "plugins": [
            {
                "name": "nodelocalvolume"
            },
            {
                "name": "nodeemptydirvolume"
            },
            {
                "name": "nodeCSIscheduling"
            },
            {
                "name": "networkresource"
            }
        ]
    }
  ]
 },
 "server_cert": "",
 "server_key": ""
}
```

After the parameters are configured, you can use the functions of the numa-aware add-on and resource_exporter at the same time.

- Using **eas_service**:

```
{
 "ca_cert": "",
 "default_scheduler_conf": {
  "actions": "allocate, backfill",
  "tiers": [
    {
        "plugins": [
            {
                "name": "priority"
            },
            {
                "name": "gang"
            },
            {
                "name": "conformance"
            }
        ]
    },
    {
        "plugins": [
            {
```

```
                "name": "drf"
            },
            {
                "name": "predicates"
            },
            {
                "name": "nodeorder"
            }
        ]
    },
    {
        "plugins": [
            {
                "name": "cce-gpu-topology-predicate"
            },
            {
                "name": "cce-gpu-topology-priority"
            },
            {
                "name": "cce-gpu"
            },
            {
                "name": "eas",
                "custom": {
                    "availability_zone_id": "",
                    "driver_id": "",
                    "endpoint": "",
                    "flavor_id": "",
                    "network_type": "",
                    "network_virtual_subnet_id": "",
                    "pool_id": "",
                    "project_id": "",
                    "secret_name": "eas-service-secret"
                }
            }
        ]
    },
    {
        "plugins": [
            {
                "name": "nodelocalvolume"
            },
            {
                "name": "nodeemptydirvolume"
            },
            {
                "name": "nodeCSIscheduling"
            },
            {
                "name": "networkresource"
            }
        ]
    }
  ]
 },
 "server_cert": "",
 "server_key": ""
}
```

- Using **ief**:

```
{
 "ca_cert": "",
 "default_scheduler_conf": {
  "actions": "allocate, backfill",
  "tiers": [
   {
    "plugins": [
     {
      "name": "priority"
     },
```

```
                {
                    "name": "gang"
                },
                {
                    "name": "conformance"
                }
            ]
        },
        {
            "plugins": [
                {
                    "name": "drf"
                },
                {
                    "name": "predicates"
                },
                {
                    "name": "nodeorder"
                }
            ]
        },
        {
            "plugins": [
                {
                    "name": "cce-gpu-topology-predicate"
                },
                {
                    "name": "cce-gpu-topology-priority"
                },
                {
                    "name": "cce-gpu"
                },
                {
                    "name": "ief",
                    "enableBestNode": true
                }
            ]
        },
        {
            "plugins": [
                {
                    "name": "nodelocalvolume"
                },
                {
                    "name": "nodeemptydirvolume"
                },
                {
                    "name": "nodeCSIscheduling"
                },
                {
                    "name": "networkresource"
                }
            ]
        }
    ]
  },
  "server_cert": "",
  "server_key": ""
}
```

Retaining the Original Configurations of volcano-scheduler-configmap

If you want to use the original configurations after the add-on is upgraded, perform the following steps:

Step 1 Check and back up the original volcano-scheduler-configmap configuration.

Example:

```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
  default-scheduler.conf: |-
    actions: "enqueue, allocate, backfill"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
        arguments:
          binpack.cpu: 100
          binpack.weight: 10
          binpack.resources: nvidia.com/gpu
          binpack.resources.nvidia.com/gpu: 10000
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
```

Step 2 Enter the customized content in the **Parameters** area on the console.

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "enqueue, allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    },
                    {
                        "name": "binpack",
                        "arguments": {
                            "binpack.cpu": 100,
                            "binpack.weight": 10,
                            "binpack.resources": "nvidia.com/gpu",
                            "binpack.resources.nvidia.com/gpu": 10000
                        }
                    }
```

```
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
      }
    ]
  },
  "server_cert": "",
  "server_key": ""
}
```

NOTE

After the parameters are configured, the original content in volcano-scheduler-configmap will be overwritten. Therefore, you must check whether volcano-scheduler-configmap has been modified during the upgrade. If volcano-scheduler-configmap has been modified, synchronize the modification to the upgrade page.

----End

Related Operations

- **Dynamic Resource Oversubscription**
- **NUMA Affinity Scheduling**

Change History

NOTICE

You are advised to upgrade Volcano to the latest version that matches the cluster.

Table 1-56 Cluster version mapping

Cluster Version	Add-on Version
v1.25	1.7.1 and 1.7.2
v1.23	1.7.1 and 1.7.2
v1.21	1.7.1 and 1.7.2
v1.19.16	1.3.7, 1.3.10, 1.4.5, 1.7.1, and 1.7.2
v1.19	1.3.7, 1.3.10, and 1.4.5
v1.17 (End of maintenance)	1.3.7, 1.3.10, and 1.4.5
v1.15 (End of maintenance)	1.3.7, 1.3.10, and 1.4.5

Table 1-57 CCE add-on versions

Add-on Version	Supported Cluster Version	Updated Feature
1.9.1	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	• Fixed the issue that the counting pipeline pod of the networkresource add-on occupies supplementary network interfaces (Sub-ENI). • Fixed the issue where the binpack add-on scores nodes with insufficient resources. • Fixed the issue of processing resources in the pod with unknown end status. • Optimized event output. • Supports HA deployment by default.
1.7.2	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	• Supported Kubernetes 1.25. • Improved Volcano scheduling.
1.7.1	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	Supported Kubernetes 1.25.
1.6.5	/v1.19.*|v1.21.*|v1.23.*/	• Served as the CCE default scheduler. • Supported unified scheduling in hybrid deployments.
1.4.5	/v1.17.*|v1.19.*|v1.21.*/	• Changed the deployment mode of volcano-scheduler from **statefulset** to **deployment**. Fixed the issue that pods cannot be automatically migrated when the node is abnormal.

Add-on Version	Supported Cluster Version	Updated Feature
1.4.2	/v1.15.*|v1.17.*|v1.19.*|v1.21.*/	● Resolved the issue that cross-GPU allocation fails. ● Supported the updated EAS API.
1.3.3	/v1.15.*|v1.17.*|v1.19.*|v1.21.*/	● Fixed the scheduler crash issue caused by GPU exceptions and the admission failure issue for privileged init containers.
1.3.1	/v1.15.*|v1.17.*|v1.19.*/	● Upgraded the RAID controller card firmware to the latest version. ● Supported Kubernetes 1.19. ● Added the numa-aware add-on. ● Fixed the deployment scaling issue in the multi-queue scenario. ● Adjusted the algorithm add-on enabled by default.
1.2.5	/v1.15.*|v1.17.*|v1.19.*/	● Fixed the OutOfcpu issue in some scenarios. ● Fixed the issue that pods cannot be scheduled when some capabilities are set for a queue. ● Made the log time of the volcano component consistent with the system time. ● Fixed the issue of preemption between multiple queues. ● Fixed the issue that the result of the ioaware add-on does not meet the expectation in some extreme scenarios. ● Supported hybrid clusters.

Add-on Version	Supported Cluster Version	Updated Feature
1.2.3	/v1.15.*|v1.17.*|v1.19.*/	• Fixed the training task OOM issue caused by insufficient precision. • Fixed the GPU scheduling issue in CCE 1.15 and later versions. Rolling upgrade of CCE versions during task distribution is not supported. • Fixed the issue where the queue status is unknown in certain scenarios. • Fixed the issue where a panic occurs when a PVC is mounted to a job in a specific scenario. • Fixed the issue that decimals cannot be configured for GPU jobs. • Added the ioaware add-on. • Added the ring controller.

1.6.11.5 gpu-device-plugin

Introduction

gpu-device-plugin is an add-on that supports GPUs in containers. If GPU nodes are used in the cluster, this add-on must be installed.

Constraints

- The driver to be downloaded must be a **.run** file.
- Only NVIDIA Tesla drivers are supported.
- When installing or reinstalling the add-on, ensure that the driver download address is correct and accessible. CCE does not verify the address validity.
- gpu-device-plugin enables you to download the driver and execute the installation script. The add-on status does not indicate whether the driver is installed successfully.
- If a node has multiple A100 or A800 GPUs, you need to manually install nvidia-fabricmanager that matches your driver version. For details, see **Installing the nvidia-fabricmanager Service**.

Installing the Add-on

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

Step 2 Locate **gpu-device-plugin** in **Add-ons Available** and click **Install**.

Step 3 In the window that slides out from the right, configure the parameters as follows:

- **Add-on Specifications**: Select **Default** or **Custom** as required.
- **Containers**: This parameter can be configured only when **Add-on Specifications** is set to **Custom**.
- **NVIDIA Driver**: Use a driver address provided by CCE or enter the address of your custom NVIDIA driver. All GPU nodes in the cluster use the same driver.

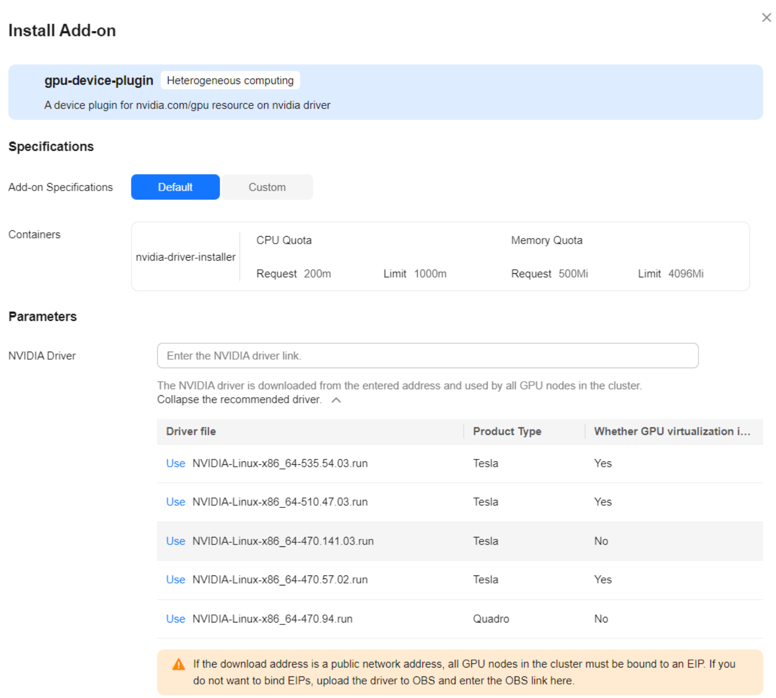
  GPU virtualization is available only in versions 470.57.02, 470.103.01, 470.141.03, 510.39.01, and 510.47.03.

  You are advised to use a driver address provided by CCE to match the driver version.

NOTICE

- If the download link is a public network address, for example, NVIDIA official website address (https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run), associate EIPs with GPU nodes. For details about how to obtain the driver link, see **Obtaining the Driver Link from Public Network**.
- If the download link is an OBS URL, there is no need to bind an EIP to each GPU node. For details about how to obtain the driver link, see **Obtaining the Driver Link from OBS**.
- Ensure that the NVIDIA driver version matches the GPU node.
- If the driver version is changed, restart the node to apply the change.
- Use driver 470 or later for Huawei Cloud EulerOS 2.0 or Ubuntu 22.04 on which Linux Kernel 5.*x* is built.

Figure 1-42 Installing gpu-device-plugin



Step 4 Click **Install**.

----End

Verifying the Add-on

After the add-on is installed, run the **nvidia-smi** command on the GPU node and the container that schedules GPU resources to verify the availability of the GPU and driver.

GPU node:

```
cd /usr/local/nvidia/bin &&./nvidia-smi
```

Container:

```
nvidia-smi
```

If GPU information is returned, the GPU is available and the add-on is successfully installed.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.118.02   Driver Version: 440.118.02   CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   31C    P0    23W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Obtaining the Driver Link from Public Network

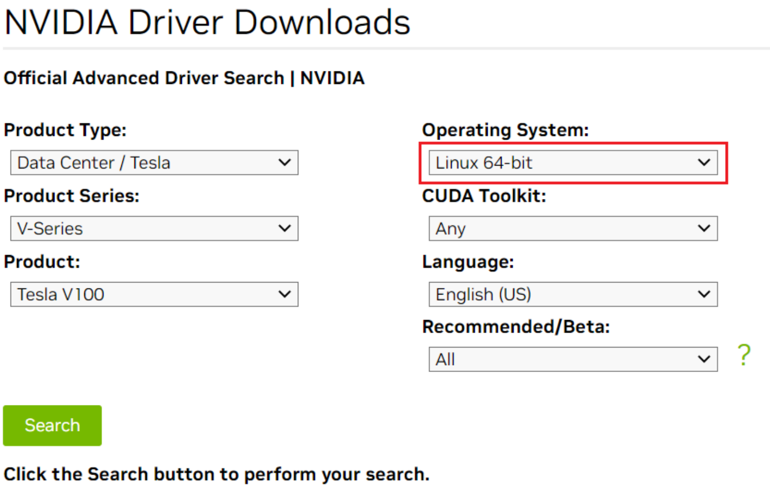
Step 1 Log in to the CCE console.

Step 2 Click **Create Node** and select the GPU node to be created in the **Specifications** area. The GPU card model of the node is displayed in the lower part of the page.

Step 3 Log in to **NVIDIA**.

Step 4 Select the driver information on the **NVIDIA Driver Downloads** page, as shown in **Figure 1-43**. **Operating System** must be **Linux 64-bit**.

Figure 1-43 Setting parameters



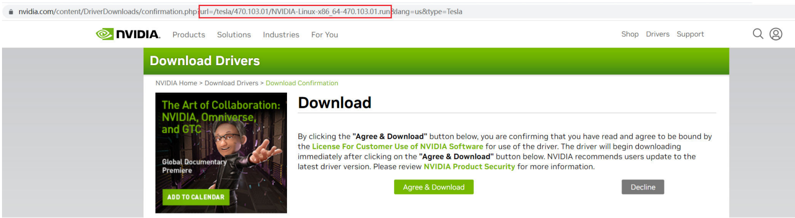
Step 5 After confirming the driver information, click **SEARCH**. A page is displayed, showing the driver information, as shown in **Figure 1-44**. Click **DOWNLOAD**.

Figure 1-44 Driver information

Step 6 Obtain the driver link in either of the following ways:

- Method 1: As shown in **Figure 1-45**, find *url=/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run* in the browser address box. Then, supplement it to obtain the driver link (https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run). By using this method, you must associate EIPs with GPU nodes.
- Method 2: As shown in **Figure 1-45**, click **Agree & Download** to download the driver. Then, upload the driver to OBS and record the OBS URL. By using this method, you do not need to associate EIPs with GPU nodes.

Figure 1-45 Obtaining the link



----End

Obtaining the Driver Link from OBS

Step 1 Upload the driver to OBS and set the driver file to public read. For details, see **Uploading a File**.

> 📖 **NOTE**
>
> When the node is restarted, the driver will be downloaded and installed again. Ensure that the OBS bucket link of the driver is valid.

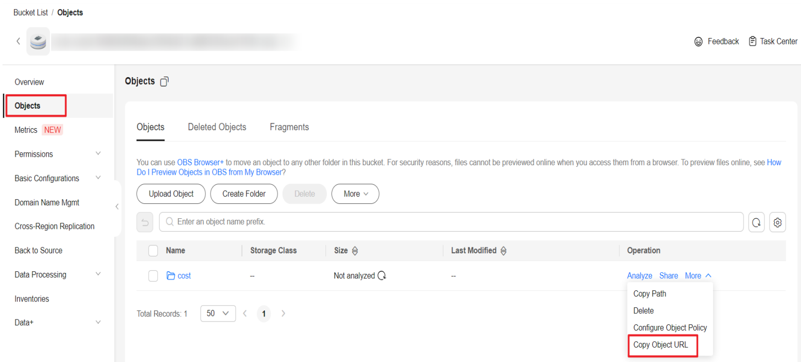
Step 2 Log in to the OBS console. In the navigation pane on the left, select **Object Storage**.

Step 3 In the bucket list, click the bucket name you want. The **Overview** page of the bucket is displayed.

Step 4 In the navigation pane on the left, choose **Objects**.

Step 5 Locate the target object and choose **More** > **Copy Object URL** to copy the driver link.

Figure 1-46 Obtaining the driver link



----End

Installing the nvidia-fabricmanager Service

A100 and A800 GPUs support NvLink and NvSwitch. If you use a node with multiple GPUs, you need to install the nvidia-fabricmanager service corresponding to your driver version to enable interconnection between GPUs. Otherwise, GPU pods may fail to be used.

This section uses driver 470.103.01 as an example. You can perform the following steps to install the driver. Replace the driver version as required.

Step 1 Log in to the target GPU node. An EIP must be bound to the node to download the nvidia-fabricmanager service.

Step 2 Install the nvidia-fabricmanager service corresponding to your driver version. You can download the installation package corresponding to your OS and driver version from the **official website**.

- **CentOS**

  Take CentOS 7 as an example:

  ```
  driver_version=470.103.01
  wget https://developer.download.nvidia.cn/compute/cuda/repos/rhel7/x86_64/cuda-drivers-fabricmanager-${driver_version}-1.x86_64.rpm
  rpm -ivh nvidia-fabric-manager-${driver_version}-1.x86_64.rpm
  ```

- **Other OSs such as Ubuntu**

  Take Ubuntu 18.04 as an example:

  ```
  driver_version=470.103.01
  driver_version_main=$(echo $driver_version | awk -F '.' '{print $1}')
  wget https://developer.download.nvidia.cn/compute/cuda/repos/ubuntu1804/x86_64/nvidia-fabricmanager-${driver_version_main}_${driver_version}-1_amd64.deb
  dpkg -i nvidia-fabricmanager-${driver_version_main}_${driver_version}-1_amd64.deb
  ```

Step 3 Start the nvidia-fabricmanager service.

```
systemctl enable nvidia-fabricmanager
systemctl start nvidia-fabricmanager
```

Step 4 Run the following command to check the nvidia-fabricmanager service status:

```
systemctl status nvidia-fabricmanager
```

----End

Helpful Links

- **How Do I Troubleshoot gpu-beta and GPU Driver Problems?**
- **What Should I Do If GPU Node Exceptions Occur?**
- **GPU Scheduling**

1.6.11.6 e-backup

Introduction

e-backup is a subsystem in Everest 2.0 (cloud native storage system) for protecting cloud native application data. With e-backup, you can back up application data (Kubernetes resources) and service data (data in PVs) to OBS and restore backup data to a specified cluster.

The backup and restoration functions of e-backup are available for:

- Single cluster DR

  The data of applications in a cluster is periodically backed up. When the cluster or an application is damaged, you can redeploy the application to the cluster to take over services in disaster scenarios.

- Intra-cluster/Cross-cluster clone

  If multiple applications need to be cloned across clusters, especially the applications that have been working in a cluster for a period of time, their data is backed up and then restored to different namespaces in the same cluster or other clusters.

- Cross-cluster/Cross-cloud migration

  If applications need to be migrated from a cluster to another cluster across regions or from another cloud to CCE due to network, cost, or service location changes, their data is backed up and then restored to the destination cluster.

Constraints

- The cluster version must be 1.15 or later and have at least one available node.
- When e-backup is installed in a cluster, the cluster image can be pulled from SWR.
- To prevent failures or incomplete data, you cannot add, delete, or modify the cluster during the backup or restoration. If there are any changes to a cluster, you are advised to wait for 15 minutes until the cluster is stable and then perform the backup operation.
- e-backup integrates the PV data backup capability of restic. e-backup can create a snapshot for the data at the backup time point and upload the data, which does not affect subsequent data read and write. However, restic does not verify the file content and service consistency.
- The memory occupied by restic depends on the size of the PV data backed up for the first time. If there is more than 300 GB of data, use the data migration method provided by the cloud storage. If you use application data management to migrate a large amount of PV data, you can modify the resource levels of the restic instance. For details, see **Modifying Add-on Settings**.
- e-backup complies with velero and restic constraints. For example, during the restoration, the Service will clear the ClusterIP to better adapt to the differences between the source and target clusters.
- When restoring an application in a CCE cluster that uses a secret (cfe/secure-opaque) for data encryption to another cluster, you need to manually create a secret with the same name and type as the original cluster. This ensures that the restored application runs normally.

Installing e-backup

NOTICE

e-backup depends on the custom resource **BackupStorageLocation** and its secret to execute backup and restore tasks. However, the resource will change if it is uninstalled and reinstalled. As a result, if you uninstall e-backup, existing backups may not be restored.

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Add-ons**. In the **Add-ons Available** area, click **Install** of e-backup.

Step 3 Configure the parameters as described in **Table 1-58**.

Table 1-58 e-backup parameters

Parameter	Description
Add-on Specifications	Select **Standalone**.
Containers	Configure resource levels for the add-on instance. ● velero: backup and restoration of Kubernetes metadata. ● restic: backup and restoration of application data storage volumes. **NOTE** ● To ensure the add-on instance can be scheduled, reserve sufficient resources in the cluster. ● To create an add-on instance, ensure the request is no more than the limit. ● To avoid add-on faults, adjust the resource limit based on the amount of data to be backed up or restored.

Step 4 Configure **volumeWorkerNum**.

volumeWorkerNum indicates the number of concurrent data volume backup tasks, which defaults to **3**.

```
{
  "volumeWorkerNum": 3
}
```

Step 5 Click **Install** and check the add-on status on the **Add-ons** page.

Running indicates the add-on has been installed in the cluster.

----End

Modifying Add-on Settings

Step 1 Access the cluster details page.

Step 2 In the navigation pane, choose **Add-ons**. In the **Add-ons Installed** area, click **Edit** of e-backup.

Step 3 Modify the add-on settings. For details about related parameters, see **Table 1-58**.

Step 4 Click **OK**. The add-on is in the **Upgrading** state. After the upgrade is complete, new settings will be used.

----End

2 Fleets

2.1 Overview

Fleets

A fleet contains multiple clusters. You can use fleets to classify associated clusters. You can also use fleets to manage multiple clusters based on permissions, security policies, configurations, and multi-cluster orchestration.

Constraints

- Only **Huawei Cloud accounts** and users with the **UCS FullAccess** permission can create and delete fleets.
- A cluster can be added to only one fleet.

Capabilities Supported by Fleets

After a cluster is connected to UCS, you can add the cluster to a fleet and enable cluster federation for the fleet for multi-cluster management. UCS supports different capabilities for clusters that have been connected to it (regardless of whether they are added to fleets), fleets without cluster federation enabled, and fleets with cluster federation enabled, as listed in **Table 2-1**.

Table 2-1 Capabilities supported by fleets

Capabilities.	Clusters Connected to UCS	Fleets Without Cluster Federation	Fleets with Cluster Federation
Federation	-	-	√
Traffic Distribution	√	-	-
Observability	√	√	√

Capabilities.	Clusters Connected to UCS	Fleets Without Cluster Federation	Fleets with Cluster Federation
Service Meshes	-	√	√
Operator Service Center	√	-	-
Policy Center	√	√	√
Configuration Management	√	-	-
Pipeline	-	-	√
Permissions	√	√	√

2.2 Managing Fleets

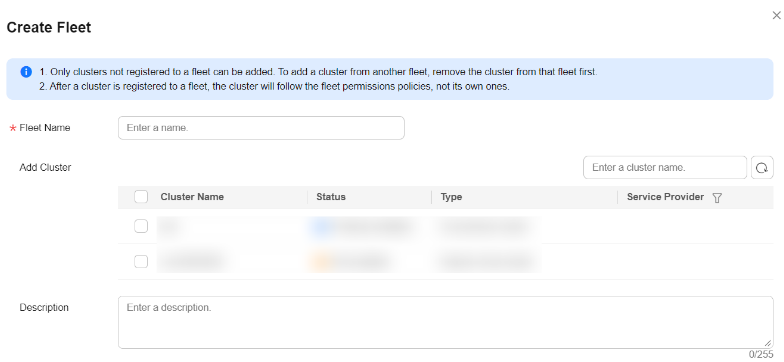
This section describes how to create a fleet, add clusters to the fleet, associate a permission policy with the fleet, remove clusters from the fleet, unregister clusters from the fleet, and delete the fleet.

Creating a Fleet

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**. On the **Fleets** tab, click **Create Fleet**.

Step 2 Enter the fleet information.

Figure 2-1 Creating a fleet



- **Fleet Name**: Enter a name, starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **Add Cluster**: Clusters not in the fleet are displayed in the list. You can add clusters when creating a fleet or after the fleet is created. If you do not select any cluster, an empty fleet will be created. After the fleet is created, see **Adding a Cluster**.

- **Description**: description of the fleet to which the cluster is added

NOTE

A registered cluster will follow the fleet permissions policies, not its own ones.

Step 3 Click **OK**.

----End

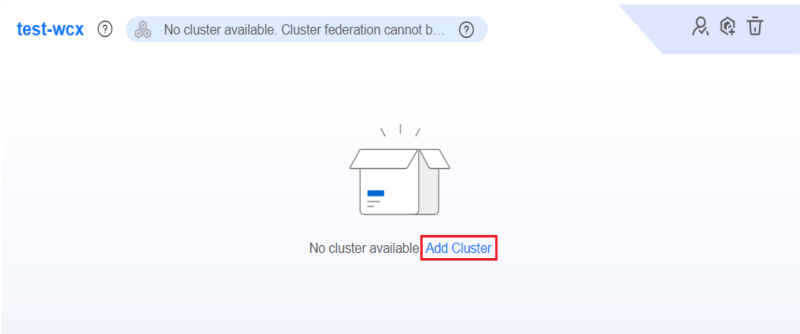
Adding a Cluster

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 In the card view of the destination fleet, click **Add Cluster**, or click  in the upper right corner.

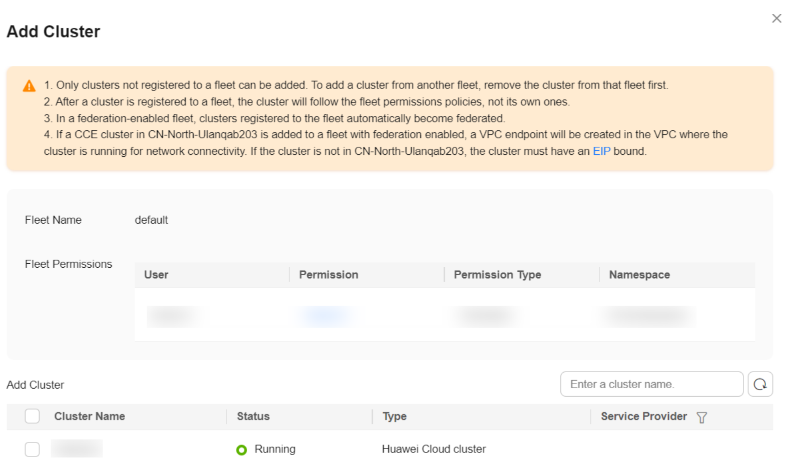
You can also click the fleet name to go to the fleet details page and click **Add Cluster** in the upper right corner of the **Container Clusters** page.

Figure 2-2 Adding a cluster to a fleet



Step 3 Select one or more existing clusters. A cluster can only be added to one fleet. The clusters displayed in the list are those have not been added to any fleet.

Figure 2-3 Adding a cluster



NOTE

- A registered cluster will follow the fleet permissions policies, not its own ones.
- In a federation-enabled fleet, registered clusters automatically become federated. For details about cluster federation, see **Enabling Cluster Federation**.

Step 4 Click **OK**.

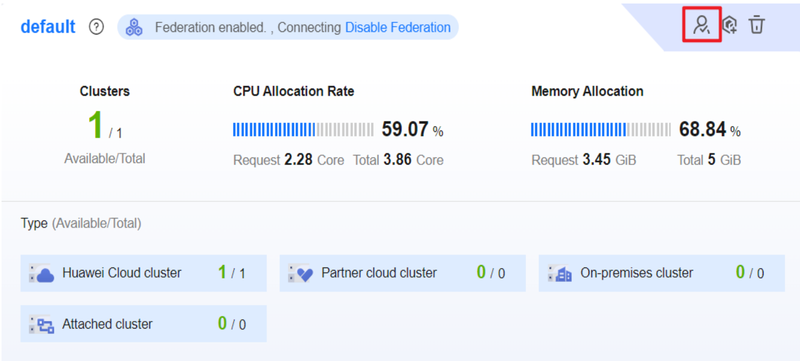
----End

Associating a Permission Policy

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

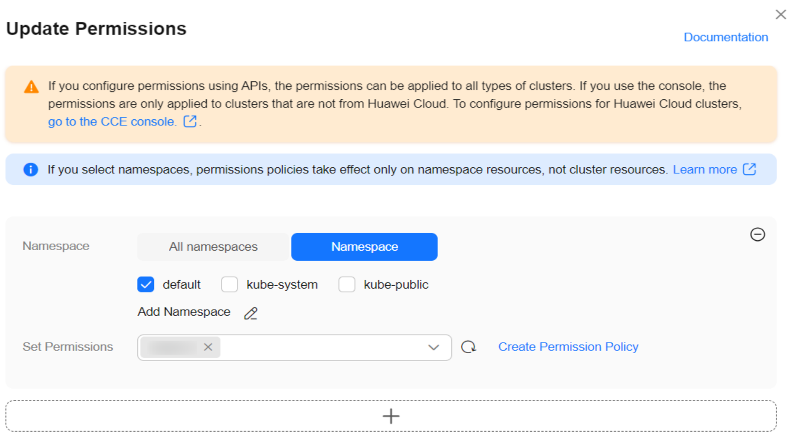
Step 2 In the card view of the target fleet, click  in the upper right corner.

Figure 2-4 Associating a permission policy with a fleet



Step 3 On the displayed page, click **Update Fleet Permissions** or **Set Permissions**. Then, associate the created permission policy with the namespace of the fleet.

Figure 2-5 Updating a permission policy



- **Namespace**: Select **All namespaces** or **Namespace**. **All namespaces** includes the existing namespace of the fleet and the namespace to be added to the fleet. **Namespace** indicates the custom range of namespaces. UCS provides several common namespaces, such as **default**, **kube-system**, and **kube-public**. You can also add a namespace, which should exist in the cluster.

If you select namespaces, permission policies take effect only on namespace resources, not cluster resources. For details about namespace and cluster resources, see **Kubernetes Resource Objects**.

- **Set Permissions**: Select permissions from the drop-down list box. You can select multiple permissions at a time to batch grant permissions.

If different namespaces are associated with different permission policies (for example, the **default** namespace is associated with the **readonly** permission policy and the **development** namespace is associated with the **develop** permission policy), you can click + to add multiple relationships of permission granting.

Step 4 Click **OK**.

If you need to update the permission policy of the fleet, select the namespace and permission again using the preceding method.

----End

Removing a Cluster from a Fleet

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab page, click a fleet name. The fleet details page is displayed.

Step 3 In the navigation pane, choose **Container Clusters**. In the card view of the destination cluster, click the icon in the upper right corner.

Step 4 Read the precautions carefully and confirm the risks. Then click **OK**.

After a cluster is removed from a fleet, it is displayed on the **Clusters Not in Fleet** tab page. You can add the cluster to the fleet again. For details, see **Managing Clusters Not in the Fleet**.

----End

Unregistering a Cluster from a Fleet

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab page, click a fleet name. The fleet details page is displayed.

Step 3 In the navigation pane on the left, choose **Container Clusters**. In the card view of the target cluster, click the icon in the upper right corner.

Step 4 In the displayed **Unregister Cluster** dialog box, read the precautions carefully, confirm the risks, and click **OK**.

Step 5 (Optional) After an attached cluster is unregistered, run the following command to uninstall the agent component from the destination cluster:

kubectl -n kube-system delete deployments/proxy-agent secret/proxy-agent-cert

Step 6 (Optional) After an on-premises cluster is unregistered, run the uninstallation command to delete the cluster from the local host and clear resources:

./ucs-ctl delete cluster [*Cluster name*]

NOTE

If the cluster fails to be deleted, perform operations in **How Do I Manually Clear Nodes of an On-premises Cluster?**.

----End

Deleting a Fleet

If a fleet is no longer used, you can delete it. There are two restrictions on deletion: there is no cluster in the fleet and cluster federation has been disabled for the fleet. If there are clusters in the fleet, you can **remove the clusters from the fleet** and then add them to another fleet. If cluster federation has been enabled for the fleet, disable it following **Disabling Cluster Federation**.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, locate the target fleet and click in the upper right corner.

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

----End

2.3 Managing Clusters Not in the Fleet

Clusters for which a fleet is not selected during registration or clusters removed from a fleet will be displayed on the **Clusters Not in Fleet** tab. This section describes how you can manage clusters that are not added to a fleet, including adding clusters to a fleet and associate a permission policy with the fleet.

Registering Clusters to a Fleet

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 Click the **Clusters Not in Fleet** tab. In the card view of the target cluster, click in the upper right corner.

Step 3 Select a fleet. A registered cluster will follow the fleet permissions policies, not its own ones.

Step 4 After you select a fleet, the current permission and adjusted permission are displayed. Confirm the information and click **OK**.

After the cluster is registered to a fleet, the cluster is displayed in the fleet and will be centrally managed by the fleet.

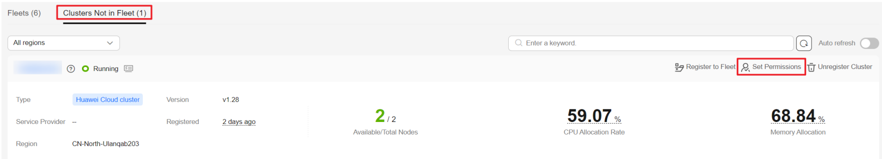
----End

Associating a Permission Policy

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 Choose the **Clusters Not in Fleet** tab page and click in the upper right corner of the card view of the destination cluster.

Figure 2-6 Viewing clusters



Step 3 On the displayed page, click **Update Fleet Permissions**. Then, associate the created permission policy with the namespace of the cluster.

- **Namespace**: Select **All namespaces** or **Namespace**. **All namespaces** includes the existing namespace of the cluster and the namespace to be added to the cluster. **Namespace** indicates the custom range of namespaces. UCS provides several common namespaces, such as **default**, **kube-system**, and **kube-public**. You can also add a namespace, which should exist in the cluster.

  If you select namespaces, permission policies take effect only on namespace resources, not cluster resources. For details about namespace and cluster resources, see **Kubernetes Resource Objects**.
- **Set Permissions**: Select permissions from the drop-down list box. You can select multiple permissions at a time to batch grant permissions.

If different namespaces are associated with different permission policies (for example, the **default** namespace is associated with the **readonly** permission policy and the **development** namespace is associated with the **develop** permission policy), you can click to add multiple relationships of permission granting.

Step 4 Click **OK**.

If you need to update the permission policy of the cluster, select the namespace and permission again using the preceding method.

----End

Unregistering a Cluster

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 Choose the **Clusters Not in Fleet** tab page and click in the upper right corner of the card view of the destination cluster.

Step 3 In the displayed **Unregister Cluster** dialog box, read the precautions carefully, confirm the risks, and click **OK**.

Step 4 (Optional) After an attached cluster is unregistered, run the following command to uninstall the agent component from the destination cluster:

kubectl -n kube-system delete deployments/proxy-agent secret/proxy-agent-cert

Step 5 (Optional) After an on-premises cluster is unregistered, run the uninstallation command to delete the cluster from the local host and clear resources:

./ucs-ctl delete cluster [*Cluster name*]

NOTE

If the cluster fails to be deleted, perform operations in **How Do I Manually Clear Nodes of an On-premises Cluster?**.

----End

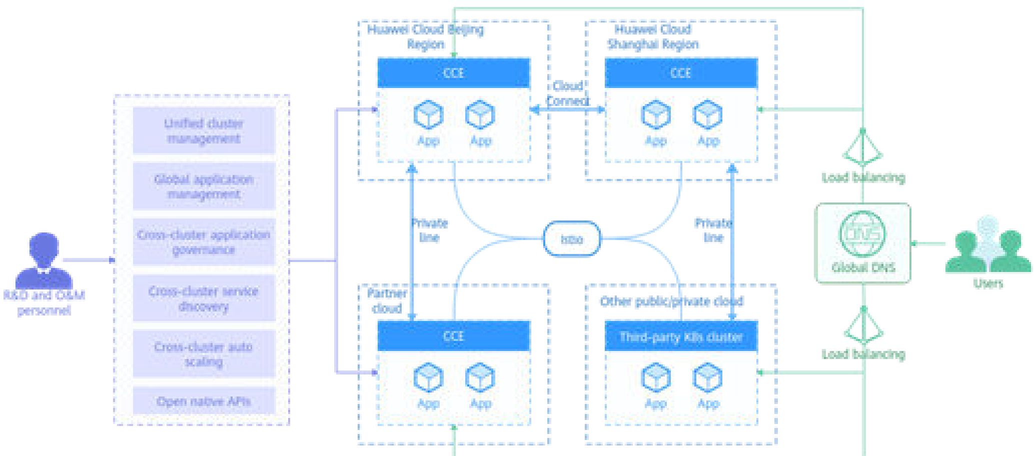
3 Cluster Federation

3.1 Overview

Introduction

Cluster federation is a multi-cloud container orchestration capability provided by **Karmada**. Cluster federation aims to manage multi-cluster applications in cross-cloud and cross-region scenarios, with features such as unified multi-cluster management, application deployment, service discovery, auto scaling, and failover.

Figure 3-1 Cluster federation architecture



Constraints

Only **Huawei Cloud** accounts or users with the **UCS FullAccess** permission can enable or disable cluster federation.

Usage

Figure 3-2 shows how to use cluster federation.

Figure 3-2 Process of using cluster federation

Cluster federation is bound to fleets. To use cluster federation for multi-cluster management, perform the following operations:

- **Connect** the cluster to be managed to UCS and **add it to a fleet**.
- **Enable cluster federation** for the fleet and **use kubectl to connect the cluster to a federation**.
- (Optional) To use the latest functions, **upgrade the federation to its latest version**.

3.2 Enabling Cluster Federation

Enabling Cluster Federation

You can enable cluster federation for a fleet with just a few clicks.

Enabling cluster federation involves two phases: enabling cluster federation and adding clusters to the federation. Enabling cluster federation for a fleet will federate the registered clusters in the fleet.

There is a quota limit for enabling cluster federation, and there are constraints on clusters in a fleet. Before enabling cluster federation, **read the following constraints** to avoid failures.

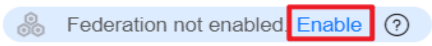
Table 3-1 Cluster constraints

Item	Constraint
Cluster version	The versions of all clusters in the fleet must be 1.19 or later.
Cluster status	All clusters in the fleet must be in the **Running** status.
Cluster network	• CCE clusters and CCE Turbo clusters: If a CCE cluster is located in AP-Singapore, UCS automatically creates a VPC endpoint in the VPC that the cluster belongs to when the federation is enabled. If a CCE cluster is not located in AP-Singapore, nodes in the cluster must be able to access the public network. For example, associate a public network address with a node to pull a public network image. • Other clusters: Ensure that the clusters connect to UCS.
Quota	The cluster federation quota is 1. This means cluster federation can be enabled only for one fleet.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, locate the target fleet displayed with **Federation not enabled**. Click **Enable**.

Figure 3-3 Enabling cluster federation



Step 3 In the displayed dialog box, click **OK**. Then, wait until cluster federation is enabled.

If the clusters in the fleet do not meet the constraints, an error message will be displayed. Modify the clusters as prompted and enable cluster federation again.

It takes about 10 minutes to enable cluster federation. You can click the federation status to view the detailed enabling progress. After cluster federation is enabled, a message indicating that cluster federation has been enabled and clusters have been federated is displayed on the top of the fleet.

----End

Adding Clusters

After cluster federation is enabled for a fleet, you can continue to add clusters to the fleet. The new clusters are automatically federated. A federation can have a maximum of 20 clusters.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 In the card view of the target fleet, click **Add Cluster**, or click  in the upper right corner.

You can also click the fleet name to go to the fleet details page and click **Add Cluster** in the upper right corner of the **Container Clusters** page.

Step 3 Select one or more existing clusters. A cluster can only be added to one fleet. The clusters displayed in the list are those have not been added to any fleet.

NOTE

Ensure that the selected clusters meet the constraints described in **Table 3-1**. Otherwise, the clusters can be added to the fleet but fail to be federated. If clusters fail to be federated, perform operations in **Why Cannot I Enable Cluster Federation for a Fleet or Register a Cluster to a Fleet After Cluster Federation Is Enabled?**

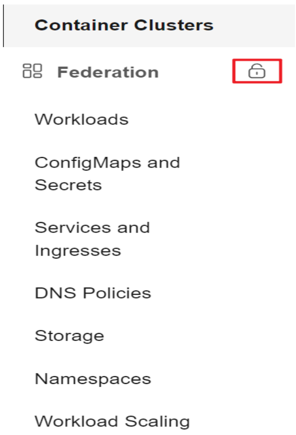
Step 4 Click **OK**.

----End

Managing Federation

After cluster federation is enabled for a fleet, the **Federation** module on the fleet's details page is automatically unlocked.

Figure 3-4 Managing federation



Next, you can create federated resources such as federated workloads, Services, and storage for deploying your service. You can also perform advanced operations such as multi-active DR and auto scaling for multi-cluster applications.

Disabling Cluster Federation

If you do not need to use cluster federation, you can disable it. After cluster federation is disabled, services running on the workloads are not affected.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab page, locate the target fleet and click **Disable Federation** in the upper right corner.

Figure 3-5 Disabling cluster federation

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

----End

FAQ

Why Cannot I Enable Cluster Federation for a Fleet or Register a Cluster to a Fleet After Cluster Federation Is Enabled?

3.3 Using kubectl to Connect to a Federation

This section describes how you can use kubectl to connect to a federation.

Permissions

When you use kubectl to connect to a federation, UCS uses **kubeconfig.json** generated on the federation for authentication. This file contains user information, based on which UCS determines which Kubernetes resources can be accessed by kubectl. The permissions recorded in a **kubeconfig.json** file vary from user to user.

Constraints

- For security purposes, the federation API server does not have a public IP address. UCS creates an endpoint in your VPC and subnet and connects the endpoint to the federation API server for the access to the federation. For each federation, only one endpoint is created in the same VPC. If a VPC already has an endpoint for connecting to the federation API server, the endpoint will be reused.
- Currently, the kubectl configuration file can be downloaded only for projects in AP-Singapore.

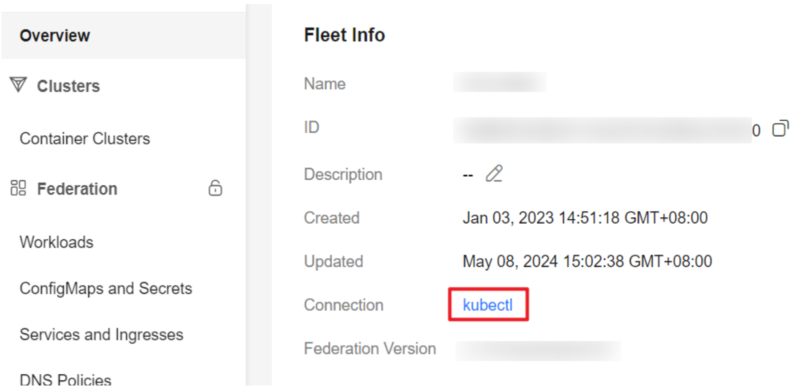
Prerequisites

- Before using kubectl to connect to a federation, ensure that the federation has been enabled (**Enabling Cluster Federation**) and is running normally.
- Only the client in a VPC can connect to a federation using kubectl. If there is no client in the VPC, create one.
- kubectl has been downloaded and uploaded to the client. For details about how to download kubectl, see **Kubernetes releases**.
- At least the custom policy **iam:clustergroups:get** is configured.

Using kubectl to Connect to a Federation

Step 1 Log in to the UCS console and click the name of the target fleet to go to its details page. Then, click **kubectl** in **Fleet Info**.

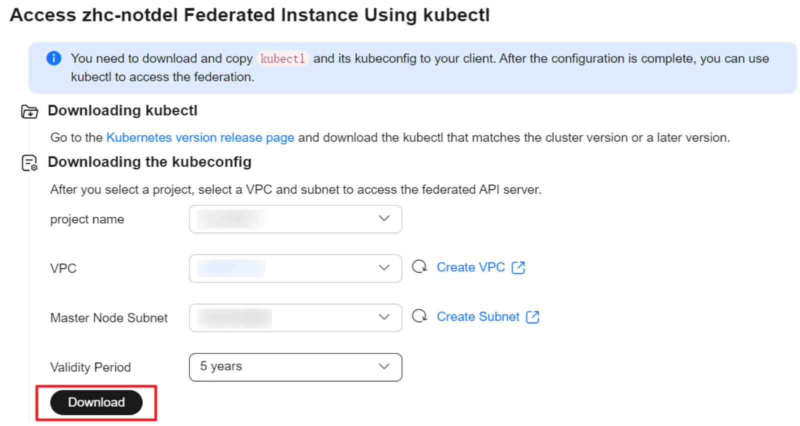
Figure 3-6 kubectl connection



Step 2 Select a project, VPC, master node subnet, and validity period as prompted and click **Download** to download the kubectl configuration file.

The name of the downloaded file is **kubeconfig.json**.

Figure 3-7 Using kubectl to connect to a federation instance



NOTICE

- If the **kubeconfig.json** file is leaked, your clusters may be attacked. Keep it secure.
- The validity period of the kubectl configuration file can be set as required. The options are 5 years, 1 year, 6 months, 30 days, and 15 days to 1 day. The minimum value is 1 day.

Step 3 Install and configure kubectl on the executor.

1. Copy kubectl and its configuration file to the **/home** directory on the executor in the selected VPC and subnet.
2. Log in to your executor 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
```

----End

Resources and Operations Supported by a Federation

Table 3-2 lists the resources and operations supported by a federation. In the table, "√" means the operation can be performed on related resources. "Partially supported" means the operation can be performed on part of the resources. If there is neither "√" nor "partially supported", the operation cannot be performed on related resources.

Table 3-2 Resources and operations supported by a federation

Group/Version	Resource	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
core/v1	pods	√	√	√	√	√	√	√

Group/Version	Resource	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
	pods/log	√						
	pods/exec	√			√			
	pods/status	√						
	configmaps	√	√	√	√	√	√	√
	secrets	√	√	√	√	√	√	√
	services	√	√	√	√	√	√	√
	nodes	√	√	√		√	√	
	namespaces	√	√	√	√	√	√	√
	endpoints	√	√					
	events	√	√					
	limitranges	√	√					
	resourcequotas	√	√					
	persistentvolume-claims	√	√					
	persistentvolumes	√	√					
	serviceaccounts	√	√					
admissionregistration.k8s.io/v1	mutatingwebhook-configurations	√	√					
	validatingwebhook configurations	√	√					
apiextensions.k8s.io/v1	customresourcede-finitions	√	√	√	√	√	√	√
apiregistration.k8s.io/v1	apiservices	√	√					
apps/v1	deployments	√	√	√	√	√	√	√
	deployments/scale	√				√		
	deployments/status	√						
	daemonsets	√	√	√	√	√	√	√
	daemonsets/status	√						
	statefulsets	√	√	√	√	√	√	√

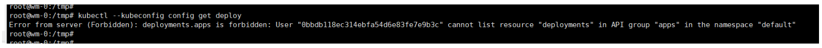
Group/Version	Resource	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
	statefulsets/status	√						
	replicasets	√	√					
autoscaling/(v1, v2, v2beta1, and v2beta2)	horizontalpodautoscalers	√	√	√	√	√	√	√
batch/v1	jobs	√	√	√	√	√	√	√
	jobs/status	√						
	cronjobs	√	√	√	√	√	√	√
	cronjobs/status	√						
discovery.k8s.io/v1	endpointslices	√	√					
events.k8s.io/v1	events	√	√					
networking.k8s.io/v1	ingresses	√	√	√	√	Partially supported	Partially supported	√
	ingressclasses	√	√					
	networkpolicies	√	√					
policy/(v1 and v1beta1)	poddisruptionbudgets	√	√	√	√	√	√	√
rbac.authorization.k8s.io/v1	clusterrolebindings	√	√	√	√	√	√	√
	clusterroles	√	√	√	√	√	√	√
	rolebindings	√	√	√	√	√	√	√
	roles	√	√	√	√	√	√	√
storage.k8s.io/v1	storageclasses	√	√					

CAUTION

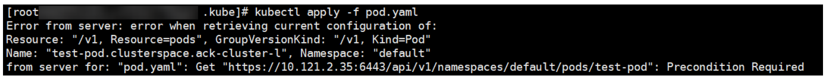
- Custom resources in a cluster can be operated through the federation entry only after the CRD is **registered** with the federation.
- The UPDATE and PATCH operations of Ingress objects can be only performed on resources in the federation control plane, not in member clusters.

FAQ

- If the following error message is displayed when you access federated resources, you do not have permissions to operate the resources. In this case, apply for permissions by referring to **Cluster Federation RBAC Authorization**.

- If "Precondition Required" is displayed when you access the federation and cluster resources, the cluster may be disconnected due to network problems or member cluster faults. Perform the following steps to locate the fault.

  a. Check whether the member cluster is running normally and whether it can be connected to UCS. For details, see **Why Cannot I Connect an Attached Cluster to UCS?**

  b. Check whether the proxy-agent deployed in the member cluster is running normally. For details, see **Why Does proxy-agent Fail to Run?**

- If "no such host" is displayed, perform the following steps to locate the fault.

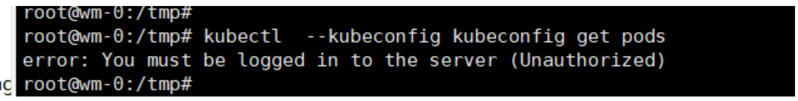
  

  a. Check whether the VPC endpoint exists or is deleted accidentally.

  ```
  server=`cat config  | jq '.clusters[0].cluster.server'` echo ${server:15:36}
  ```

  b. If the VPC endpoint exists, check whether the executor and the VPC endpoint are in the same VPC and whether the network connection between them is normal.

- If "You must be logged in to the server (Unauthorized)" is displayed, perform the following steps to locate the fault.

  a. Check whether the certificates are correct.

  Save the certificates to a temporary file.

  ```
  cd ~/.kube
  cat config  | jq '.clusters[0].cluster."certificate-authority-data"' | tr -d '"' | base64 -d > ca.crt
  cat config  | jq '.users[0].user."client-certificate-data"' | tr -d '"' | base64 -d > tls.crt
  cat config  | jq '.users[0].user."client-key-data"' | tr -d '"' | base64 -d > tls.key
  ```

  Check whether the TLS certificate and the CA certificate match.

  ```
  openssl verify -CAfile ca.crt tls.crt
  ```

  If "tls.crt: OK" is displayed, the CA certificate is correct. If "tls.crt: OK" is not displayed, download the kubeconfig file again.

b. Check whether the public key and private key of the certificate match.

```
diff -eq <(openssl x509 -pubkey -noout -in tls.crt) <(openssl rsa -pubout -in tls.key)
```

If "writing RSA key" is displayed, the public key and private key match. If they do not match, download the kubeconfig file again. After the verification is complete, delete the temporary file.

```
rm -f ca.crt tls.crt tls.key
```

c. Check whether the certificate has expired.

Save the certificate to a temporary file.

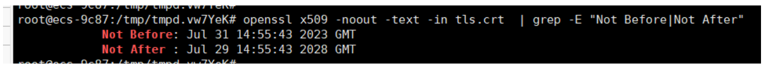
```
cd ~/.kube
cat config  | jq '.users[0].user."client-certificate-data"' | tr -d '"' | base64 -d > tls.crt
```

Check the certificate validity period.

```
openssl x509 -noout -text -in tls.crt  | grep -E "Not Before|Not After"
```

The certificate validity period is shown in the following figure. Check whether the current certificate is within the validity period. If the certificate expires, download the kubeconfig file again and delete the temporary file.

```
rm -f tls.crt
```

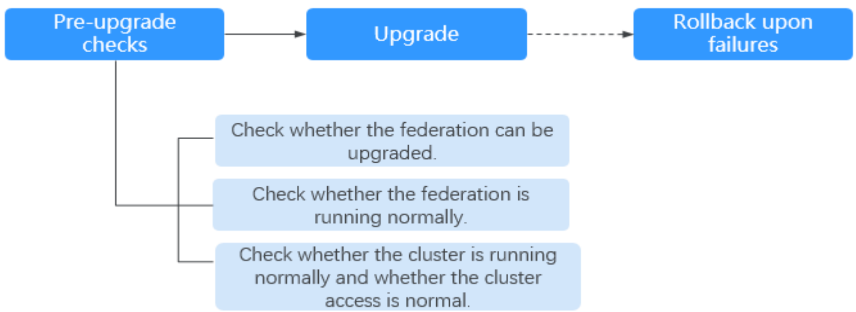


3.4 Upgrading a Federation

After a new federation version is released, you can upgrade the existing federation version to use functions supported by the new version. For details about the features in each version, see **Federation Upgrade Path**.

The federation upgrade process includes pre-upgrade check, upgrade, and rollback upon failures, as shown in **Federation Upgrade Process**. You can upgrade the federation version on the UCS console.

Figure 3-8 Federation upgrade process



1. Pre-upgrade checks

   Before the federation upgrade, UCS checks the federation running status, cluster running status, and cluster access status to avoid upgrade failures. If any exception is detected, rectify the fault as prompted on the console.

2. Upgrade

   Upgrade the federation.

3. Rollback upon failures

If the upgrade fails, you can upgrade the federation again or roll back the federation to the original version.

Federation Upgrade Path

After the latest federation version is available on UCS, UCS will describe the changes in this version. The following table describes the target version to which the federation version can be upgraded and the version differences.

Table 3-3 Federation version description

Version	Description
v1.7.0-r14	Fixed some bugs in operating federation resources using kubectl.

Upgrading a Federation

UCS allows you to view the federation version and upgrade the federation to a later version.

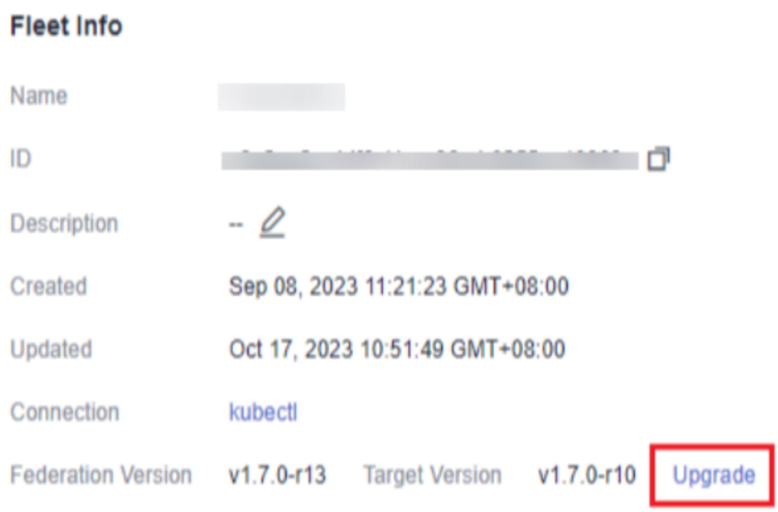
CAUTION

During the federation upgrade, do not move the cluster into or out of the cluster, or perform federation operations. Otherwise, the federation upgrade may fail.

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 On the **Fleets** tab, click the name of the fleet whose federation needs to be upgraded, and click **Upgrade** in **Fleet Info**.

Figure 3-9 Upgrading the federation version



Step 3 In the displayed dialog box, check the target version and click **Next** to perform the pre-upgrade check.

Step 4 If the check is passed, click **Start Upgrade** and wait for 2 minutes.

If the check failed, click **Troubleshoot** and rectify the fault by referring to the documentation.

Step 5 If "Federation enabled." is displayed in the upper right corner, view the new version in **Fleet Info**.

If "Failed to upgrade federation." is displayed in the upper right corner, **roll back the federation upgrade**.

----End

Rolling Back Federation Upgrade

If the federation fails to be upgraded, UCS can upgrade the federation again or roll back the federation to the original version.

CAUTION

- If the federation has been upgraded, the federation version cannot be rolled back.
- During the federation version rollback, you cannot add a cluster to or remove a cluster from the federation and perform any federation operations.
- If the fault persists, submit a service ticket for technical support.

Step 1 Click **Failed to upgrade federation.** to view the failure cause.

Step 2 Click **Try again** to upgrade the federation again. For details, see **Federation Upgrade**.

Step 3 Click **Roll Back** to roll back the federation to the original version. In the displayed dialog box, click **OK**.

----End

3.5 Workloads

3.5.1 Workload Creation

3.5.1.1 Deployments

The federation function of UCS allows you to manage Kubernetes clusters in different regions or clouds, deploy applications globally in a unified manner, and deploy different workloads, such as Deployments, StatefulSets, and DaemonSets, to clusters in a federation.

Deployments are a type of workloads that do not store any data or status while running. An example of this is Nginx. You can create a Deployment using the console or kubectl.

Creating a Deployment

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane, choose **Workloads**. On the displayed page, click the **Deployments** tab. Then, click **Create from Image**.

NOTE

To use an existing YAML file to create a Deployment, click **Create from YAML** in the upper right corner.

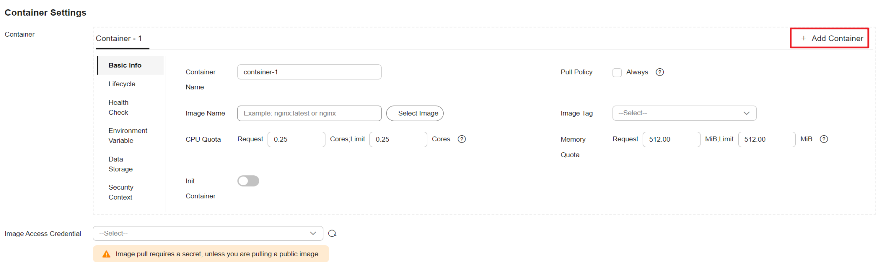
Step 4 Configure basic information about the workload.

- **Type**: Select **Deployment**.
- **Name**: name of the workload, which must be unique.
- **Namespace**: namespace that the workload belongs to. For details about how to create a namespace, see **Creating a Namespace**.
- **Description**: description of the workload.
- **Pods**: number of pods in each cluster of the multi-cluster workload. The default value is **2**. Each workload pod consists of the same containers. On UCS, you can set an auto scaling policy to dynamically adjust the number of workload pods based on the workload resource usage.

Step 5 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 3-10 Container settings



- **Basic Info**

Table 3-4 Basic information parameters

Parameter	Description
Container Name	Name the container.

Parameter	Description
Image Name	Click **Select Image** and select the image used by the container. – **My Images**: images in the Huawei Cloud image repository of the current region. If no image is available, click **Upload Image** to upload an image. – **Open Source Images**: official images in the open source image repository. – **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	– **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	– **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

- **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
- **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
- **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.

- **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Storage**.
- **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 6 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services and Ingresses**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Name**: name of the Service to be added. It is customizable and must be unique.
- **Type**
  - **ClusterIP**: The Service is only reachable from within the cluster.
  - **NodePort**: The Service can be accessed from any node in the cluster.
- **Affinity** (for node access only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.

- **Auto**: The system automatically assigns a port number.
- **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

Step 7 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade**: upgrade mode of the Deployment, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
  - **Replace upgrade**: Old pods are deleted before new pods are created. Services will be interrupted during a replace upgrade.
- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Configuring a Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.
- **Toleration**: When the node where the workload pods are located is unavailable for the specified amount of time, the pods will be rescheduled to other available nodes. By default, the toleration time window is 300s.

Step 8 Click **Next: Scheduling and Differentiation**. After selecting clusters to which the workload can be scheduled, configure the differentiated settings for the containers.

- **Scheduling Policy**
  - **Scheduling Mode**
    - **Weight**: Manually set the weight of each cluster. The number of pods in each cluster is allocated based on the configured weight.
    - **Auto balancing**: The workload is automatically deployed in the selected clusters based on available resources.
  - **Cluster**: Select clusters to which the workload can be scheduled. The number of clusters depends on your service requirements.
    - If you use cluster weighted scheduling, you need to manually set the weight of each cluster. If you set the weight of a cluster to a value other than **0**, the cluster is automatically selected as a cluster to which the workload can be scheduled. If you set it to **0**, the workload will not be scheduled to the cluster. Weights cannot be set for clusters in abnormal state.
    - If you use auto scaling, you can click a cluster to select it as a cluster to which the workload can be scheduled.
- **Differentiated Settings**

  When deploying a workload in multiple clusters, you can configure differentiated settings for these clusters. Click in the upper right corner of a target cluster to configure differentiated settings. The configured differentiated container settings take effect only for this cluster.

For parameter description, see **Container Settings**.

Step 9 After completing the settings, click **Create Workload**, then you can click **Back to Workload List** to view the created workload.

----End

3.5.1.2 StatefulSets

StatefulSets are a type of workloads that store data or status while running. Each pod in a StatefulSet is given a persistent identifier that remains even if the pod is migrated, destroyed, or restarted. StatefulSets do not support auto scaling and apply to scenarios that require persistent storage, such as etcd.

Creating a StatefulSet

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane, choose **Workloads**. On the displayed page, click the **StatefulSets** tab. Then, click **Create from Image** in the upper right corner.

NOTE

To use an existing YAML file to create a StatefulSet, click **Create from YAML** in the upper right corner.

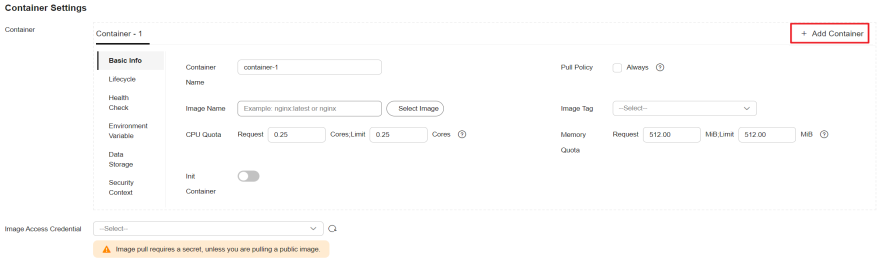
Step 4 Configure basic information about the workload.

- **Type**: Select **StatefulSet**.
- **Name**: name of the workload, which must be unique.
- **Namespace**: namespace that the workload belongs to. For details about how to create a namespace, see **Creating a Namespace**.
- **Description**: description of the workload.
- **Pods**: number of pods in each cluster of the multi-cluster workload. The default value is **2**. Each workload pod consists of the same containers. On UCS, you can set an auto scaling policy to dynamically adjust the number of workload pods based on the workload resource usage.

Step 5 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 3-11 Container settings



- **Basic Info**

Table 3-5 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. – **My Images**: images in the Huawei Cloud image repository of the current region. If no image is available, click **Upload Image** to upload an image. – **Open Source Images**: official images in the open source image repository. – **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	– **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	– **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

- **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.

- **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
- **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
- **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Storage**.
- **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 6 Configure the headless Service parameters for the workload.

StatefulSet pods discover each other through headless Services. No cluster IP is allocated for a headless Service, and the DNS records of all pods are returned during query. In this way, the IP addresses of all pods can be queried.

- **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**
  - **Port Name**: name of the container port. You are advised to enter a name that indicates the function of the port.
  - **Service Port**: port of the Service.
  - **Container Port**: listening port of the container.

Step 7 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services and Ingresses**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Name**: name of the Service to be added. It is customizable and must be unique.
- **Type**
  - **ClusterIP**: The Service is only reachable from within the cluster.
  - **NodePort**: The Service can be accessed from any node in the cluster.

- **Affinity** (for node access only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

Step 8 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade Policy**: upgrade mode of the StatefulSet, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
  - **Replace upgrade**: You need to delete old pods manually before new pods are created. Services will be interrupted during a replace upgrade.
- **Pod Management**
  - **OrderedReady**: The StatefulSet will launch, terminate, or scale pods sequentially. It will wait for the state of the pods to change to **Running** and **Ready** or completely terminated before it launches or terminates another pod.
  - **Parallel**: The StatefulSet will launch or terminate all pods in parallel. It will not wait for the state of the pods to change to Running and Ready or completely terminated before it launches or terminates another pod.
- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Configuring a Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.

Step 9 Click **Next** to configure the scheduling and differentiated settings for the selected clusters. After selecting clusters to which the workload can be scheduled, configure the differentiated settings for the containers.

- **Scheduling Policy**
  - **Scheduling Mode**
    - **Replication**: The workload will be deployed in all clusters selected below.
  - **Cluster**: Click to select clusters to which the workload can be scheduled. The number of clusters depends on your service requirements.
- **Differentiated Settings**

  When deploying a workload in multiple clusters, you can configure differentiated settings for these clusters. Click in the upper right corner of a target cluster to configure differentiated settings. The configured differentiated container settings take effect only for this cluster.

  For parameter description, see **Container Settings**.

Step 10 After completing the settings, click **Create Workload**.

----End

3.5.1.3 DaemonSets

A DaemonSet ensures that a pod runs on all (or some) nodes in a cluster. When a new node is added to the cluster, a pod will be automatically deployed on it. When a node is removed from the cluster, the pod on the node is also reclaimed. A typical use of a DaemonSet is running a log collection daemon on every node in the cluster. If a DaemonSet is deleted, all pods created by it will be deleted.

Creating a DaemonSet

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane, choose **Workloads**. On the displayed page, click the **DaemonSets** tab. Then, click **Create from Image** in the upper right corner.

> **NOTE**
>
> To use an existing YAML file to create a DaemonSet, click **Create from YAML** in the upper right corner.

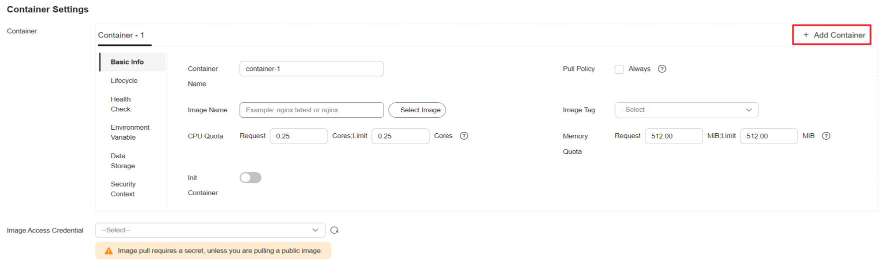
Step 4 Configure basic information about the workload.

- **Type**: Select **DaemonSet**.
- **Name**: name of the workload, which must be unique.
- **Namespace**: namespace that the workload belongs to. For details about how to create a namespace, see **Creating a Namespace**.
- **Description**: description of the workload.

Step 5 Configure the container settings for the workload.

Multiple containers can be configured in a pod. You can click **Add Container** on the right to configure multiple containers for the pod.

Figure 3-12 Container settings



- **Basic Info**

Table 3-6 Basic information parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container. – **My Images**: images in the Huawei Cloud image repository of the current region. If no image is available, click **Upload Image** to upload an image. – **Open Source Images**: official images in the open source image repository. – **Shared Images**: private images shared by another account. For details, see **Sharing Private Images**.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.
CPU Quota	– **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. – **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.

Parameter	Description
Memory Quota	– **Request**: minimum amount of memory required by a container. The default value is 512 MiB. – **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit** of CPU or memory, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

- **Lifecycle**: The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function. Currently, lifecycle callback functions, such as startup, post-start, and pre-stop are provided. For details, see **Setting Container Lifecycle Parameters**.
- **Health Check**: Set health check parameters to periodically check the health status of the container during container running. For details, see **Setting Health Check for a Container**.
- **Environment Variable**: Environment variables affect the way a running container will behave. Configuration items set by environment variables will not change if the pod lifecycle ends. For details, see **Setting Environment Variables**.
- **Data Storage**: Store container data using **Local Volumes** and **PersistentVolumeClaims (PVCs)**. You are advised to use PVCs to store workload pod data on a cloud volume. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored. For details about container storage, see **Storage**.
- **Security Context**: Set container permissions to protect the system and other containers from being affected. Enter a user ID and the container will run with the user permissions you specify.
- **Image Access Credential**: Select the credential for accessing the image repository. This credential is used only for accessing a private image repository. If the selected image is a public image, you do not need to select a secret. For details on how to create a secret, see **Creating a Secret**.

Step 6 (Optional) Click + in the **Service Settings** area to configure a Service for the workload.

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 **Services and Ingresses**.

You can also create a Service after creating a workload. For details, see **ClusterIP** and **NodePort**.

- **Name**: name of the Service to be added. It is customizable and must be unique.

- **Type**
  - **ClusterIP**: The Service is only reachable from within the cluster.
  - **NodePort**: The Service can be accessed from any node in the cluster.
- **Affinity** (for node access only)
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  - **Node Port** (for NodePort only): 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 **Auto**.
    - **Auto**: The system automatically assigns a port number.
    - **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.

Step 7 (Optional) Click **Expand** to set advanced settings for the workload.

- **Upgrade**: upgrade mode of the DaemonSet, including **Replace upgrade** and **Rolling upgrade**. For details, see **Configuring a Workload 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.
  - **Replace upgrade**: You need to delete old pods manually before new pods are created. Services will be interrupted during a replace upgrade.
- **Scheduling**: You can set affinity and anti-affinity to implement planned scheduling for pods. For details, see **Configuring a Scheduling Policy (Affinity/Anti-affinity)**.
- **Labels and Annotations**: You can click **Confirm** to add a label or annotation for the pod. The key of the new label or annotation cannot be the same as that of an existing one.

Step 8 Click **Next: Scheduling and Differentiation**. After selecting clusters to which the workload can be scheduled, configure the differentiated settings for the containers.

- **Scheduling Policy**

- **Scheduling Mode**
  - **Replication**: The workload will be deployed in all clusters selected below.
- **Cluster**: Click to select clusters to which the workload can be scheduled. The number of clusters depends on your service requirements.

- **Differentiated Settings**

  When deploying a workload in multiple clusters, you can configure differentiated settings for these clusters. Click in the upper right corner of a target cluster to configure differentiated settings. The configured differentiated container settings take effect only for this cluster.

  For parameter description, see **Container Settings**.

Step 9 Click **Create Workload**.

----End

3.5.2 Container Settings

3.5.2.1 Setting Basic Container Information

A workload is an abstract model of a group of pods. One pod can encapsulate one or more containers. You can click **Add Container** in the upper right corner to add multiple container images and set them separately.

Figure 3-13 Adding a container

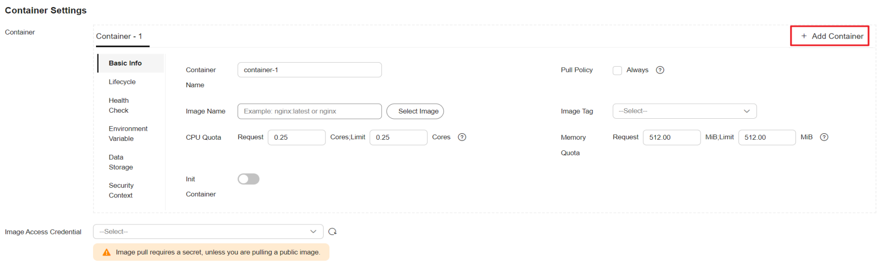


Table 3-7 Image parameters

Parameter	Description
Container Name	Name the container.
Image Name	Click **Select Image** and select the image used by the container.
Image Tag	Select the image tag to be deployed.
Pull Policy	Image update or pull policy. If you select **Always**, the image is pulled from the image repository each time. If you do not select **Always**, the existing image of the node is preferentially used. If the image does not exist in the node, it is pulled from the image repository.

Parameter	Description
CPU Quota	● **Request**: minimum number of CPU cores required by a container. The default value is 0.25 cores. ● **Limit**: maximum number of CPU cores available for a container. Do not leave **Limit** unspecified. Otherwise, intensive use of container resources will occur and your workload may exhibit unexpected behavior.
Memory Quota	● **Request**: minimum amount of memory required by a container. The default value is 512 MiB. ● **Limit**: maximum amount of memory available for a container. When memory usage exceeds the specified memory limit, the container will be terminated. For details about **Request** and **Limit**, see **Setting Container Specifications**.
Init Container	Select whether to use the container as an init container. An init container is a special container that runs before app containers in a pod. For details, see **Init Containers**.

3.5.2.2 Setting Container Specifications

Scenario

UCS allows you to set resource limits for added containers during workload creation. You can apply for and limit the CPU and memory quotas used by each pod in the workload.

Configuration Description

- CPU quotas:

Table 3-8 Description of CPU quotas

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

Recommended configuration

Actual available CPU of a node ≥ Sum of CPU limits of all containers of the current pod ≥ Sum of CPU requests of all containers on the current pod. You can view the actual available CPUs of a node by choosing **Clusters** in the navigation pane, clicking the name of the target cluster, and choosing **Nodes** on the displayed page.

- Memory quotas:

Table 3-9 Description of memory quotas

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

Recommended configuration

Actual available memory of a node ≥ Sum of memory limits of all containers on the current pod ≥ Sum of memory requests of all containers on the current pod. You can view the actual available memory of a node by choosing **Clusters** in the navigation pane, clicking the name of the target cluster, and choosing **Nodes** on the displayed page.

NOTE

The allocatable resources are calculated based on the resource request value (**Request**), which indicates the upper limit of resources that can be requested by pods on this node, but does not indicate the actual available resources of the node. The calculation formula is as follows:

- Allocatable CPU = Total CPU – Requested CPU of all pods – Reserved CPU for other resources
- Allocatable memory = Total memory – Requested memory of all pods – Reserved memory for other resources

Example

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

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

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

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

3.5.2.3 Setting Container Lifecycle Parameters

Scenario

The lifecycle callback functions can be called in specific phases of the container. For example, if you want the container to perform a certain operation before stopping, set the corresponding function.

UCS provides the following lifecycle callback functions:

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

Startup Commands

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

A Docker image has metadata that stores image information. If lifecycle commands and arguments are not set, UCS runs the default commands and arguments, that is, Docker instructions **ENTRYPOINT** and **CMD**, provided during image creation.

If the commands and arguments used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image build. The rules are as follows:

Table 3-10 Commands and arguments used to run a container

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

Step 1 Log in to the UCS console and access the **Federation** page. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Enter a command and arguments on the **Startup Command** tab page.

Table 3-11 Container startup command

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

----End

Post-Start Processing

Step 1 Log in to the UCS console and access the **Federation** page. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Set the post-start processing parameters on the **Post-Start** tab page.

Table 3-12 Post-start processing parameters

Parameter	Description

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

Parameter	Description
HTTP request	Send an HTTP request for post-start processing. The related parameters are described as follows: ● **Path**: (optional) request URL. ● **Port**: (mandatory) request port. ● **Host**: (optional) requested host IP address. The default value is the IP address of the pod.

----End

Pre-Stop Processing

Step 1 Log in to the UCS console and access the **Federation** page. When creating a workload, configure container information and select **Lifecycle**.

Step 2 Set the pre-start processing parameters on the **Pre-Stop** tab page.

Table 3-13 Pre-stop processing parameters

Parameter	Description

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

HTTP request	Send an HTTP request for pre-stop processing. The related parameters are described as follows: ● **Path**: (optional) request URL. ● **Port**: (mandatory) request port. ● **Host**: (optional) requested host IP address. The default value is the IP address of the pod.

----End

YAML Example

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        command:
        - sleep 3600                    #Startup command
        imagePullPolicy: Always
        lifecycle:
          postStart:
            exec:
              command:
              - /bin/bash
              - install.sh                  #Post-start command
          preStop:
            exec:
              command:
              - /bin/bash
              - uninstall.sh                 #Pre-stop command
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

3.5.2.4 Setting Health Check for a Container

Scenarios

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

Kubernetes provides the following health check probes:

- **Liveness probe** (livenessProbe): checks whether a container is still alive. It is similar to the **ps** command that checks whether a process exists. If the liveness check of a container fails, the cluster restarts the container. If the liveness check is successful, no operation is executed.
- **Readiness probe** (readinessProbe): checks whether a container is ready to process user requests. Upon that the container is detected unready, service traffic will not be directed to the container. It may take a long time for some

applications to start up before they can provide services. This is because that they need to load disk data or rely on startup of an external module. In this case, the application process is running, but the application cannot provide services. To address this issue, this health check probe is used. If the container readiness check fails, the cluster masks all requests sent to the container. If the container readiness check is successful, the container can be accessed.

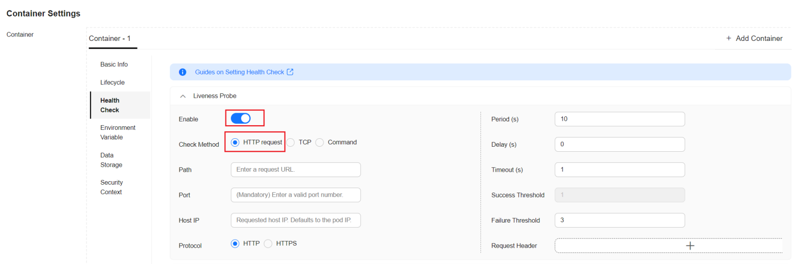
Check Methods

- **HTTP request**

  This health check mode can be used for containers that provide HTTP/HTTPS services. The cluster periodically initiates an HTTP/HTTPS GET request to such containers. If the return code of the HTTP/HTTPS response is within 200–399, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port and an HTTP/HTTPS request path.

  For example, for a container that provides HTTP services, the HTTP check path is **/health-check**, the port is 80, and the host address is optional (which defaults to the container IP address). Here, 172.16.0.186 is used as an example, and we can get such a request: GET http://172.16.0.186:80/health-check. The cluster periodically initiates this request to the container.

  Figure 3-14 HTTP request-based check

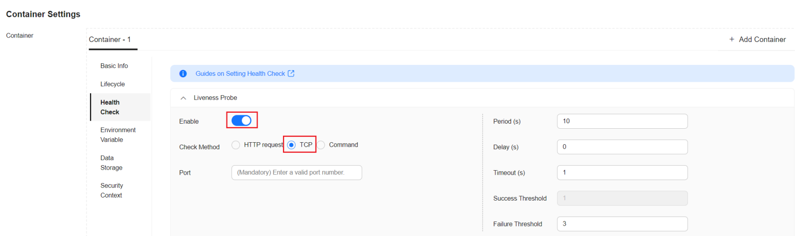


- **TCP**

  For a container that provides TCP communication services, the cluster periodically establishes a TCP connection to the container. If the connection is successful, the probe is successful. Otherwise, the probe fails. In this health check mode, you must specify a container listening port.

  For example, if you have a Nginx container with service port 80, after you specify TCP port 80 for container listening, the cluster will periodically initiate a TCP connection to port 80 of the container. If the connection is successful, the probe is successful. Otherwise, the probe fails.

  Figure 3-15 TCP port-based check



- **Command**

CLI is an efficient tool for health check. When using the CLI, you must specify an executable command in a container. The cluster periodically runs the command in the container. If the command output is 0, the health check is successful. Otherwise, the health check fails.

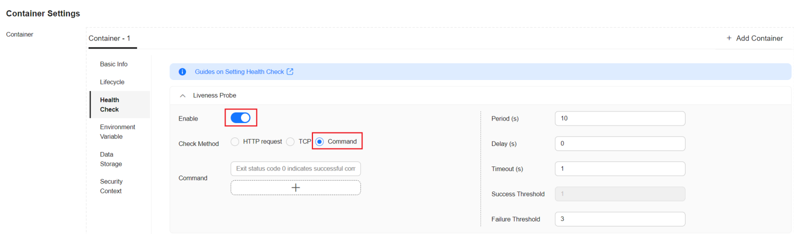
The CLI mode can be used to replace the HTTP request-based and TCP port-based health check.

- For a TCP port, you can use a script to connect to a container port. If the connection is successful, the script returns **0**. Otherwise, the script returns **–1**.
- For an HTTP request, you can use a script to run the **wget** command for a container.

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

  Check the return code of the response. If the return code is within 200–399, the script returns **0**. Otherwise, the script returns **–1**.

Figure 3-16 Command-based check



NOTICE

- Put the program to be executed in the container image so that the program can be executed.
- If the command to be executed is a shell script, do not directly specify the script as the command, but add a script parser. For example, if the script is **/data/scripts/health_check.sh**, the program is **sh/data/scripts/health_check.sh**. The reason is that the cluster is not in the terminal environment when executing programs in a container.

Common Parameters

Table 3-14 Common parameter description

Parameter	Description
Period (periodSeconds)	Probe detection period, in seconds. For example, if this parameter is set to **30**, the detection is performed every 30 seconds.

Parameter	Description
Delay (initialDelaySeconds)	Check delay time in seconds. Set this parameter according to the normal startup time of services. For example, if this parameter is set to 30, the health check will be started 30 seconds after the container is started. The time is reserved for containerized services to start.
Timeout (timeoutSeconds)	Timeout duration. Unit: second. For example, if this parameter is set to **10**, the timeout wait time for performing a health check is 10s. If the wait time elapses, the health check is regarded as a failure. If the parameter is left blank or set to **0**, the default timeout time is 1s.
Success Threshold (successThreshold)	Minimum consecutive successes for the probe to be considered successful after having failed. The default value is **1**, which is also the minimum value. The value of this parameter is fixed to **1** in **Liveness Probe**.
Failure Threshold (failureThreshold)	Number of retry times when the detection fails. Giving up in case of liveness probe means to restart the container. In case of readiness probe the pod will be marked **Unready**. The default value is **3**, and the minimum value is **1**.

YAML Example

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      exec:
        command:
          - cat
          - /tmp/healthy
```

```
  initialDelaySeconds: 5
  periodSeconds: 5
```

3.5.2.5 Setting Environment Variables

Scenario

An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after workloads are deployed, increasing flexibility in workload configuration.

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

NOTICE

After a container is started, do not modify configurations in the container. If configurations in the container are modified (for example, passwords, certificates, and environment variables of a containerized application are added to the container), the configurations will be lost after the container restarts and container services will become abnormal. An example scenario of container restart is pod rescheduling due to node anomalies.

Configurations must be imported to a container as arguments. Otherwise, configurations will be lost after the container restarts.

Environment variables can be set in the following modes:

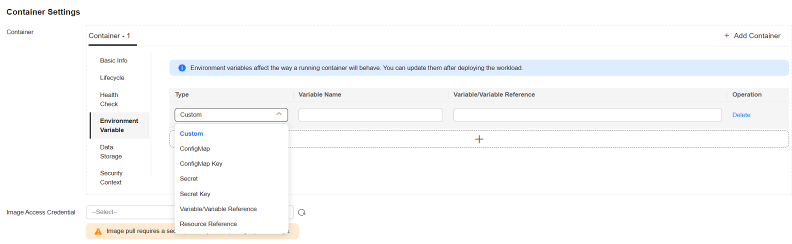
- **Custom**
- **ConfigMap**: Import all keys in a ConfigMap as environment variables.
- **ConfigMap Key**: Import a key in a ConfigMap as the value of an environment variable. For example, if you import **configmap_value** of **configmap_key** in ConfigMap **configmap-example** as the value of environment variable **key1**, an environment variable named **key1** with its value **is configmap_value** exists in the container.
- **Secret**: Import all keys in a secret as environment variables.
- **Secret Key**: Import the value of a key in a secret as the value of an environment variable. For example, if you import **secret_value** of **secret_key** in secret **secret-example** as the value of environment variable **key2**, an environment variable named **key2** with its value **secret_value** exists in the container.
- **Variable/Variable Reference**: Use the field defined by a pod as the value of the environment variable, for example, the pod name.
- **Resource Reference**: Use the field defined by a container as the value of the environment variable, for example, the CPU limit of the container.

Environment Variables

Step 1 Log in to the UCS console and access the **Federation** page. When creating a workload, configure container information and select **Environment Variable**.

Step 2 Configure environment variables.

Figure 3-17 Configuring environment variables



----End

YAML Example

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

```
                resource: limits.cpu
                divisor: 1
        envFrom:
          - configMapRef:                 # Added from ConfigMap.
              name: configmap-example
          - secretRef:                    # Added from secret.
              name: secret-example
      imagePullSecrets:
        - name: default-secret
```

Viewing Environment Variables

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

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

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

The environment variables in the pod are as follows:

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

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

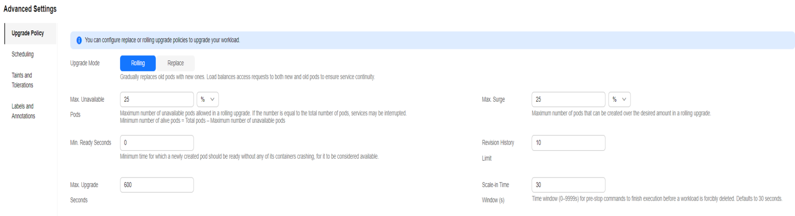
3.5.2.6 Configuring a Workload Upgrade Policy

In actual applications, upgrade is a common operation. A Deployment, StatefulSet, or DaemonSet can easily support application upgrade.

You can set different upgrade policies:

- **Rolling** (RollingUpdate): New pods are created gradually and then old pods are deleted. This is the default policy.
- **Replace** (Recreate): The current pods are deleted and then new pods are created.

Figure 3-18 Configuring a workload upgrade policy



Upgrade Parameters

- **Max. Surge** (maxSurge)

  Specifies the maximum number of pods that can exist over **spec.replicas**. The default value is 25%. For example, if **spec.replicas** is set to **4**, no more than 5 pods can exist during the upgrade process, that is, the upgrade step is 1. The absolute number is calculated from the percentage by rounding up. The value can also be set to an absolute number.

  This parameter is available only when **Rolling** is selected for Deployments.

- **Max. Unavailable Pods** (maxUnavailable)

  Specifies the maximum number of pods that can be unavailable during the upgrade process. The default value is 25%. For example, if **spec.replicas** is set to **4**, at least 3 pods exist during the upgrade process. The deletion step is 1. The value can also be set to an absolute number.

  This parameter is available only when **Rolling** is selected for Deployments or DaemonSets.

- **Min. Ready Seconds** (minReadySeconds)

  A pod is considered available only when the minimum readiness time is exceeded without any of its containers crashing. The default value is **0** (the pod is considered available immediately after it is ready).

  This parameter is available only to Deployments and DaemonSets.

- **Revision History Limit** (revisionHistoryLimit)

  Specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of **kubectl get rs**. The configuration of each workload revision is stored in its ReplicaSets. Therefore, once the old ReplicaSet is deleted, you lose the ability to roll back to that revision of the workload. By default, 10 old ReplicaSets will be kept, but the ideal value depends on the frequency and stability of the new workloads.

- **Max. Upgrade Duration** (progressDeadlineSeconds)

  Specifies the number of seconds that the system waits for a Deployment to make progress before reporting a Deployment progress failure. It is surfaced as a condition with Type=Progressing, Status=False, and Reason=ProgressDeadlineExceeded in the status of the resource. The Deployment controller will keep retrying the Deployment. In the future, once automatic rollback will be implemented, the Deployment controller will roll back a Deployment as soon as it observes such a condition.

  If this parameter is specified, the value of this parameter must be greater than that of **.spec.minReadySeconds**.

  This parameter is only available for Deployments.

- **Scale-In Time Window** (terminationGracePeriodSeconds)

  Graceful deletion time. The default value is 30 seconds. When a pod is deleted, a SIGTERM signal is sent and the system waits for the applications in the container to terminate. If the application is not terminated within the time specified by **terminationGracePeriodSeconds**, a SIGKILL signal is sent to forcibly terminate the pod.

Upgrade Example

The Deployment can be upgraded in a declarative mode. That is, you only need to modify the YAML definition of the Deployment. For example, you can run the **kubectl edit** command to change the Deployment image to **nginx:alpine**. After the modification, query the ReplicaSet and pod. The query result shows that a new ReplicaSet is created and the pod is re-created.

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME               DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd   2         2         2         1m
nginx-7f98958cdf   0         0         0         48m

$ kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

The Deployment can use the **maxSurge** and **maxUnavailable** parameters to control the proportion of pods to be re-created during the upgrade, which is useful in many scenarios. The configuration is as follows:

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

In the preceding example, the value of **spec.replicas** is **2**. If both **maxSurge** and **maxUnavailable** are the default value 25%, **maxSurge** allows a maximum of three pods to exist (2 x 1.25 = 2.5, rounded up to 3), and **maxUnavailable** does not allow a maximum of two pods to be unavailable (2 x 0.75 = 1.5, rounded up to 2). That is, during the upgrade process, there will always be two pods running. Each time a new pod is created, an old pod is deleted, until all pods are new.

Rollback

Rollback is to roll an application back to the source version when a fault occurs during the upgrade. A Deployment can be easily rolled back to the source version.

For example, if the upgraded image is faulty, you can run the **kubectl rollout undo** command to roll back the Deployment.

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

A Deployment can be easily rolled back because it uses a ReplicaSet to control a pod. After the upgrade, the previous ReplicaSet still exists. The Deployment is rolled back by using the previous ReplicaSet to re-create the pod. The number of

ReplicaSets stored in a Deployment can be restricted by the **revisionHistoryLimit** parameter. The default value is **10**.

3.5.2.7 Configuring a Scheduling Policy (Affinity/Anti-affinity)

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

Configuring Scheduling Policies

Step 1 Log in to the UCS console and go to the **Federation** page.

Step 2 When creating a workload, click **Scheduling** in the **Advanced Settings** area.

Table 3-15 Node affinity settings

Parameter	Description
Required	A hard rule that must be met for scheduling. It corresponds to **requiredDuringSchedulingIgnoredDuringExecution** in Kubernetes. You can add multiple required rules, and scheduling will be performed if any of them is met.
Preferred	A soft rule specifying preferences that the scheduler will try to enforce but will not guarantee. It corresponds to **preferredDuringSchedulingIgnoredDuringExecution** in Kubernetes. You can add multiple preferred rules, and scheduling will be performed if any or none of them is met.

Step 3 Under **Node affinity**, **Workload affinity**, and **Workload anti-affinity**, click + to add scheduling policies.

Table 3-16 Scheduling policy configuration

Parameter	Description
Label Key	Node label. You can use the default label or customize a label.

Parameter	Description
Operator	The following relations are supported: **In**, **NotIn**, **Exists**, **DoesNotExist**, **Gt**, and **Lt** ● **In**: A label exists in the label list. ● **NotIn**: A label does not exist in the label list. ● **Exists**: A specific label exists. ● **DoesNotExist**: A specific label does not exist. ● **Gt**: The label value is greater than a specified value (string comparison). ● **Lt**: The label value is less than a specified value (string comparison).
Label Value	Label value.
Namespace	This parameter is available only in a workload affinity or anti-affinity scheduling policy. Namespace for which the scheduling policy takes effect.
Topology Key	This parameter is available only in a workload affinity or anti-affinity scheduling policy. Select the scope specified by **topologyKey** and then select the content defined by the policy.
Weight	This parameter can be set only in a **Preferred** scheduling policy.

----End

Node Affinity (nodeAffinity)

In the pod template, you can configure **nodeSelector** to create a pod on a node with a specified label. The following example shows how to use a nodeSelector to deploy pods only on the nodes with the **gpu=true** label.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:                # Node selection. A pod is deployed only on the node with the gpu=true label.
    gpu: true
...
```

You can also use node affinity to do so.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
```



```
template:
  metadata:
    labels:
      app:  gpu
  spec:
    containers:
    - image:  nginx:alpine
      name:  gpu
      resources:
        requests:
          cpu: 100m
          memory: 200Mi
        limits:
          cpu: 100m
          memory: 200Mi
    imagePullSecrets:
    - name: default-secret
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

A node affinity rule contains more lines, but it is more expressive.

requiredDuringSchedulingIgnoredDuringExecution seems to be complex, but it can be easily understood as a combination of two parts.

- **requiredDuringScheduling** indicates that pods can be scheduled to the node only when all the defined selector rules are met.
- **IgnoredDuringExecution** means that if the node labels change after Kubernetes schedules the pod, the pod continues to run.

In addition, the operator **In** indicates that the label value must fall in the range specified by **values**. Other available operator values are as follows:

- **NotIn**: The label value is not in the specified list.
- **Exists**: A specific label exists.
- **DoesNotExist**: A specific label does not exist.
- **Gt**: The label value is greater than a specified value (string comparison).
- **Lt**: The label value is less than a specified value (string comparison).

Note that there is no such thing as nodeAntiAffinity because operators **NotIn** and **DoesNotExist** provide the same function.

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

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

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

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

$ kubectl get node -L gpu
```



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

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

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

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

Node Preference Rule

The preceding **requiredDuringSchedulingIgnoredDuringExecution** rule is a hard rule. The other type, or a soft rule, is **preferredDuringSchedulingIgnoredDuringExecution**, which specifies which nodes are preferred during scheduling.

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

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
```

```
    imagePullSecrets:
    - name: default-secret
    affinity:
      nodeAffinity:
        preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 80
          preference:
            matchExpressions:
            - key: DISK
              operator: In
              values:
              - SSD
        - weight: 20
          preference:
            matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

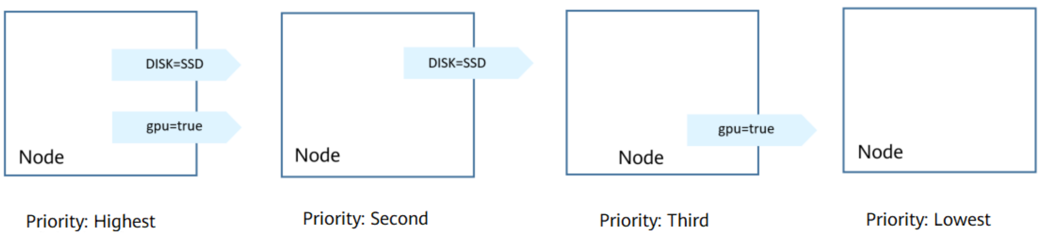
After the deployment, you can find that five pods are deployed on the **192.168.0.212** node, and two pods are deployed on the **192.168.0.100** node.

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

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

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

Figure 3-19 Scheduling priority



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

Workload Affinity (podAffinity)

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

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

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```

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

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

$ kubectl get po -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67   192.168.0.100
```

```
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70   192.168.0.100
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71   192.168.0.100
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72   192.168.0.100
```

The **topologyKey** field specifies the selection range. The scheduler selects nodes within the range based on the affinity rule defined. The effect of **topologyKey** is not fully demonstrated in the preceding example because all the nodes have the **kubernetes.io/hostname** label, that is, all the nodes are within the range.

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

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

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

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

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

Define **topologyKey** in the **podAffinity** section as **prefer**.

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

The scheduler recognizes the nodes with the **prefer** label, that is, **192.168.0.97** and **192.168.0.94**, and then finds the pods with the **app=backend** label. In this way, all frontend pods are deployed onto **192.168.0.97**.

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

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

Workload Anti-Affinity (podAntiAffinity)

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

The following example defines an inter-pod anti-affinity rule, which specifies that pods must not be scheduled to nodes that already have pods with the

app=frontend label, that is, to deploy the pods of the frontend to different nodes with each node has only one replica.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - frontend
```

Deploy the frontend and query the deployment results. You can find that each node has only one frontend pod and one pod of the Deployment is **Pending**. This is because when the scheduler is deploying the fifth pod, all nodes already have one pod with the **app=frontend** label on them. There is no available node. Therefore, the fifth pod will remain in the **Pending** status.

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

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

3.5.2.8 Configuring Scheduling and Differentiation

Scheduling Policies

The current page supports two policies: cluster weighted scheduling and automatic balancing.

Configuring a Scheduling Policy on the Console

Step 1 Log in to the UCS console.

Step 2 When creating a workload, click **Next: Scheduling and Differentiation**.

Step 3 Add a scheduling policy.

Table 3-17 Scheduling policies

Policy	Description
Cluster weighted scheduling	You need to select a cluster and configure the scheduling weight. Pods will be scheduled based on the cluster weight ratio.
Auto balancing	The system automatically selects a cluster to distribute pods based on the remaining resources. No additional configuration is required.

----End

Tolerance Policies

A tolerance policy allows the scheduler to schedule pods to clusters with corresponding taints. This policy must be used together with cluster taints.

Using the Default Tolerance Policy

When you create a workload, UCS configures a default tolerance policy for your workload. The default tolerance policy adds taints listed in **Table 3-18** to a faulty cluster. If the tolerance duration is exceeded, all pods in the cluster will be automatically evicted.

CAUTION

After all pods in the faulty cluster are evicted, UCS does not migrate the pods back to the cluster based on the original scheduling policy after the cluster recovers. If you want to continue with the original scheduling policy configuration, you can **reschedule** the workload.

Table 3-18 Taints for faulty clusters

Taint Key	Tolerance Policy
cluster.karmada.io/not-ready	When the cluster is not ready, this taint is automatically added. If the tolerance duration is exceeded, all pods in the cluster will be automatically evicted.

Taint Key	Tolerance Policy
cluster.karmada.io/unreachable	When the cluster is unavailable, this taint is automatically added. If the tolerance duration is exceeded, all pods in the cluster will be automatically evicted.

Configuring a Tolerance Policy on the Console

Step 1 Log in to the UCS console.

Step 2 When creating a workload, click **Next: Scheduling and Differentiation**.

Step 3 Add a tolerance policy.

Parameter	Description
Taint Key	Taint key of the cluster.
Operator	• **Equal**: matches the nodes with the specified taint key (mandatory) and value. If the taint value is left blank, all taints with the key the same as the specified taint key will be matched. • **Exists**: matches the nodes with the specified taint key. In this case, the taint value cannot be specified. If the taint key is left blank, all taints will be tolerated.
Taint Value	• If the value of **Operator** is **Exists**, the value attribute can be omitted. • If the value of **Operator** is **Equal**, the relationship between the key and value is **Equal**. • If **Operator** is not specified, the default value is **Equal**.
Taint Policy	• **All**: All taint policies are matched. • **NoSchedule**: Only the **NoSchedule** taint is matched. • **NoExecute**: Only the **NoExecute** taint is matched.
Toleration Time Window	**tolerationSeconds**, which is configurable only when **Taint Policy** is set to **NoExecute**. Within the tolerance time window, pods still run on the node with taints. After the time expires, the pods will be evicted. If the tolerance time window is not configured, the pods will never be evicted.

----End

3.5.3 Managing a Workload

Scenarios

After a workload is created, you can view its details, upgrade it, edit YAML, redeploy it, reschedule it, and delete it.

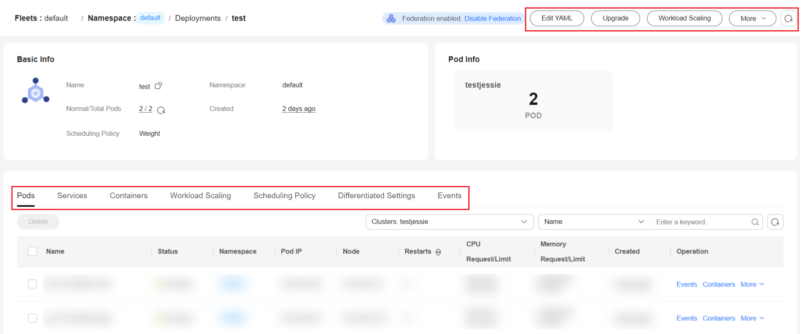
Table 3-19 Workload management

Operation	Description
Viewing Workload Details	You can view the basic information, events, and status of pods and workloads, and modify workload settings.
Editing a YAML File	You can modify and download the YAML file of a workload online. YAML files of common jobs can only be viewed, copied, and downloaded.
Upgrading a Workload	You can quickly upgrade a workload by replacing its image or image version without interrupting services.
Redeploying a Workload	You can redeploy a workload. After the workload is redeployed, all pods in the workload will be restarted. Only Deployments can be redeployed.
Rescheduling a Workload	You can reschedule a workload. After being rescheduled, workloads are scheduled based on existing scheduling policies. Only Deployments can be redeployed.
Deleting a Workload	If a workload is no longer used, you can delete it. Deleted workloads or tasks cannot be restored.

Viewing Workload Details

Click the name of a created workload to go to its details page. On this page, you can view the basic information, events, and status of pods and workloads, and modify workload settings.

Figure 3-20 Workload details page



Editing a YAML File

You can modify and download the YAML files of Deployments, StatefulSets, DaemonSets, CronJobs, and pods. YAML files of jobs can only be viewed, copied, and downloaded. This section uses a Deployment as an example to describe how to edit the YAML file.

Step 1 Log in to the UCS console, access an existing fleet, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab, locate your Deployment, and click **Edit YAML** in the **Operation** column. In the dialog box displayed, modify the YAML file.

Step 3 Click **OK**.

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

----End

Upgrading a Workload

You can quickly upgrade a workload on the UCS console. This section uses a Deployment as an example to describe how to upgrade a workload.

Step 1 Log in to the UCS console, access an existing fleet, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab, locate your workload, and click **Upgrade** in the **Operation** column.

Step 3 Upgrade the workload based on service requirements. The method for setting parameters is the same as that for creating a workload.

Step 4 After the update is complete, click **Upgrade Workload**, manually confirm the YAML file, and submit the upgrade.

----End

Redeploying a Workload (Available Only for Deployments)

After you redeploy a workload, all pods in the workload will be restarted. This section uses a Deployment as an example to describe how to redeploy a workload.

Step 1 Log in to the UCS console, access an existing fleet, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab, locate your workload, and choose **More** > **Redeploy** in the **Operation** column.

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

----End

Rescheduling a Workload (Available Only for Deployments)

A workload can be rescheduled to a cluster based on the scheduling policy. This section uses a Deployment as an example to describe how to reschedule a workload.

Step 1 Log in to the UCS console, access an existing fleet, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab, locate your workload, and choose **More** > **Reschedule** in the **Operation** column.

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

----End

NOTE

A workload can be rescheduled based on the original scheduling policy after a faulty cluster is recovered.

Deleting a Workload

You can delete a workload or job that is no longer needed. Deleted workloads or jobs cannot be recovered. This section uses a Deployment as an example to describe how to delete a workload.

Step 1 Log in to the UCS console, access an existing fleet, and choose **Workloads** in the navigation pane.

Step 2 Click the **Deployments** tab, locate your workload, and choose **More** > **Delete** in the **Operation** column.

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

----End

3.6 ConfigMaps and Secrets

3.6.1 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.

NOTE

- After a ConfigMap is created on the UCS console, it is in the undeployed state by default. You need to mount the ConfigMap when creating or updating a workload. For details, see **ConfigMap**.
- After a ConfigMap is mounted to a workload, a ConfigMap with the same name is created in each cluster to which the workload belongs.

Creating a ConfigMap

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **ConfigMaps and Secrets** in the navigation pane and click the **ConfigMaps** tab.

Step 4 Select the namespace for which you want to create a ConfigMap and click **Create ConfigMap** in the upper right corner.

Step 5 Set the parameters listed in **Table 3-20**.

Table 3-20 Parameters for creating a ConfigMap

Parameter	Description
Name	Name of a ConfigMap, which must be unique in a namespace.
Namespace	Namespace that the ConfigMap belongs to. The current namespace is used by default.
Description	Description of the ConfigMap.
Data	The workload configuration data can be used in a container or used to store the configuration data. Click + and enter the key and value. **Key** indicates the configuration name, and **Value** indicates the configuration content.
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. Enter the label key and value. 2. Click **Confirm**.

Step 6 Click **OK**.

----End

Using a ConfigMap

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

Related Operations

You can also perform operations described in **Table 3-21**.

Table 3-21 Related operations

Operation	Description
Creating a ConfigMap from a YAML file	Click **Create from YAML** in the upper right corner to create a ConfigMap from an existing YAML file.
Viewing details	Click the ConfigMap name to view its details.
Editing a YAML file	Click **Edit YAML** in the row where the target ConfigMap resides to edit its YAML file.
Updating a ConfigMap	1. Choose **More** > **Update** in the **Operation** column of the target ConfigMap. 2. Modify the ConfigMap information according to **Table 3-20**. 3. Click **OK** to submit the modified information.
Deleting a ConfigMap	Choose **More** > **Delete** in the row where the target ConfigMap resides, and click **Yes**.
Deleting ConfigMaps in batches	1. Select the ConfigMaps to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.6.2 Secrets

A secret is a type of resource that holds sensitive data, such as authentication and key information. Its content is user-defined.

NOTE

- After a secret is created on the UCS console, it is in the undeployed state by default. You need to mount the secret when creating or updating a workload. For details, see **Secret**.
- After a secret is mounted to a workload, a secret with the same name is created in each cluster to which the workload belongs.

Creating a Secret

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **ConfigMaps and Secrets** in the navigation pane and click the **Secrets** tab.

Step 4 Select the namespace for which you want to create a secret and click **Create Secret** in the upper right corner.

Step 5 Set the parameters listed in **Table 3-22**.

Table 3-22 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. The current namespace is used by default.
Description	Description of the secret.
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. If you select this secret type, enter the image repository address. ● IngressTLS: a secret that stores the certificate required by an Ingress. If you select this secret type, upload the certificate file and private key file. ● 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 select **Auto Base64-encoded** to Base64-encode the entered value. For details about manual Base64 encoding, see **Base64 Encoding**. ● If the secret type is **kubernetes.io/dockerconfigjson**, enter the username 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 **Confirm**. 2. Enter the key and value.

Step 6 Click **OK**.

The new secret is displayed in the secret list.

----End

Using a Secret

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

Base64 Encoding

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

```
echo -n "Content to be encoded" | base64
```

Related Operations

You can also perform operations described in **Table 3-23**.

Table 3-23 Related operations

Operation	Description
Creating a secret from a YAML file	Click **Create from YAML** in the upper right corner to create a secret from an existing YAML file.
Viewing details	Click the secret name to view its details.
Editing a YAML file	Click **Edit YAML** in the row where the target secret resides to edit 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 3-22**. 3. Click **OK** to submit the modified information.
Deleting a secret	Choose **More** > **Delete** in the row where the target secret resides, and click **Yes**.
Deleting secrets in batches	1. Select the secrets to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.7 Services and Ingresses

3.7.1 Overview

UCS clusters allow workload access in different scenarios via Services and Ingresses.

NOTICE

- After a Service or Ingress is created on the UCS console, a Service or Ingress with the same name will be created in the cluster that each associated workload belongs to.
- You can modify or delete the Services and Ingresses automatically created by UCS in the cluster console. However, if the Service or Ingress settings on the UCS console are not modified accordingly, the modified or deleted Services or Ingresses will be re-created by UCS. Therefore, you are advised to change the settings on the UCS console, not the cluster console.
- When an exception occurs in your cluster, Services in the cluster will be migrated to a healthy cluster. When your cluster recovers, you need to manually modify the Service template to deploy the Services again.

- **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 from outside the cluster. A NodePort Service is exposed on each node's IP address at a static port. If a node in the cluster is bound to an EIP, workloads on the node can be accessed from public networks by requesting <*EIP*>:<*NodePort*>.

- **LoadBalancer**

  A workload can be accessed from a public network through a load balancer. LoadBalancer provides higher reliability than EIP-based NodePort because the former needs no EIP. 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**

  Load balancers are used for Ingresses. Compared with load balancers at Layer 4, load balancers at Layer 7 support URI configurations and distribute traffic to the corresponding Services 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**.

3.7.2 Services

3.7.2.1 ClusterIP

A ClusterIP Service allows workloads in the same cluster to use their **cluster-internal domain names** to access each other. 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**.

Creating a Service

You can create a Service in either of the following ways:

- Create one when creating a workload. For details, see **During Workload Creation**.
- Create one after creating a workload. For details, see **After Workload Creation**.

During Workload Creation

The procedure of creating a Service is the same for different types of workloads, such as Deployments, StatefulSets, and DaemonSets.

Step 1 In the **Service Settings** step of **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**, click + to configure the Service.

- **Name**: name of the Service to be created.
- **Type**: Select **ClusterIP**.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).

Figure 3-21 Workload Service settings

Step 2 Click **OK**.

Step 3 Click **Next: Set Scheduling and Differentiation** to configure the scheduling and differentiated settings for the selected clusters. After completing the settings, click **Create Workload**.

Step 4 Obtain the access address.

1. In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.
2. Click the name of the added Service to go to its details page and obtain the access address of the deployment cluster.

----End

After Workload Creation

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

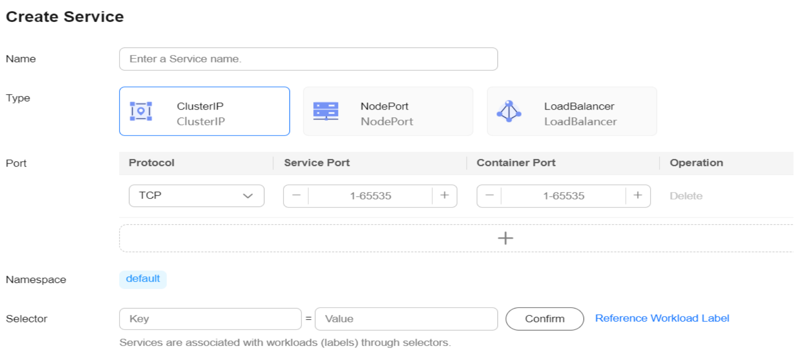
Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.

Step 4 Select the namespace that the Service will belong to and click **Create Service** in the upper right corner. For details about how to create a namespace, see **Creating a Namespace**.

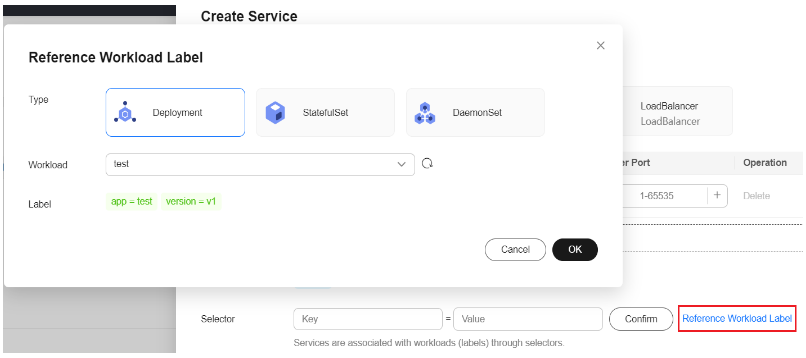
Step 5 Set access parameters.

Figure 3-22 Creating a Service



- **Name**: Can be the same as the workload name.
- **Type**: Select **ClusterIP**.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
- **Namespace**: namespace to which the Service belongs.
- **Selector**: Services are associated with workloads (labels) through selectors. Click **Reference Workload Label** to reference the labels of an existing workload.
  - **Type**: Select the desired workload type.
  - **Workload**: Select an existing workload. If your workload is not displayed in the list, click to refresh it.
  - **Label**: After a workload is selected, its labels are displayed and cannot be modified.

Figure 3-23 Referencing a workload label



Step 6 Click **OK**. After the Service is created, you can view it in the list on the **Services** tab page.

----End

Related Operations

You can also perform operations described in **Table 3-24**.

Table 3-24 Related operations

Operation	Description
Creating a Service from a YAML file	Click **Create from YAML** in the upper right corner to create a Service from an existing YAML file.
Viewing details	1. Select the namespace to which the Service belongs. 2. (Optional) Search for a Service by its name. 3. Click the Service name to view its details, including the basic information and cluster deployment information. 4. On the **Service Details** page, click **View YAML** in the **Cluster** area to view or download YAML files of Service instances deployed in each cluster.
Editing a YAML file	Click **Edit YAML** in the row where the target Service resides to view and edit the YAML file of the Service.
Updating a Service	1. Choose **More** > **Update** in the row where the target Service resides. 2. Modify the information by referring to **Step 5**. 3. Click **OK** to submit the modified information.
Deleting a Service	Choose **More** > **Delete** in the row where the target Service resides, and click **Yes**.

Operation	Description
Deleting Services in batches	1. Select the Services to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.7.2.2 NodePort

A NodePort Service is exposed on a node at a static port, allowing access from outside the cluster to the workloads on the node. A ClusterIP Service, to which the NodePort Service routes, is automatically created, and it transfers access requests to the backing containers. If a node in the cluster is bound to an EIP, you can also request <*EIP*>:<*NodePort*> to access the workloads from public networks.

Creating a Service

You can create a Service in either of the following ways:

- Create one when creating a workload. For details, see **During Workload Creation**.
- Create one after creating a workload. For details, see **After Workload Creation**.

During Workload Creation

The procedure of creating a Service is the same for different types of workloads, such as Deployments, StatefulSets, and DaemonSets.

Step 1 In the **Service Settings** step of **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**, click + to configure the Service.

- **Name**: name of the Service to be created.
- **Type**: Select **NodePort**.
- **Affinity**
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. No performance loss due to hops, and source IP addresses can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service 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.

– **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
– **Node 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 **Auto**.
  - **Auto**: The system automatically assigns a port number.
  - **Custom**: 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: Set Scheduling and Differentiation** to configure the scheduling and differentiated settings for the selected clusters. After completing the settings, click **Create Workload**.

Step 4 Obtain the access address.

1. In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.
2. Click the name of the added Service to go to its details page and obtain the access address of the deployment cluster. If a node in the cluster is bound to an EIP, you can access the backend workload through the EIP and node port of the node where the workload is deployed.

----End

After Workload Creation

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

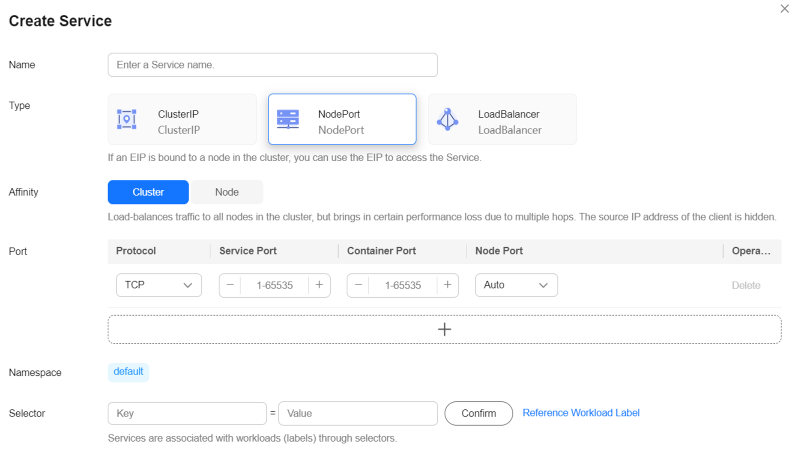
Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.

Step 4 Select the namespace that the Service will belong to and click **Create Service** in the upper right corner. For details about how to create a namespace, see **Creating a Namespace**.

Step 5 Configure access parameters.

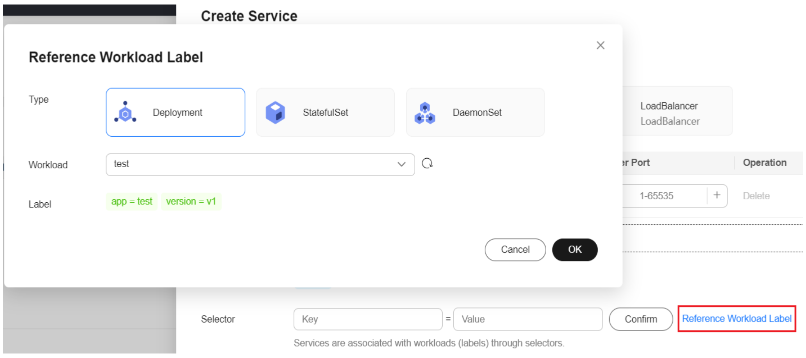
Figure 3-24 Creating a Service



- **Name**: Can be the same as the workload name.
- **Type**: Select **NodePort**.
- **Affinity**
  – **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  – **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. No performance loss due to hops, and source IP addresses can be obtained.
- **Port**
  – **Protocol**: Select **TCP** or **UDP**.
  – **Service 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.
  – **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
  – **Node 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 **Auto**.
    ▪ **Auto**: The system automatically assigns a port number.
    ▪ **Custom**: Specify a fixed node port. The port number range is 30000–32767. Ensure that the port is unique in a cluster.
- **Namespace**: namespace to which the Service belongs.
- **Selector**: Services are associated with workloads (labels) through selectors. Click **Reference Workload Label** to reference the labels of an existing workload.
  – **Type**: Select the desired workload type.

– **Workload**: Select an existing workload. If your workload is not displayed in the list, click to refresh it.
– **Label**: After a workload is selected, its labels are displayed and cannot be modified.

Figure 3-25 Referencing a workload label



Step 6 Click **OK**. After the Service is created, you can view it in the list on the **Services** tab page.

Step 7 Obtain the access address.

1. In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.
2. Click the name of the added Service to go to its details page and obtain the access address of the deployment cluster. If a node in the cluster is bound to an EIP, you can access the backend workload through the EIP and node port of the node where the workload is deployed.

----End

Related Operations

You can also perform operations described in **Table 3-25**.

Table 3-25 Related operations

Operation	Description
Creating a Service from a YAML file	Click **Create from YAML** in the upper right corner to create a Service from an existing YAML file.

Operation	Description
Viewing details	1. Select the namespace to which the Service belongs. 2. (Optional) Search for a Service by its name. 3. Click the Service name to view its details, including the basic information and cluster deployment information. 4. On the **Service Details** page, click **View YAML** in the **Cluster** area to view or download YAML files of Service instances deployed in each cluster.
Editing a YAML file	Click **Edit YAML** in the row where the target Service resides to view and edit the YAML file of the Service.
Updating a Service	1. Choose **More** > **Update** in the row where the target Service resides. 2. Modify the information by referring to **Step 5**. 3. Click **OK** to submit the modified information.
Deleting a Service	Choose **More** > **Delete** in the row where the target Service resides, and click **Yes**.
Deleting Services in batches	1. Select the Services to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.7.2.3 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 **Workloads**.

Creating a Service

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

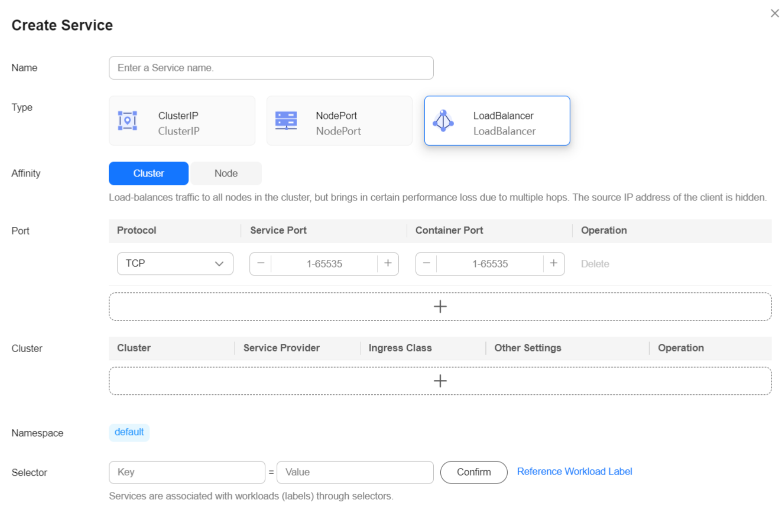
Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.

Step 4 Select the namespace that the Service will belong to and click **Create Service** in the upper right corner. For details about how to create a namespace, see **Creating a Namespace**.

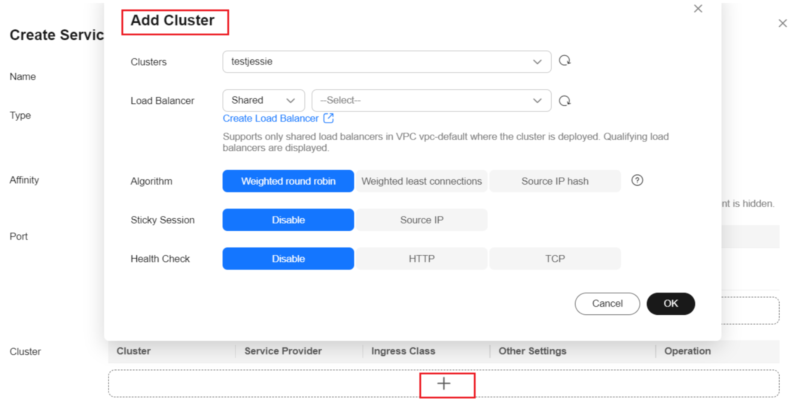
Step 5 Configure access parameters.

Figure 3-26 Creating a Service



- **Name**: name of the Service to be created.
- **Type**: Select **LoadBalancer**.
- **Affinity**
  - **Cluster-level**: The IP addresses and access ports of all nodes in a cluster can be used to access the workloads associated with the Service. However, performance loss is introduced due to hops, and source IP addresses cannot be obtained.
  - **Node-level**: Only the IP address and access port of the node where the workload is located can be used to access the workload associated with the Service. Service access will not cause performance loss due to route redirection, and the source IP address of the client can be obtained.
- **Port**
  - **Protocol**: Select **TCP** or **UDP**.
  - **Service Port**: Specify a port to map a container port to the load balancer. The port range is 1–65535. The port will be used when the application is accessed through the load balancer.
  - **Container Port**: Port on which the workload listens, defined in the container image. For example, the Nginx application listens on port 80 (container port).
- **Cluster**: Add a cluster where load balancers are to be deployed and complete differentiated load balancer settings.

Figure 3-27 Adding a cluster



– CCE cluster:

- **Load Balancer**: Only load balancers in the VPC where the cluster resides are supported.
- **Algorithm**

  Weighted round robin: Distributes requests to backend servers based on weights.

  Weighted least connections: Distributes requests to backend servers with the smallest ratio (current connections divided by weight).

  Source IP hash: Allocates requests from the client IP address to a fixed server, allowing the entire session to be processed by the same server.
- **Sticky Session**: This function is disabled by default. You can select **Source IP**. Listeners ensure session stickiness based on IP addresses. Requests from the same IP address will be routed to the same backend server.
- **Health Check**: This function is disabled by default. You can select either HTTP or TCP to enable health checks for your load balancer. For details about the parameters, see **Table 3-26**.

Table 3-26 Health check parameters

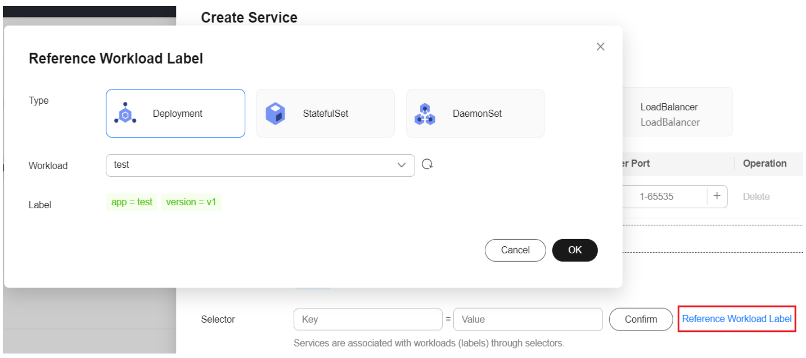
Parameter	Description	Example
Check Path	This parameter is available if you have selected **HTTP** for **Health Check**. Specify the URL for health checks. The check path must start with a slash (/) and contain 1 to 80 characters.	/

Parameter	Description	Example
Port	Health check port. The port number ranges from 1 to 65535. By default, the Service ports (node port and container port of the NodePort Service) are used.	80
Check Interval (s)	Maximum time between health checks, in seconds. The value ranges from 1 to 50.	5
Timeout (s)	Maximum time required for waiting for a response from the health check, in seconds. The value ranges from 1 to 50.	10
Max. Retries	Maximum number of health check retries. The value ranges from 1 to 10.	5

– Other clouds: Enter annotations in the key-value pair format based on your service and vendor requirements.

- **Namespace**: namespace to which the Service belongs.
- **Selector**: Services are associated with workloads (labels) through selectors. Click **Reference Workload Label** to reference the labels of an existing workload.
  – **Type**: Select the desired workload type.
  – **Workload**: Select an existing workload. If your workload is not displayed in the list, click to refresh it.
  – **Label**: After a workload is selected, its labels are displayed and cannot be modified.

Figure 3-28 Referencing a workload label



Step 6 Click **OK**.

Step 7 Obtain the access address.

1. In the navigation pane on the left, choose **Services & Ingresses**. The **Services** tab is displayed by default.
2. Click the name of the added Service to go to its details page and obtain the access address of the deployment cluster. You can access a backend pod using the EIP and port number of the load balancer.

----End

Related Operations

You can also perform operations described in **Table 3-27**.

Table 3-27 Related operations

Operation	Description
Creating a Service from a YAML file	Click **Create from YAML** in the upper right corner to create a Service from an existing YAML file.
Viewing details	1. Select the namespace to which the Service belongs. 2. (Optional) Search for a Service by its name. 3. Click the Service name to view its details, including the basic information and cluster deployment information. 4. On the **Service Details** page, click **View YAML** in the **Cluster** area to view or download YAML files of Service instances deployed in each cluster.
Editing a YAML file	Click **Edit YAML** in the row where the target Service resides to view and edit the YAML file of the Service.
Updating a Service	1. Choose **More** > **Update** in the row where the target Service resides. 2. Modify the information by referring to **Step 5**. 3. Click **OK** to submit the modified information.
Deleting a Service	Choose **More** > **Delete** in the row where the target Service resides, and click **Yes**.
Deleting Services in batches	1. Select the Services to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.7.3 Ingresses

An Ingress uses load balancers as the entry for external traffic. Compared with load balancers at Layer 4, load balancers at Layer 7 support URI configurations and distribute traffic to the corresponding Services based on the corresponding

URIs. You can customize forwarding rules based on domain names and URLs to implement fine-grained distribution of access traffic. 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 **Workloads**.

Creating an Ingress

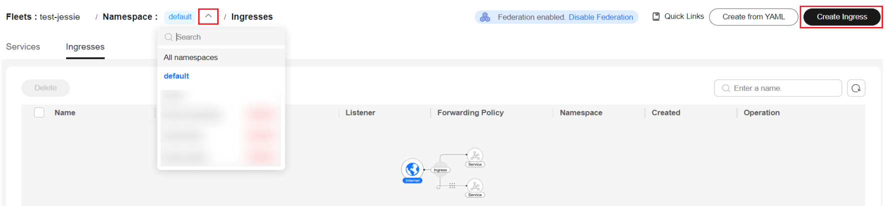
Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane on the left, choose **Services & Ingresses**. Then, click the **Ingresses** tab.

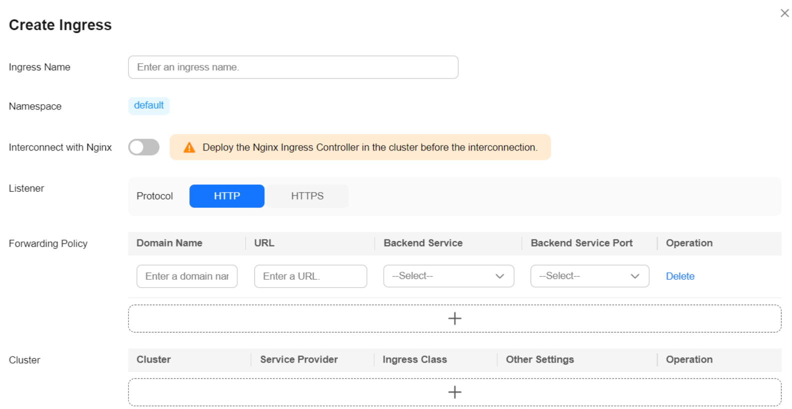
Step 4 Select the namespace that the Ingress will belong to and click **Create Ingress** in the upper right corner. For details about how to create a namespace, see **Creating a Namespace**.

Figure 3-29 Selecting a namespace



Step 5 Configure Ingress parameters.

Figure 3-30 Creating a route



- **Ingress Name**: name of the Ingress to be created.
- **Namespace**: namespace that the Ingress belongs to.
- **Interconnect with Nginx**: There are ELB Ingress Controller and Nginx Ingress Controller. Both of them are supported in UCS. ELB Ingress Controller

forwards traffic through ELB. Nginx Ingress Controller uses the templates and images maintained by the Kubernetes community to forward traffic through the Nginx component.

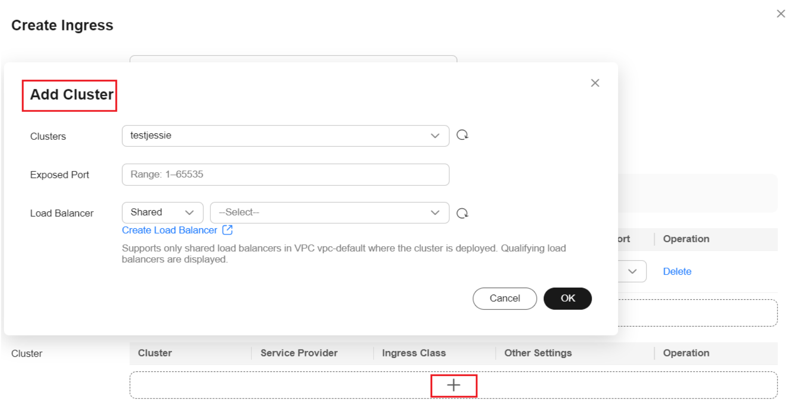
– ELB Ingress: Do not enable **Interconnect with Nginx**.

– Nginx Ingress: Click to enable **Interconnect with Nginx**.

> **CAUTION**
>
> Before creating an Nginx Ingress, install the Nginx Ingress Controller add-on for the corresponding cluster.
>
> - For details about how to install the add-on for the CCE cluster, see **Creating Nginx Ingress Controller on the Console**.
> - For details about how to install the add-on for the on-premises cluster, see **Ingress-NGINX for Load Balancing at Layer 7**.
> - For details about how to install add-ons for other types of clusters, see **Nginx Ingress Controller**.

- **Listener**: Select an external protocol. **HTTP** and **HTTPS** are supported. If you select **HTTPS**, select an IngressTLS server certificate. If no desired certificate is available, click **Create IngressTLS Secret** to create an IngressTLS secret. For details, see **Secrets**.
  – **SNI**: Server Name Indication (SNI) is an extended protocol of TLS. It allows multiple TLS-based access domain names to be provided for external systems using the same IP address and port number. Different domain names can use different security certificates.
- **Forwarding Policy**: When the access address of a request matches the forwarding policy (a forwarding policy consists of a domain name and URL, for example, 10.117.117.117:80/helloworld), the request is forwarded to the corresponding target Service for processing. You can add multiple forwarding policies.
  – **Domain Name**: (Optional) actual domain name. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.
  – **URL**: access path to be registered, for example, **/healthz**. The access path must be the same as the URL exposed by the backend application. Otherwise, a 404 error will be returned.
  – **Backend Service**: Select a Service name. You need to create the NodePort Service first. For details, see **NodePort**.
  – **Backend Service Port**: After you select the backend Service, the corresponding container port is automatically filled in.
- **Cluster**: Select the cluster where the Ingress is to be deployed.

Figure 3-31 Adding a cluster



– CCE cluster:

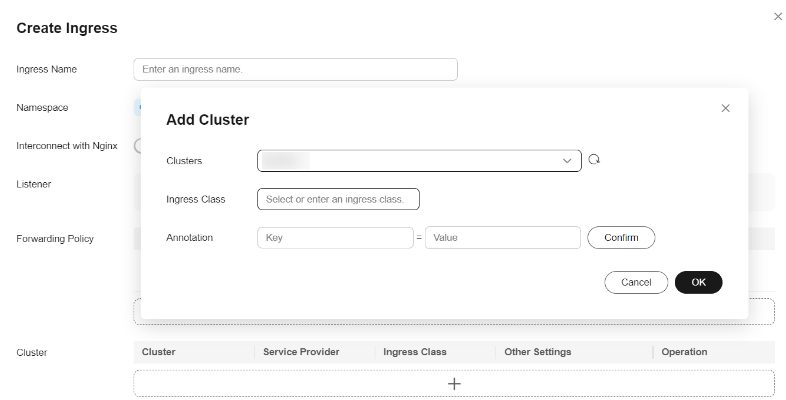
  - **Exposed Port**: port opened on the load balancer, which can be specified randomly.
  - **Load Balancer**: Only load balancers in the VPC where the cluster resides are supported. If no load balancer is available, click **Create Load Balancer**. After the load balancer is created, click the refresh button.

CAUTION

When creating an Nginx Ingress, you do not need to manually select a load balancer because a load balancer has been associated during add-on installation.

– Other clouds

Figure 3-32 Adding a cluster



  - **Ingress Class**: You can select an existing Ingress class or manually enter an Ingress class name.
  - **Annotation**: Enter an annotation in a key-value pair based on your service and vendor requirements.

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

Step 6 Click **OK**. After the Ingress is created, you can view it in the list on the **Ingresses** tab.

Step 7 Obtain the access address.

1. In the navigation pane on the left, choose **Services & Ingresses**. Then, click the **Ingresses** tab.
2. Click the name of the created Ingress. On the **Ingress Details** page displayed, view the load balancer and listener port configurations. You can access a backend pod using the EIP of the load balancer, listener port, and URL, for example, **10.117.117.117:8088/helloworld**.

----End

Related Operations

You can also perform operations described in **Table 3-28**.

Table 3-28 Related operations

Operation	Description
Creating an Ingress from a YAML file	Click **Create from YAML** in the upper right corner to create an Ingress from an existing YAML file.
Viewing details	1. Select the namespace that the Ingress belongs to. 2. (Optional) Search for an Ingress by its name. 3. Click the Ingress name to view its details, including the basic information and cluster deployment information. 4. On the **Ingress Details** page, click **View YAML** in the **Cluster** area to view or download YAML files of Ingress instances deployed in each cluster.
Editing a YAML file	Click **Edit YAML** in the row where the target Ingress resides to view and edit the YAML file of the Ingress.
Updating an Ingress	1. Choose **More** > **Update** in the row where the target Ingress resides. 2. Modify the information by referring to **Step 5**. 3. Click **OK** to submit the modified information.
Deleting an ingress	Choose **More** > **Delete** in the row where the target Ingress resides. Then, click **Yes**.
Deleting Ingresses in batches	1. Select the Ingresses to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.8 MCI

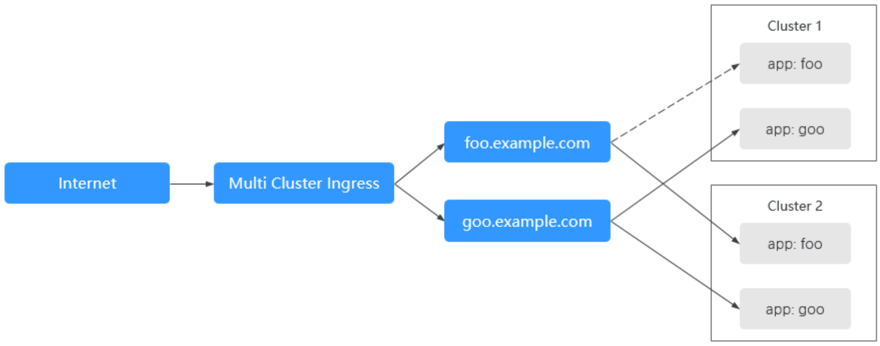
3.8.1 Overview

Why MCI?

Traditionally, each Kubernetes cluster has its load balancer and Ingress, which brings complexities around load balancing and traffic routing across clusters and regions. UCS Multi Cluster Ingress (MCI) abstracts away such complexities and improves the availability and reliability of applications.

MCI accepts traffic coming from the Internet and routes it to pods running in clusters based on forwarding rules. With MCI, you can customize forwarding rules to provide fine-grained control over how your load balancer behaves. The following diagram shows how traffic flows from MCI to two clusters. Traffic from **foo.example.com** flows to the pods that have the **app:foo** label across both clusters. Traffic from **goo.example.com** flows to the pods that have the **app:goo** label across both clusters.

Figure 3-33 MCI diagram



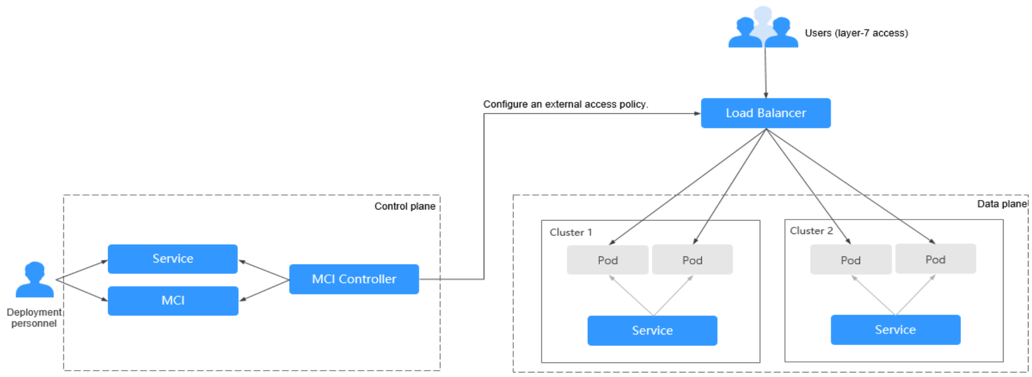
MCI has the following advantages:

- Multi-cluster load balancing: MCI provides an Ingress for traffic routing across multiple clusters, without the need to know cluster locations.
- Traffic routing: MCI allows you to customize forwarding rules based on different conditions (such as URLs, HTTP headers, and source IP addresses) to flexibly route traffic across clusters.
- High availability: MCI supports health checks and automatic traffic switchover for multi-cluster and regional high availability.
- Scalability: MCI discovers and manages application resources in multiple clusters for automatic application expansion and deployment.
- Security: MCI supports TLS security policies and certificate management for application security.

How MCI Works

MCI Controller acts as an executor for request forwarding and enables MCI functions. MCI Controller is deployed on the control plane to monitor resource object changes in real time, parse rules defined for MCI objects, and forward requests to backend services.

Figure 3-34 Working principle of MCI Controller



MCI Controller allows you to configure different domain names, ports, and forwarding rules for the same load balancer. **Figure 3-34** shows the working principle.

1. The deployment personnel create a workload on the control plane and configure a Service object for the workload.
2. The deployment personnel create an MCI object on the control plane and configure a traffic access rule that consists of the load balancer, URL, and backend service and port.
3. When detecting that the MCI object changes, the MCI Controller reconfigures the listener and backend server route according to the traffic access rule.
4. When a user accesses a workload, the traffic is forwarded to the corresponding backend service over the port based on the configured forwarding policy, and then forwarded to each associated workload through the Service.

3.8.2 Using MCI

Constraints

- MCI is only available in CCE Turbo clusters 1.21 or later.
- A Service, with both MCI and Multi-Cluster Service (MCS) configured, can only be delivered to the cluster where the Service is deployed, the cluster that accesses the Service, and the cluster where the corresponding workload is deployed in MCS.

Preparations

- If no load balancer is available in the VPC of the cluster, create a load balancer first. For details, see **Creating a Dedicated Load Balancer**. The load balancer to be created must:
  - Be a dedicated load balancer.
  - Be of the application type (HTTP/HTTPS).

– Have a private IP address to route traffic over a private network.

- MCI provides a unified entry and Layer-7 network access to cross-cluster backends. You need to deploy available workloads (Deployments) and Services in the federation in advance. If no workload or Service is available, create one by referring to **Deployments** and **ClusterIP**.

Creating an MCI Object

Step 1 Use kubectl to connect to the federation. For details, see Using kubectl to Connect to a Federation.

Step 2 Create and edit a **mci.yaml** file as follows. For details about the parameters in this file, see **Table 3-29**.

vi mci.yaml

```
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterIngress
metadata:
  name: nginx-ingress
  namespace: default
  annotations:
    karmada.io/elb.conditions.nginx-svc:
      '[{
          "type": "header",
          "headerConfig": {
              "key":"x-header",
              "values": [
              "green"
              ]
          }
      }]'
    karmada.io/elb.id: 90f9f782-1243-41cc-a57d-6157f6cb85bf
    karmada.io/elb.projectid: 65382450e8f64ac0870cd180d14e684b
    karmada.io/elb.health-check-flag: "on"
    karmada.io/elb.health-check-option.nginx-svc: '{"protocol":"TCP"}'
spec:
  ingressClassName: public-elb
  rules:
  - host: demo.localdev.me
    http:
      paths:
      - backend:
          service:
            name: nginx-svc       # Prepare a federated Service named nginx-svc.
            port:
              number: 8080       # Set the port number to 8080.
        path: /web
        pathType: Prefix
```

The structure definition of the MCI object is the same as that of the **Ingress of networking.kubernetes.io/v1** except that the backend must be set to a federated Service created on the UCS console. For details, see **ClusterIP**.

NOTICE

Parameters in the MCI file must meet the following requirements:

- **apiVersion**, **kind**, and **name** must be specified.
- **spec** cannot contain the **TLS** and **DefaultBackend** fields.
- **rules** and **paths** cannot be left blank.
- The value of **host** must be a domain name and cannot be an IP address.
- There must be a backend service specified for a Service, with correct information (such as the port number). Otherwise, the Service cannot be accessed. If you have created an MCI object with incorrect information, update the MCI object by referring to **Step 4**.
- In **paths**, the more advanced forwarding policies configured for a backend server, the earlier the backend server is configured. (**karmada.io/elb.conditions.***{service name}* indicates the advanced forwarding policy.) The earlier the backend server is configured, the higher the forwarding priority is.

  For example, if two forwarding policies a and b are configured for backend X, and one forwarding policy a is configured for backend Y, X must be configured earlier than Y in **paths**. Otherwise, traffic that meets both forwarding policies is forwarded to Y with the higher priority.
- **backend** cannot contain the **resource** field.
- The value of **path** must be an absolute path. An invalid path is as follows: invalidPathSequences = []string{"//", "/./", "/../", "%2f","%2F"}, invalidPathSuffixes = []string{"/..", "/."}.
- The value of **pathType** can be **Exact**, **Prefix**, or **ImplementationSpecific**.

Table 3-29 Key parameters

Parameter	Mandatory	Type	Description
karmada.io/elb.id	Yes	String	ID of the load balancer associated with the MCI. This parameter cannot be left blank and its value must be 1 to 32 characters long.
karmada.io/elb.projectid	Yes	String	ID of the project that the load balancer associated with the MCI belongs to. The value must be 1 to 32 characters long.
karmada.io/elb.port	No	String	Port number of the load balancer associated with the MCI. If this parameter is left blank, **80** is used by default. The value ranges from **1** to **65535**.

Parameter	Mandatory	Type	Description
karmada.io/elb.health-check-flag	No	String	Whether to enable health check. The options are as follows: ● **on**: Enable ● **off**: Disable The default value is **off**.
karmada.io/elb.health-check-option	No	**HealthCheck** Object	Health check parameters. For details, see **HealthCheck**. **NOTE** The following is an example of health check parameter settings: karmada.io/elb.health-check-option.nginx-svc: '{"protocol":"TCP","delay":"5","connect_port":"80","timeout":"1","max_retries":"1","path":"/wd"}'
karmada.io/elb.conditions.{service name}	No	Array of **Condition** Object	Advanced forwarding policy. For details, see **Condition**. *{service name}*: name of the federated Service.
karmada.io/elb.lb-algorithm.{service name}	No	String	Forwarding algorithms. The options are as follows: ● **ROUND_ROBIN**: weighted round robin ● **LEAST_CONNECTIONS**: weighted least connections ● **SOURCE_IP**: source IP hash The default value is **ROUND_ROBIN**. *{service name}*: name of the federated Service.
ingressClassName	Yes	String	Ingress controller. The value must be **public-elb**.
host	No	String	Domain name for accessing the Service. By default, this parameter is left blank, and the domain name needs to be fully matched. Ensure that the domain name has been registered and licensed. Once a forwarding policy is configured with a domain name specified, you must use the domain name for access.

Parameter	Mandatory	Type	Description
backend	No	**Backend** Object	A backend is a combination of Service and port names. HTTP (and HTTPS) requests to MCI that match the host and path of the rule are sent to the listed backend. **CAUTION** The earlier a backend server is configured in **paths**, the higher the forwarding priority is. For example, if two forwarding policies a and b are configured for backend X, and one forwarding policy a is configured for backend Y, X must be configured earlier than Y in **paths**. Otherwise, traffic that meets both forwarding policies is forwarded to Y with the higher priority.
path	Yes	String	User-defined route path. All external access requests must match **host** and **path**. **NOTE** The access path added here must exist in the backend application. Otherwise, the forwarding fails. For example, the default access URL of the Nginx application is **/usr/share/nginx/html**. When adding **/test** to the ingress forwarding policy, ensure the access URL of your Nginx application contains **/usr/share/nginx/html/test**. Otherwise, error 404 will be returned.

Parameter	Mandatory	Type	Description
pathType	Yes	String	Path type. The options are as follows: ● **ImplementationSpecific**: The matching method varies with the Ingress controller. The matching method defined by **ingress.beta.kubernetes.io/url-match-mode** is used in CCE. ● **Exact**: exact matching of the URL, which is case-sensitive. ● **Prefix**: prefix matching, which is case-sensitive. With this method, the URL path is separated into multiple elements by slashes (/) and the elements are matched one by one. If each element in the URL matches the path, the subpaths of the URL can be routed normally. **NOTE** – During prefix matching, each element must be exactly matched. If the last element of the URL is the substring of the last element in the request path, no matching is performed. For example, **/foo/bar** matches **/foo/bar/baz** but does not match **/foo/barbaz**. – If the URL or request path ends with a slash (/), the slash (/) will be ignored. For example, **/foo/bar** matches **/foo/bar/**. See **examples** of Ingress path matching.

Table 3-30 HealthCheck parameters

Parameter	Mandatory	Type	Description
protocol	No	String	Protocol used for health checks. The value can be **TCP** or **HTTP**.
connect_port	No	Int	Port used for health checks. The value ranges from **1** to **65535**.
delay	No	Integer	The interval between the time when the application is delivered and the time when a health check is started, in seconds. The value ranges from **1** to **50**.
timeout	No	Integer	Health check timeout duration, in seconds. The value ranges from **1** to **50**.
path	No	String	Health check request URL. This parameter is valid only when **type** is set to **HTTP** or **HTTPS**. The default value is **/**. Enter 1 to 80 characters starting with a slash (/). Only letters, digits, hyphens (-), slashes (/), periods (.), percent signs (%), question marks (?), number signs (#), ampersands (&), and extended character sets are allowed.
max_retries	No	Integer	Maximum number of retries. The value ranges from **1** to **10**.

Table 3-31 Condition parameters

Parameter	Mandatory	Type	Description
type	Yes	String	Type of the advanced forwarding policy. Currently, only header is supported.
headerConfig	Yes	**headerConfig** Object	Advanced forwarding policy object. For details, see **headerConfig**.

Table 3-32 headerConfig parameters

Parameter	Mandatory	Type	Description
key	Yes	String	Name of the header parameter. Enter 1 to 40 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.
values	Yes	String array	Values of the header parameter. Enter 1 to 128 characters. Asterisks (*) and question marks (?) are allowed, but spaces and double quotation marks are not allowed. An asterisk can match zero or more characters, and a question mark can match 1 character.

Step 3 Run the following command to create an MCI object:

kubectl apply -f mci.yaml

Information similar to the following is displayed:

```
multiClusterIngress.networking.karmada.io/nginx-ingress created
```

Step 4 After creating the MCI object, perform operations on it. **nginx-ingress** is the name of the MCI object.

- To obtain the MCI object, run **kubectl get mci nginx-ingress**.
- To update the MCI object, run **kubectl edit mci nginx-ingress**.
- To delete the MCI object, run **kubectl delete mci nginx-ingress**.

----End

Accessing Services Through MCI

After an MCI object is created, you can access the backends through **http://***IP:port/path*. *IP:port* indicates the IP address and port number of the load balancer associated with the MCI object, and *path* indicates the path defined in the MCI object.

You can also set an external domain name in the MCI object so that you can access the load balancer using the domain name and then access backend services.

```
spec:
  rules:
  - host: www.example.com       # Domain name
    http:
      paths:
      - path: /
        backend:
```

```
      serviceName: nginx    # Prepare a federated Service named nginx.
      servicePort: 80       # Set the port number to 80.
```

NOTE

To access the load balancer using a domain name, you need to point the domain name to the IP address of the load balancer. For details about how to configure record sets for the domain name, see **Domain Name Service (DNS)**.

3.8.3 Configuring Automatic Traffic Switchover

3.8.3.1 Overview

Why Automatic Traffic Switchover?

MCI provides load balancing and traffic routing across clusters to improve the availability and reliability of applications. However, when a cluster is faulty, service requests allocated by MCI to the cluster will be rejected.

UCS provides automatic traffic switchover to automatically redirect traffic to an available cluster for service availability. You can use this feature in the following scenarios:

- Cluster fault identification and automatic traffic switchover: When CoreDNS in a cluster is faulty, the system automatically detects the fault and reports the fault to the control plane in a timely manner. Traffic will then be redirected to an available cluster to prevent service unavailability caused by a single component failure. For details, see **Configuring Conditional Automatic Traffic Switchover**.
- Traffic switchover in advance for smooth upgrade: The traffic to a cluster is redirected before the cluster upgrade and then routed back to the cluster once the upgrade is complete. In this way, the upgrade of a cluster will not affect the availability of services. For details, see **Configuring Unconditional Automatic Traffic Switchover**.

Constraints

Automatic traffic switchover is only available for CCE Turbo clusters 1.21 or later.

3.8.3.2 Configuring Unconditional Automatic Traffic Switchover

When the administrator of a cluster performs operations such as cluster upgrade, the cluster may be unavailable due to inappropriate upgrade policy, incorrect upgrade configuration, or incorrect operations performed by the operator. This section describes how to create a Remedy object to unconditionally redirect the traffic of the cluster to be upgraded.

Remedy objects can be created to perform specific actions when certain conditions are met. Before upgrading the target cluster (for example, member1), the cluster administrator can create the following Remedy object to redirect the traffic of member1.

The following is an example YAML file for creating a Remedy object. If the trigger condition is left empty, traffic switchover is triggered unconditionally. The cluster federation controller will immediately redirect the traffic of member1. After the cluster is successfully upgraded, delete the Remedy object. The traffic is automatically routed back to member1. In this way, the upgrade of a cluster will not affect the availability of services. For details about the parameters of the Remedy object, see **Table 3-33**.

```
apiVersion: remedy.karmada.io/v1alpha1
kind: Remedy
metadata:
  name: foo
spec:
  clusterAffinity:
    clusterNames:
      - member1
  actions:
  - TrafficControl
```

Table 3-33 Remedy parameters

Parameter	Description
spec.clusterAffinity.clusterNames	List of clusters controlled by the policy. The specified action is performed only for clusters in the list. If this parameter is left blank, no action is performed.
spec.decisionMatches	Trigger condition list. When a cluster in the cluster list meets any trigger condition, the specified action is performed. If this parameter is left blank, the specified action is triggered unconditionally.
conditionType	Type of a trigger condition. Only **ServiceDomainNameResolutionReady** (domain name resolution of CoreDNS reported by CPD) is supported.
operator	Judgment logic. Only **Equal** (equal to) and **NotEqual** (not equal to) are supported.
conditionStatus	Status of a trigger condition.
actions	Action to be performed by the policy. Currently, only **TrafficControl** (traffic control) is supported.

3.8.3.3 Configuring Conditional Automatic Traffic Switchover

This section describes how to configure conditional automatic traffic switchover to identify CoreDNS faults in a cluster and automatically redirect traffic.

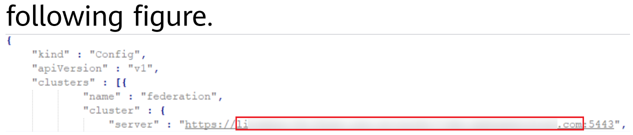
Installing CPD for a Cluster to Identify Faults

Before configuring automatic traffic switchover, you need to install cluster-problem-detector (CPD) in a cluster to automatically detect whether CoreDNS runs normally and report the results.

CPD periodically checks whether CoreDNS can resolve **kubernetes.default** and updates the result to **conditions** of the node object. The active CPD pod collects **conditions** on each node, determines whether cluster domain name resolution is normal, and reports the result to the federation control plane of the cluster.

CPD needs to be independently deployed as a DaemonSet on all nodes in each cluster. The following is an example CPD configuration file. You can modify the parameters by referring to **Table 3-34**.

Table 3-34 CPD parameters

Parameter	Description
<federation-version>	Version of the federation that the cluster belongs to. On the **Fleets** tab, click the fleet name to obtain the version.
<your-cluster-name>	Name of the cluster where CPD is to be installed.
<kubeconfig-of-karmada>	The kubeconfig file of the federation control plane. For details about how to download the kubeconfig file that meets the requirements, see **kubeconfig**. **CAUTION** ● When downloading the kubeconfig file, you need to select the VPC where the cluster resides, or the VPC that can communicate with the VPC where the cluster resides over a Cloud Connect or VPC peering connection. ● If the IP address of the federation control plane in the kubeconfig file is set to a domain name, you need to configure **hostAliases** in the YAML file.
hostAliases	If the IP address of the federation control plane in the kubeconfig file is set to a domain name, you need to configure **hostAliases** in the YAML file. If the IP address is not a domain name, delete **hostAliases** from the YAML file. ● Replace **<host name of karmada server>** with the domain name of the federation control plane. To obtain the domain name of the federation control plane, view the **server** field in the kubeconfig file, as shown in the following figure.  ● Replace **<ip of host name of karmada server>** with the IP address mapped to the domain name of the federation control plane. Access the domain name of the federation control plane on the cluster node to obtain the resolved IP address.
coredns-detect-period	Interval for CoreDNS to detect and report data, which defaults to **5s** (recommended value). A smaller value indicates more frequent data detection and reporting.

Parameter	Description
coredns-success-threshold	Threshold of the duration in which CoreDNS successfully resolves a domain name, which defaults to **30s** (recommended value). If the duration exceeds this threshold, CoreDNS is normal. A higher value indicates more stable detection but lower sensitivity, while a lower value indicates less stable detection but higher sensitivity.
coredns-failure-threshold	Threshold of the duration in which CoreDNS fails to resolve a domain name, which defaults to **30s** (recommended value). If the duration exceeds this threshold, CoreDNS is faulty. A higher value indicates more stable detection but lower sensitivity, while a lower value indicates less stable detection but higher sensitivity.

```
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: cluster-problem-detector
  namespace: kube-system
  labels:
    app: cluster-problem-detector
spec:
  selector:
    matchLabels:
      app: cluster-problem-detector
  template:
    metadata:
      labels:
        app: cluster-problem-detector
    spec:
      containers:
        - image: swr.ap-southeast-3.myhuaweicloud.com/hwofficial/cluster-problem-detector:<federation-
version>
          name: cluster-problem-detector
          command:
            - /bin/sh
            - '-c'
            - /var/paas/cluster-problem-detector/cluster-problem-detector
              --karmada-kubeconfig=/tmp/config
              --karmada-context=federation
              --cluster-name=<your-cluster-name>
              --host-name=${HOST_NAME}
              --bind-address=${POD_ADDRESS}
              --healthz-port=8081
              --detectors=*
              --coredns-detect-period=5s
              --coredns-success-threshold=30s
              --coredns-failure-threshold=30s
              --coredns-stale-threshold=60s
          env:
            - name: POD_ADDRESS
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: status.podIP
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: POD_NAMESPACE
```

```
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.namespace
            - name: HOST_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: spec.nodeName
          livenessProbe:
            httpGet:
              path: /healthz
              port: 8081
              scheme: HTTP
            initialDelaySeconds: 3
            timeoutSeconds: 3
            periodSeconds: 5
            successThreshold: 1
            failureThreshold: 3
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8081
              scheme: HTTP
            initialDelaySeconds: 3
            timeoutSeconds: 3
            periodSeconds: 5
            successThreshold: 1
            failureThreshold: 3
          volumeMounts:
            - mountPath: /tmp
              name: karmada-config
      serviceAccountName: cluster-problem-detector
      volumes:
        - configMap:
            name: karmada-kubeconfig
            items:
              - key: kubeconfig
                path: config
          name: karmada-config
      securityContext:
        fsGroup: 10000
        runAsUser: 10000
        seccompProfile:
          type: RuntimeDefault
      hostAliases:
        hostnames:
          - <host name of karmada server>
        ip: <ip of host name of karmada server>
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cluster-problem-detector
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cpd-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:cluster-problem-detector
subjects:
  - kind: ServiceAccount
    name: cluster-problem-detector
    namespace: kube-system
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:cluster-problem-detector
rules:
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - nodes/status
    verbs:
      - patch
      - update
  - apiGroups:
      - ""
      - events.k8s.io
    resources:
      - events
    verbs:
      - create
      - patch
      - update
  - apiGroups:
      - coordination.k8s.io
    resources:
      - leases
    verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: karmada-kubeconfig
  namespace: kube-system
data:
  kubeconfig: |+
    <kubeconfig-of-karmada>
```

Checking Whether CPD Runs Normally

After deploying CPD, check whether CPD runs normally.

- Run the following command to check whether the **ServiceDomainNameResolutionReady** condition exists in **conditions** of the node and whether **lastHeartBeatTime** of this condition is updated in a timely manner:

  kubectl get node *<node-name>* **-oyaml | grep** -B4 **ServiceDomainNameResolutionReady**

  If the condition does not exist or **lastHeartBeatTime** of the condition is not updated for a long time:

  a. Check whether the CPD pod is in the **Ready** state.

b. Check whether there is a **LoadCorednsConditionFailed** or **StoreCorednsConditionFailed** event in the member cluster. If the event exists, rectify the fault based on the error message in the event.

- Run the following command to check whether the **ServiceDomainNameResolutionReady** condition exists in the federation cluster object:

  kubectl --kubeconfig *<kubeconfig-of-federation>* **get cluster** *<cluster-name>* **-oyaml | grep ServiceDomainNameResolutionReady**

  If the cluster object does not contain the preceding condition:

  a. Check "failed to sync corendns condition to control plane, requeuing" in the CPD log.

  b. Check the kubeconfig file configuration. If the kubeconfig file configuration is updated, deploy CPD again.

  c. Check the network connectivity between the node where CPD resides and the VPC of the cluster you selected when the kubeconfig file is downloaded.

Configuring a Policy for Conditional Automatic Traffic Switchover

Once CPD is deployed and runs normally, you need to create a Remedy object to perform specific actions when certain conditions are met. For example, if CoreDNS in a cluster is faulty, the cluster traffic will be redirected to an available cluster.

The following is an example configuration file of the Remedy object. The Remedy object is defined to report exceptions of CoreDNS using CPD in the cluster member1 or member2. If CoreDNS is faulty, the cluster traffic will be redirected to an available cluster automatically. For details about the parameters of the Remedy object, see **Table 3-35**.

```
apiVersion: remedy.karmada.io/v1alpha1
kind: Remedy
metadata:
  name: foo
spec:
  clusterAffinity:
    clusterNames:
      - member1
      - member2
  decisionMatches:
  - clusterConditionMatch:
      conditionType: ServiceDomainNameResolutionReady
      operator: Equal
      conditionStatus: "False"
  actions:
  - TrafficControl
```

Table 3-35 Remedy parameters

Parameter	Description
spec.clusterAffinity.clusterNames	List of clusters controlled by the policy. The specified action is performed only for clusters in the list. If this parameter is left blank, no action is performed.

Parameter	Description
spec.decisionMatches	Trigger condition list. When a cluster in the cluster list meets any trigger condition, the specified action is performed. If this parameter is left blank, the specified action is triggered unconditionally.
conditionType	Type of a trigger condition. Only **ServiceDomainNameResolutionReady** (domain name resolution of CoreDNS reported by CPD) is supported.
operator	Judgment logic. Only **Equal** (equal to) and **NotEqual** (not equal to) are supported.
conditionStatus	Status of a trigger condition.
actions	Action to be performed by the policy. Currently, only **TrafficControl** (traffic control) is supported.

3.9 MCS

3.9.1 Overview

Why MCS?

A **Service** in Kubernetes is an object that defines a logical set of pods and a policy to access the pods. The Service provides a stable IP address and DNS name for accessing pods. The Service lets you discover and access available pods in a single cluster. However, sometimes you might want to split applications into multiple clusters, to address data sovereignty, state management, and scalability requirements. With MCS, you can build applications that span multiple clusters.

MCS is a cross-cluster Service discovery and invocation mechanism that leverages the existing Service object. Services enabled with this feature are discoverable and accessible across clusters.

Using MCS provides you with the following benefits:

- Application DR: Running the same Service across clusters in multiple regions provides you with improved fault tolerance. If a Service in a cluster is unavailable, the request can fail over and be served from other clusters.
- Service sharing: Instead of each cluster requiring its own local Service replica, MCS makes it easier to set up common shared Services (such as monitoring and logging) in a separate cluster that all functional clusters use.
- Application migration: MCS provides you with a mechanism to help bridge the communication between Services, making it easier to migrate your applications. This is especially helpful as you can deploy the same Service to two different clusters and traffic is allowed to shift from one cluster or application to another.

How MCS Works

MCS functions are implemented by the control plane component karmada-controller-manager. karmada-controller-manager monitors Service and MCS changes in real time, parses rules defined by MCS objects, and forwards requests to backend services.

Figure 3-35 Working principle of MCS

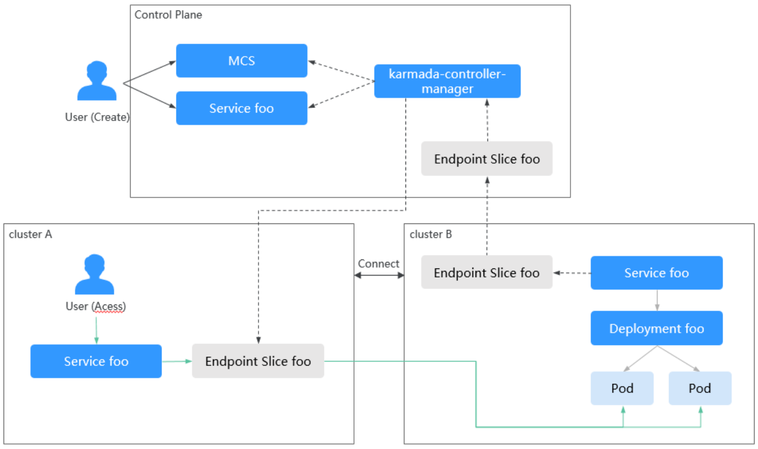


Figure 3-35 shows the working principle of MCS. The details are as follows:

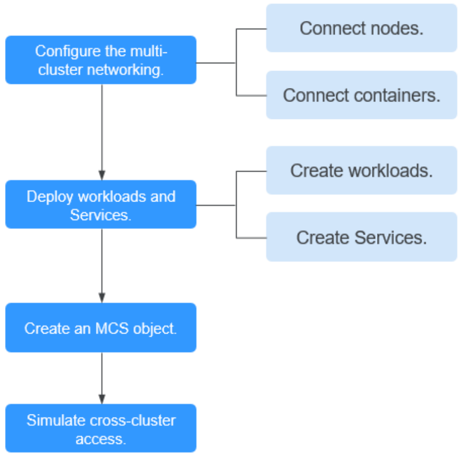
1. A user creates a workload on the federation control plane and deploys Service **foo** in cluster B for the workload.
2. The user creates an MCS object on the federation control plane and configures an access rule. In the rule, the Service is delivered to cluster B and a user can access the Service from cluster A.
3. karmada-controller-manager monitors the changes of the Service and MCS object, delivers the Service to cluster B, and collects and sends endpoint slices of cluster B to cluster A.
4. When a user accesses the Service from cluster A, the request is routed to the service backend of cluster B. This is how cross-cluster service discovery and access occur.

Process of Using MCS

Figure 3-36 shows the process of using MCS. The details are as follows:

1. Check the connectivity of both cluster nodes and containers. If they cannot communicate with each other, connect them as required. For details, see **Configuring the Multi-Cluster Networking**.
2. Deploy available workloads and Services in the federation in advance. For details, see **Preparations**.
3. Create an MCS object and configure an access rule. For details, see **Creating an MCS Object**.
4. Simulate cross-cluster access. This means the Service delivered to one cluster will be accessed by the other cluster configured in MCS. For details, see **Cross-Cluster Access**.

Figure 3-36 Process of using MCS



3.9.2 Using MCS

3.9.2.1 Configuring the Multi-Cluster Networking

Before creating an MCS object, ensure connectivity of both inter-cluster nodes and containers.

Check the connectivity by referring to **Table 3-36**. If the network between nodes or containers is not connected, connect them. If they still cannot be connected, rectify the fault by referring to **FAQ**.

Table 3-36 Connectivity of both inter-cluster nodes and containers

Inter-Cluster Network	How to Check	How to Connect
Node connectivity	Ping the IP address of a node in cluster B from cluster A. If the ping succeeds, the node connectivity is normal.	1. Configure the network type. Set the network type to underlay for inter-cluster pod communication. For details, see **Cluster Network Types**. 2. Connect the network between clusters. ● Network connectivity between CCE clusters: CCE clusters in the same VPC can communicate with each other by default. Use a VPC peering connection to connect CCE clusters across VPCs. ● Network connectivity between on-premises clusters: Set Cilium to the underlay mode and enable BGP for Cilium. For details, see **Cilium**. ● Network connectivity between clusters of other types: Enable the network between clusters of other types as required.
Container connectivity	Use cURL to access a pod in cluster B from cluster A. If the access succeeds, the container connectivity is normal.	

Cluster Network Types

Ensure that underlay networks are supported for inter-cluster pod communication. The following table lists the types of clusters that support underlay networks.

Table 3-37 Types of clusters that support underlay networks

Cluster Type	Cluster Subtype	Network Type	Support Underlay Network
Huawei Cloud clusters	CCE clusters	Container tunnel network	No
		VPC network	Yes
	CCE Turbo clusters	Cloud native network 2.0	Yes

Cluster Type	Cluster Subtype	Network Type	Support Underlay Network
On-premises clusters	On-premises clusters	Overlay and underlay networks The overlay network is used by default. You need to manually enable the underlay network. For details about the underlay network, see **Cilium**.	Yes
Multi-cloud clusters	Multi-cloud clusters	Overlay and underlay networks The overlay network is used by default. You need to manually enable the underlay network. For details about the underlay network, see **Cilium**.	Yes
Attached clusters	Attached clusters	Unknown	Network type–based

FAQ

If nodes or containers in different clusters cannot access each other, check the items listed in the following table.

Table 3-38

Check Item	Fault Locating	Solution
Whether the cluster version is v1.21 or later	Access the cluster details page.	Upgrade the cluster. For details, see **Upgrading a Cluster**.
Whether clusters can be accessed over an underlay network	Check this item by referring to **Table 3-37**.	Configure the network type by referring to **Table 3-37**.
Whether CIDR blocks overlap in the routes of the VPC peering connection	Go to the VPC peering connection details page.	Modify the overlapped CIDR blocks.

3.9.2.2 Creating an MCS Object

Constraints

- MCS is available only in clusters 1.21 or later.
- A Service, with both MCI and Multi-Cluster Service (MCS) configured, can only be delivered to the cluster where the Service is deployed, the cluster that accesses the Service, and the cluster where the corresponding workload is deployed in MCS.

Preparations

- Deploying Workloads and Services
  Deploy available workloads (Deployments) and Services on the federation control plane. If no workload or Service is available, create one by referring to **Deployments** and **ClusterIP**.
- Configuring the Cluster Network
  Check and configure the network connectivity of both inter-cluster nodes and containers by referring to **Configuring the Multi-Cluster Networking**.

Creating an MCS Object

Step 1 Use kubectl to connect to the federation. For details, see Using kubectl to Connect to a Federation.

Step 2 Create and edit an **mcs.yaml** file. For details about the parameters, see **Table 3-39**.

vi mcs.yaml

In the example, the defined MCS object is associated with Service **foo**. This Service is deployed in cluster B and can be accessed from cluster A.

```
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterService
metadata:
  name: foo                     # MCS object name
  namespace: default                # Name of the namespace where the MCS object is located
spec:
  types:
    - CrossCluster              # Inter-cluster service discovery
  providerClusters:              # Cluster that the Service is delivered to
    - name: clusterB
  consumerClusters:                # Cluster that accesses the Service
    - name: clusterA
```

Table 3-39 Key parameters

Parameter	Mandatory	Type	Description
metadata.name	Yes	String	Name of the MCS object, which must be the same as that of the associated Service.

Parameter	Mandatory	Type	Description
metadata.namespace	No	String	Name of the namespace where the MCS object is located, which must be the same as that of the namespace where the associated Service is located. If this parameter is left blank, **default** is used.
spec.types	Yes	String	Traffic direction. To enable service discovery across clusters, set this parameter to **CrossCluster**.
spec.providerClusters.name	No	String	Name of the cluster that the Service is delivered to. Set this parameter to the cluster where the Service is deployed. If this parameter is left blank, the Service is delivered to all clusters in the federation by default. **CAUTION** If a Service is deployed in cluster B but cluster A and cluster B are both configured as the delivery targets, the Service is delivered to both clusters. The original Service with the same name in cluster A will be overwritten.
spec.consumerClusters.name	No	String	Name of the cluster that accesses the Service. Set this parameter to the name of the cluster that is expected to access the Service across clusters through MCS. If this parameter is left blank, all clusters in the federation can access the Service by default.

Step 3 Create an MCS object.

kubectl apply -f mcs.yaml

Step 4 Check the status of the MCS object (named **foo**).

kubectl describe mcs foo

The **status** field in the YAML file records the MCS object status. If the following information is displayed, the endpoint slices are successfully delivered and synchronized, and cross-cluster service discovery is available:

```
status:
  conditions:
  - lastTransitionTime: "2023-11-20T02:30:49Z"
  message: EndpointSlices are propagated to target clusters.
    reason: EndpointSliceAppliedSuccess
    status: "True"
  type: EndpointSliceApplied
```

Run the following commands to operate the MCS object (named **foo**):

- **kubectl get mcs foo**: obtains the MCS object.
- **kubectl edit mcs foo**: updates the MCS object.
- **kubectl delete mcs foo**: deletes the MCS object.

----End

Cross-Cluster Access

After the MCS object is created, you can access the Service from the cluster specified by **consumerClusters.name**.

In the cluster specified by **consumerClusters.name**, create a pod, access the container, and run the **curl http://***Service name*:*Port number* command to access the Service.

If the following information is displayed, the access is successful:

```
/ # curl http://Service name:Port number
...
<h1>Welcome to foo!</h1>
...
```

3.10 DNS Policies

Applications deployed in different clusters can be accessed using a unified public domain name. After you configure a public domain name, UCS can use it as a root domain name to generate a complete domain name for applications. You can configure a DNS policy 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

Before configuring a DNS policy for an application, ensure that the domain name has been registered with the domain name service provider and submitted for ICP license. Otherwise, the domain name cannot be accessed.

If you have a registered and licensed domain name, go to **Step 3** to create a public zone.

If you have not registered a domain name, **buy a public zone** and complete the licensing, resolution, and configuration of the domain name as prompted. The procedure for domain name registration and licensing is as follows:

Step 1 Buy a public domain name, for example, **ucsclub.cn**.

- If you have not purchased a public zone, **buy one**.
- If you have bought a public domain name, go to **Step 2**.

Step 2 Submit your domain name for license.

- If your public domain name has not been licensed, apply for a license at the **Huawei Cloud ICP License Service**.
- If your public domain name has been licensed, go to **Step 3**.

Step 3 Create a public zone.

- If you have not created a public zone, **create one**.
- If you have created a public zone, go to **Step 4**.

Step 4 Configure a domain name.

Select the domain name that has been configured and click **Set**.

----End

Creating a DNS Policy

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 DNS Policy**.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **DNS Policies** in the navigation pane, and click **Create DNS Policy**.

Step 4 Set parameters of the associated Service.

- **Namespace**: Select a namespace.
- **Target Service**: Select a target Service. If no LoadBalancer Service is available, create one first. For details about how to create a Service, see **LoadBalancer**.

Step 5 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.
- **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 6 Click **Create DNS Policy**. The creation task will take a period of time. You can click **Back to DNS Policies** or **View DNS Policy Details** to view the created DNS policy.

----End

Modifying an Alias

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **DNS Policies** in the navigation pane and click the name of a policy to access its details page.

Step 4 Click , enter an alias, and click .

----End

Modifying the Traffic Distribution Ratio

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **DNS Policies** in the navigation pane and click the name of a policy to access its details page.

Step 4 On the topology tab, click **Edit**.

Step 5 Modify parameters and click **OK**.

----End

Viewing the DNS Policy Address

After a DNS policy is created, you can view its address in the DNS policy list.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **DNS Policies** in the navigation pane. In the DNS policy list, view the value in the **Domain Name** column.

----End

Deleting a DNS Policy

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Click **Delete** in the **Operation** column of the target DNS policy.

Step 4 In the **Delete DNS Policy** dialog box, click **Yes**.

----End

3.11 Storage

3.11.1 Overview

You can configure a storage class in the **Add Container** step of creating a workload.

Local Storage

Mount the file directory of the host where a container is located to a specified container path (corresponding to hostPath in Kubernetes). Alternatively, you can leave the source path empty (corresponding to emptyDir in Kubernetes). If the source path is left empty, a temporary directory of the host will be mounted to the mounting point of the container. A specified source path is used when data needs to be persistently stored on the host, while emptyDir is used when temporary storage is needed. A ConfigMap is a type of resource that stores configuration information required by a workload. Its content is user-defined. A secret is a type of resource that holds sensitive data, such as authentication and key information, required by a workload. Its content is user-defined. For details, see **Mounting a Local Volume**.

PersistentVolumeClaims (PVCs)

You can create persistent volumes and mount them to a container path. When containers are migrated, the cloud storage is mounted to the new container to ensure data reliability. For details, see **Mounting a PV**. Therefore, you are advised to select PVCs when creating a workload and store pod data on the corresponding cloud storage. If you store pod data on a local volume and a fault occurs on the node, the data cannot be restored.

- Huawei Cloud cluster: UCS allows you to use Elastic Volume Service (EVS), Object Storage Service (OBS), and Scalable File Service (SFS) and mounts them to the container path of the Huawei Cloud cluster.
  - EVS: offers scalable block storage with high reliability, high performance, and extensive specifications for containers. EVS stores binary data and cannot store files directly. This storage class is applicable when data needs to be stored permanently.
  - SFS: provides high-performance file storage (NAS) that can be expanded on demand. It provides shared file access for containers and is used for persistent storage in ReadWriteMany scenarios, including media processing, content management and web services, big data and application analysis.
  - OBS: provides unlimited storage capacity, stores files in any format, and caters to the needs of common users. It is mainly designed for scenarios involving storage and analysis of massive amounts of data, query of historical data details, analysis on a large number of behavior logs, and statistical analysis on public transactions.
- When a non-Huawei Cloud cluster uses a PVC to mount cloud storage, the cluster provider must support StorageClasses. For details, see **Storage Classes**.

3.11.2 Mounting a Local Volume

Scenarios

There are four types of local volumes:

- hostPath: mounts a file directory of the host where the container is located to the specified mount point of the container. For example, if the container needs to access **/etc/hosts**, you can use hostPath volume to map **/etc/hosts**.

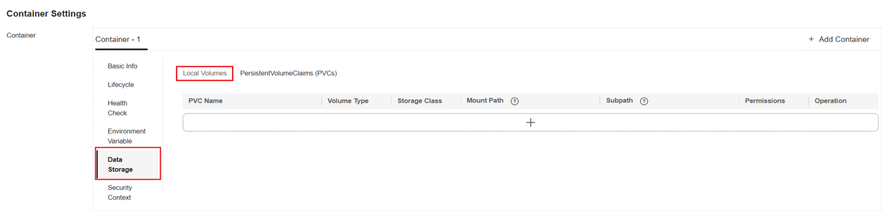
- emptyDir: applies to temporary data storage, disaster recovery, and runtime data sharing. It will be deleted upon deletion or transfer of workload pods. The lifecycle is the same as that of the container pod. When the pod is deleted, the emptyDir volume is deleted and its data is lost.
- ConfigMap: After you mount a ConfigMap to a container, you can read the ConfigMap data from the mount path of the container.
- Secret: After you mount a secret to a container, you can read the secret data from the mount path of the container.

HostPath

HostPath is a path for mounting a file or directory from a host's file system into a container. Such a volume is usually used to store containerized application logs that need to be stored permanently or containerized applications that need to access internal data structure of the Docker engine on the host.

Step 1 Set the basic container information by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**. After setting the basic container information, click **Data Storage**. On the **Local Volumes** tab page, click +.

Figure 3-37 Container storage configuration



Step 2 Set parameters for adding a local volume, as listed in **Table 3-40**.

Table 3-40 HostPath parameters

Parameter	Description
Volume Type	Select **hostPath**.
hostPath	Path on the host, for example, **/etc/hosts**.
Mount Path	Container path to which the data volume will be mounted. **NOTICE** • The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, the container may not function normally. Select an empty directory. 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, and the container cannot start normally. As a result, the application may not be created. • If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

Parameter	Description
Subpath	A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used.
Permissions	• **Read-only**: You can only read the data in the mounted volume. • **Read-write**: You can modify the volume mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 3 You can add multiple settings. Click **OK** to complete the configuration.

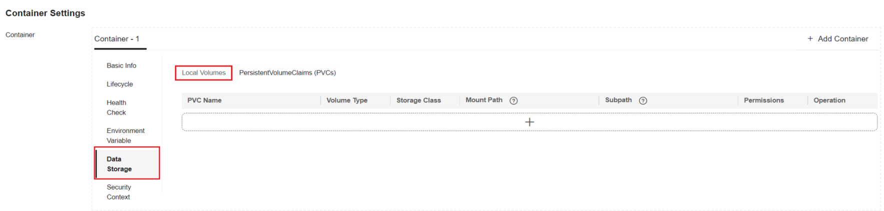
----End

emptyDir

emptyDir applies to temporary data storage, disaster recovery, and runtime data sharing. It will be deleted upon deletion or transfer of workload pods.

Step 1 Set the basic container information by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**. After setting the basic container information, click **Data Storage**. On the **Local Volumes** tab page, click +.

Figure 3-38 Container storage configuration



Step 2 Set parameters for adding a local volume, as listed in **Table 3-41**.

Table 3-41 emptyDir parameters

Parameter	Description
Volume Type	Select **emptyDir**.
Medium	• **Default**: Data is stored in disks. This approach is used when there is a large amount of data, with low requirements on reading and writing efficiency. • **Memory**: You can select this option to improve the running speed, but the storage capacity is subject to the memory size. This mode applies to a small amount of data with high requirements on reading and writing efficiency.

Parameter	Description
Mount Path	Container path to which the data volume will be mounted. **NOTICE** • The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, the container may not function normally. Select an empty directory. 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, and the container cannot start normally. As a result, the application may not be created. • If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used.
Permissions	• **Read-only**: You can only read the data in the mounted volume. • **Read-write**: You can modify the volume mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 3 You can add multiple settings. Click **OK** to complete the configuration.

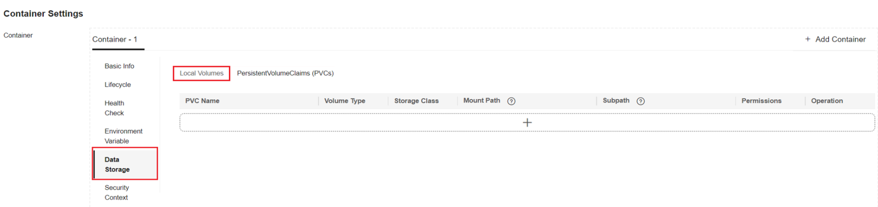
----End

ConfigMap

ConfigMap is used to process workload configuration parameters. Before that, you need to create ConfigMaps. For details, see **ConfigMaps**.

Step 1 Set the basic container information by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**. After setting the basic container information, click **Data Storage**. On the **Local Volumes** tab page, click +.

Figure 3-39 Container storage configuration



Step 2 Set parameters for adding a local volume, as listed in **Table 3-42**.

Table 3-42 ConfigMap parameters

Parameter	Description
Storage Type	Select **ConfigMap**.
ConfigMap	Select the desired ConfigMap name. **NOTE** A ConfigMap must be created in advance. For details, see **ConfigMaps**.
Mount Path	Container path to which the data volume will be mounted. **NOTICE** ● The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, the container may not function normally. Select an empty directory. 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, and the container cannot start normally. As a result, the application may not be created. ● If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used.
Permissions	Only **Read-only** is supported. You can only read the file system in the container path.

Step 3 You can add multiple settings. Click **OK** to complete the configuration.

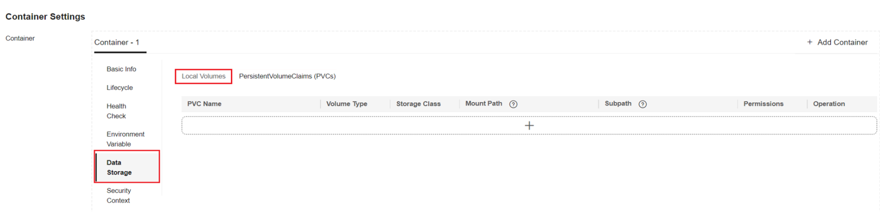
----End

Secret

Mount the data in the secret to the specified container. The content of the secret is user-defined. Before that, you need to create a secret. For details, see **Secrets**.

Step 1 Set the basic container information by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**. After setting the basic container information, click **Data Storage**. On the **Local Volumes** tab page, click + .

Figure 3-40 Container storage configuration



Step 2 Set parameters for adding a local volume, as listed in **Table 3-43**.

Table 3-43 Secret parameters

Parameter	Description
Volume Type	Select **Secret**.
Secrets	Select the desired secret name. **NOTE** A secret must be created in advance. For details, see **Secrets**.
Mount Path	Container path to which the data volume will be mounted. **NOTICE** • The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, the container may not function normally. Select an empty directory. 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, and the container cannot start normally. As a result, the application may not be created. • If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.
Subpath	A subpath is used to mount a local volume so that the same data volume is used in a single pod. If this parameter is left blank, the root path is used.
Permissions	Only **Read-only** is supported. You can only read the file system in the container path.

Step 3 You can add multiple settings. Click **OK** to complete the configuration.

----End

3.11.3 Mounting a PV

A PVC provides persistent storage management for containers in multiple clouds. The cloud storage can be mounted to containers based on actual requirements, ensuring high reliability of applications.

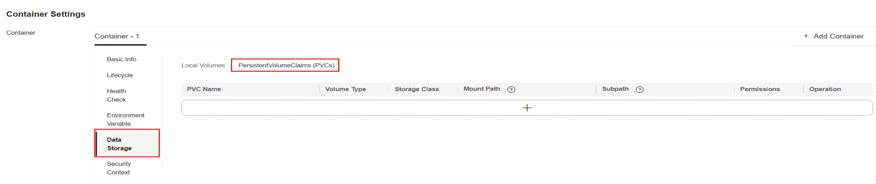
NOTICE

- After a PVC is created on the UCS console, a PVC with the same name is automatically created in your cluster. Also a PersistentVolume (PV) is created and bound with the PVC. If you are not familiar with the relationship among PVs, PVCs, and StorageClasses in Kubernetes, see **Persistent Storage**.
- You can modify or delete the PVCs automatically created by UCS on the cluster details page. However, if the PVC settings on the UCS console are not modified accordingly, the modified or deleted PVCs will be re-created by UCS. Therefore, you are advised to change the settings on the UCS console, not on the cluster details page.
- When a non-Huawei Cloud cluster uses a PVC to mount cloud storage, the cluster provider must support StorageClasses to dynamically create PVs. Run the following command to query the StorageClass configuration and the interconnected backend storage resources of the cluster. For more information about StorageClass, see **Storage Classes**.
```
kubectl get storageclass
```

Mounting a PVC to a Cloud Storage Volume

Step 1 Set the basic container information by referring to **Creating a Deployment**, **Creating a StatefulSet**, or **Creating a DaemonSet**. After setting the basic container information, click **Data Storage**. On the **PersistentVolumeClaims (PVCs)** tab page, click +.

Figure 3-41 Container storage configuration



Step 2 Select the target PVC. If no PVC is available, click **Create PVC**. For details about related parameters, see **Creating a PVC**. Click **OK**.

Step 3 Set the container mount options.

- Set **Mount Path** to a path to which the data volume is mounted.

NOTICE

- The container path cannot be a system directory, such as **/** or /**var/run**. Otherwise, the container may not function normally. Select an empty directory. 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, and the container cannot start normally. As a result, the workload may not be deployed.
- If a volume is mounted to a high-risk directory, use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

- Set **Subpath** to a path of the data volume in the Kubernetes. It is the subpath of the volume instead of the root path. If this parameter is left blank, the root path is used.
- Set permissions.
  - **Read-only**: You can only read the data in the mounted volume.
  - **Read-write**: You can modify the volume mounted to the path. Newly written data will not be migrated if the container is migrated, which may cause data loss.

Step 4 You can add multiple PVCs.

----End

3.11.4 Creating a PVC

NOTICE

- After a PVC is created on the UCS console, a PVC with the same name is automatically created in your cluster. Also a PersistentVolume (PV) is created and bound with the PVC. If you are not familiar with the relationship among PVs, PVCs, and StorageClasses in Kubernetes, see **Persistent Storage**.
- You can modify or delete the PVCs automatically created by UCS on the cluster details page. However, if the PVC settings on the UCS console are not modified accordingly, the modified or deleted PVCs will be re-created by UCS. Therefore, you are advised to change the settings on the UCS console, not on the cluster details page.
- When a non-Huawei Cloud cluster uses a PVC to mount cloud storage, the cluster provider must support StorageClasses to dynamically create PVs. Run the following command to query the StorageClass configuration and the interconnected backend storage resources of the cluster. For more information about StorageClass, see **Storage Classes**.
```
kubectl get storageclass
```

Creating a PVC

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

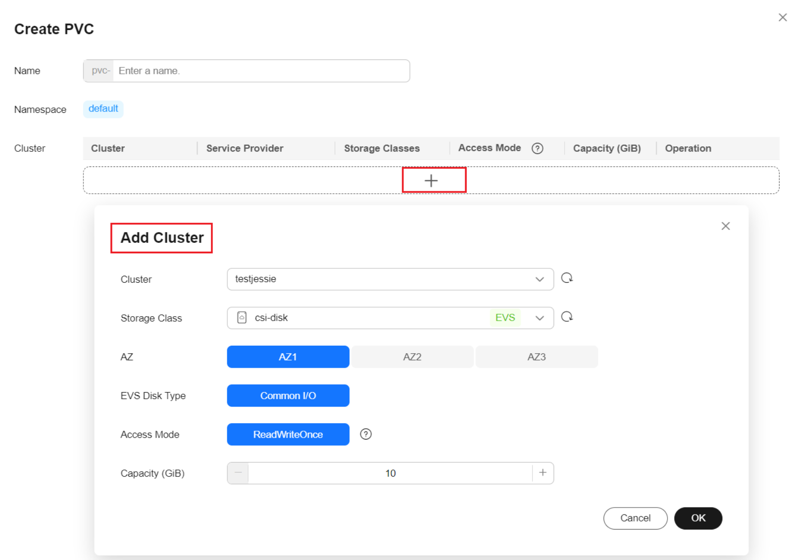
Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **Storage** in the navigation pane. On the **PersistentVolumeClaims (PVCs)** tab page, click **Create PVC** in the upper right corner.

Step 4 Specify basic information.

- **Name**: Enter a unique name of a PVC to be added.
- **Namespace**: namespace that the PVC will belong to. If this parameter is not specified, the default namespace is used.
- **Cluster**: Click + to select the cluster where the PVC is to be deployed.

Figure 3-42 Adding a cluster



  - For details about the parameters for adding a Huawei Cloud cluster, see **Table 3-44**.
  - For details about the parameters for adding a non-Huawei Cloud cluster, see **Table 3-45**.

Table 3-44 Adding a Huawei Cloud cluster

Parameter	Description
Cluster	Select a Huawei Cloud cluster.

Parameter	Description
Storage Class	• **csi-disk**: EVS disk. Specify the AZ and disk type. – **AZ**: Specify the AZ where the EVS disk is located. The supported EVS disk types may vary in different AZs. – **EVS Disk Type**: Available disk types are common I/O, high I/O, and ultra-high I/O, and the storage pools corresponding to the disk types are SATA, SAS, and SSD. • **csi-nas**: indicates SFS. • **csi-obs**: indicates OBS. You need to specify the instance type and object storage type, and add the access key. – **Instance Type**: an object bucket or a parallel file system. Parallel file system is a high-performance file system provided by OBS. It provides high-performance object-based access. – **OBS Class**: **Standard** and **Infrequent access** OBS buckets are supported. OBS Infrequent Access is highly reliable and cost-effective for real-time access. It is ideal for storing data that is semi-frequently accessed (less than 12 times a year). The application scenarios include file synchronization or sharing, and enterprise-level backup.
Access Mode	• If **csi-disk** is selected, **Access Mode** must be set to **ReadWriteOnce**, that is, the volume can be mounted as read-write by only a single node. • If **csi-nas** or **csi-obs** is selected, **Access Mode** must be set to **ReadWriteMany**, that is, the volume can be mounted as read-write by multiple nodes.
Capacity (GiB)	The capacity of the created PVC cannot be less than 10 GiB. Set this parameter only when **csi-disk** or **csi-nas** is selected. If **csi-obs** is selected, the capacity is used on demand and does not need to be set.

Table 3-45 Adding a non-Huawei Cloud cluster

Parameter	Description
Cluster	Select a non-Huawei Cloud cluster.
Storage Class	The storage classes supported by a cluster depend on the actual environment of the registered cluster. For details, see **Storage Classes**.

Parameter	Description
Access Mode	• **ReadWriteOnce** (RWO): The PVC can be mounted as read-write only by a single node. • **ReadWriteMany** (RWX): The PVC can be mounted as read-write by multiple nodes.
Capacity (GiB)	The capacity of the created PVC cannot be less than 10 GiB.
Annotation	Set key and value and click **Confirm**. Annotations are attached to PVCs in the form of key-value pairs.

Step 5 The key and value can be added repeatedly to configure differentiated settings for each cluster.

Step 6 Click **OK**. After the PVC is successfully created, you can click the PVC name to view the details.

----End

Related Operations

You can also perform operations described in **Table 3-46**.

Table 3-46 Related operations

Operation	Description
Creating a PVC from a YAML file	Click **Create from YAML** in the upper right corner to create a PVC from an existing YAML file.
Viewing details	1. Select the namespace to which the VPC will belong. 2. (Optional) Search for a PVC by its name. 3. Click the PVC name to view its details, including the basic information and deployment information of each cluster. 4. On the **PVC Details** page, click **View YAML** in the **Cluster** area to view or download YAML files of PVCs deployed in each cluster.
Viewing the YAML file	Click **View YAML** next to the PVC name to view the YAML file of the current PVC.
Update (Expanding a PVC)	1. Choose **More** > **Update** in the row where the target PVC resides. 2. Modify the cluster deployment parameters based on the **PVC parameters**, or click **Expand** to expand the PVC. 3. Click **OK** to submit the modified information.

Operation	Description
Deleting a PVC	Choose **More** > **Delete** in the row where the target PVC resides, and click **Yes**.
Deleting PVCs in batches	1. Select PVCs to be deleted. 2. Click **Delete** in the upper left corner. 3. Click **Yes**.

3.12 Namespaces

A namespace is an abstract integration of a group of resources and objects in a cluster. Namespace-level resource quotas limit the amount of resources available to teams or projects that use the same cluster.

Creating a Namespace

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **Namespaces** in the navigation pane and click **Create Namespace** in the upper right corner.

Step 4 Set namespace parameters based on **Table 3-47**.

Table 3-47 Parameters for creating a namespace

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

Step 5 When the configuration is complete, click **OK**.

After the creation is complete, you can click **View YAML** to view and download the YAML file.

----End

Using Namespaces

Namespaces can be used when creating Services, Ingresses, and PVCs. The following uses workload creation as an example to describe how a namespace is used.

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 In the navigation pane on the left, choose **Workloads**. On the **Deployments** tab, click **Create from Image** in the upper right corner.

Step 4 Configure the basic information about the workload and select the namespace where the workload is located.

Step 5 Complete the configuration.

----End

Deleting a Namespace

NOTICE

- Deleting a namespace on the UCS console will delete the namespace with the same name in each cluster as well as all data resources related to the namespace. Exercise caution when performing this operation.
- To ensure that UCS runs properly, namespaces whose source is **System** or **Default** cannot be deleted.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

Step 2 On the **Fleets** tab, click the name of the federation-enabled fleet to access its details page.

Step 3 Choose **Namespaces** in the navigation pane. In the namespace list, click **Delete** in the row of the target namespace.

To delete multiple namespaces at a time, select the namespaces and click **Delete** in the upper left corner.

Step 4 Click **Yes** as prompted.

----End

3.13 Multi-Cluster Workload Scaling

3.13.1 Overview

Why Workload Scaling?

The ever-changing application traffic brings changing resource requirements to container workloads. During workload deployment and management, if resources are reserved for a workload based on the service requirements at peak hours, a large number of resources will be wasted. If a resource threshold is set for a workload, applications may be abnormal when the resource usage exceeds the threshold. In Kubernetes, a **Horizontal Pod Autoscaler (HPA)** can automatically scale in or out pods for workloads in a single cluster in response to metric changes. However, the HPA does not apply to multi-cluster scenarios.

UCS provides you with automatic workload scaling in multi-cluster scenarios. The automatic workload scaling is based on metric changes or at regular intervals, which raises scaling flexibility and stability.

Advantages

UCS workload scaling has the following advantages:

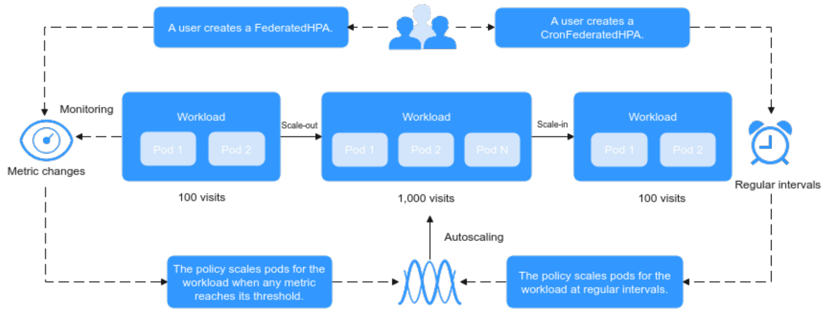
- Multi-cluster: You can configure the same scaling policy for multiple clusters in the federation.
- High availability: Pods in your workload can be quickly scaled out at peak hours to ensure workload availability, or scaled in at off-peak hours to save resources.
- Multi-function: Pods in your workload can be scaled in or out based on metric changes or at regular intervals in complex scenarios.
- Multi-scenario: You can configure scaling policies for online services, large-scale computing and training, and training and inference on deep learning GPUs or shared GPUs.

Working Principles

UCS workload scaling is implemented by FederatedHPA and CronFederatedHPA, as shown in **Figure 3-43**.

- FederatedHPA can automatically scale in or out pods for workloads in response to system metrics or custom metrics. When the metric reaches the desired value, workload scaling is triggered.
- CronFederatedHPA can automatically scale in or out pods for workloads at regular intervals. When the triggering time arrives, workload scaling is triggered.

Figure 3-43 Working principles of workload scaling



Constraints

- UCS scaling policies apply only to Deployments. For details about the comparisons among different types of workloads, see Workloads.

- UCS scaling policies are used to scale in or out pods for workloads. To schedule the pods to specific clusters, you need to configure scheduling policies.

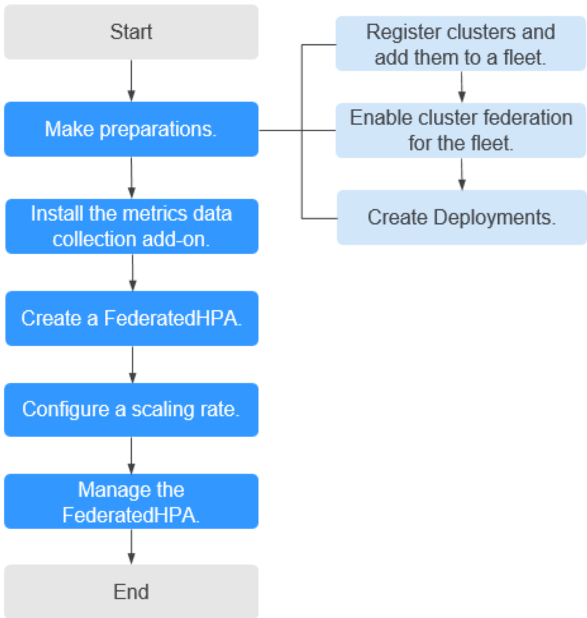
3.13.2 Using Scaling Policies

This section describes how to use FederatedHPA and CronFederatedHPA.

Using FederatedHPA

Figure 3-44 shows how to use FederatedHPA.

Figure 3-44 Using FederatedHPA



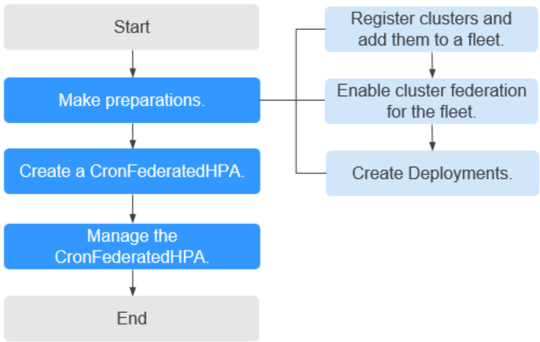
1. Add clusters to a fleet, enable cluster federation for the fleet, and create Deployments. (Workload scaling is based on workloads deployed in multiple clusters.) For details, see Registering a Cluster, Enabling Cluster Federation, and Creating a Workload.
2. Install the metrics data collection add-on for clusters. For details, see **Installing a Metric Collection Add-on**.
3. Create a FederatedHPA. For details, see **Creating a FederatedHPA to Scale Pods Based on Metric Changes**.
4. Configure a scaling rate. For details, see **Configuring a FederatedHPA to Control the Scaling Rate**.
5. Modify or delete the FederatedHPA. For details, see **Managing a FederatedHPA**.

Using CronFederatedHPA

Using CronFederatedHPA Separately

Figure 3-45 shows how to use CronFederatedHPA separately.

Figure 3-45 Process of using CronFederatedHPA separately

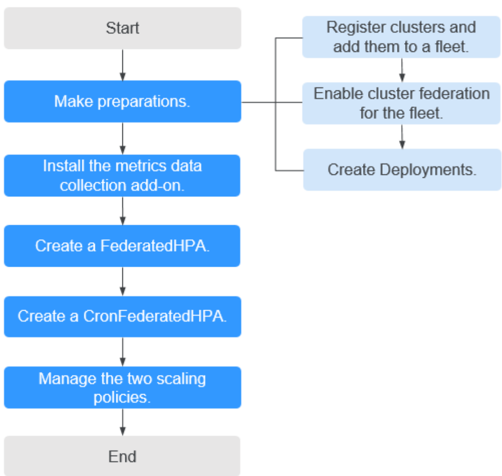


1. Add clusters to a fleet, enable cluster federation for the fleet, and create Deployments. (Workload scaling is based on workloads deployed in multiple clusters.) For details, see Registering a Cluster, Enabling Cluster Federation, and Creating a Workload.
2. Create a CronFederatedHPA. For details, see **Creating a CronFederatedHPA to Scale Pods at Regular Intervals**.
3. Modify or delete the CronFederatedHPA. For details, see **Managing a CronFederatedHPA**.

Using CronFederatedHPA and FederatedHPA Together

Figure 3-46 shows how to use CronFederatedHPA and FederatedHPA together.

Figure 3-46 Process of using CronFederatedHPA and FederatedHPA together



1. Add clusters to a fleet, enable cluster federation for the fleet, and create Deployments. (Workload scaling is based on workloads deployed in multiple clusters.) For details, see Registering a Cluster, Enabling Cluster Federation, and Creating a Workload.
2. Install the metrics data collection add-on for clusters. For details, see **Installing a Metric Collection Add-on**.
3. Create a FederatedHPA. For details, see **Creating a FederatedHPA to Scale Pods Based on Metric Changes**.

4. Create a CronFederatedHPA. For details, see **Creating a CronFederatedHPA to Scale Pods at Regular Intervals**.
5. Modify or delete the two scaling policies. For details, see **Managing a FederatedHPA** and **Managing a CronFederatedHPA**.

3.13.3 FederatedHPA

3.13.3.1 How FederatedHPA Works

FederatedHPA can automatically scale in or out pods for workloads in response to system metrics (CPU usage and memory usage) or custom metrics.

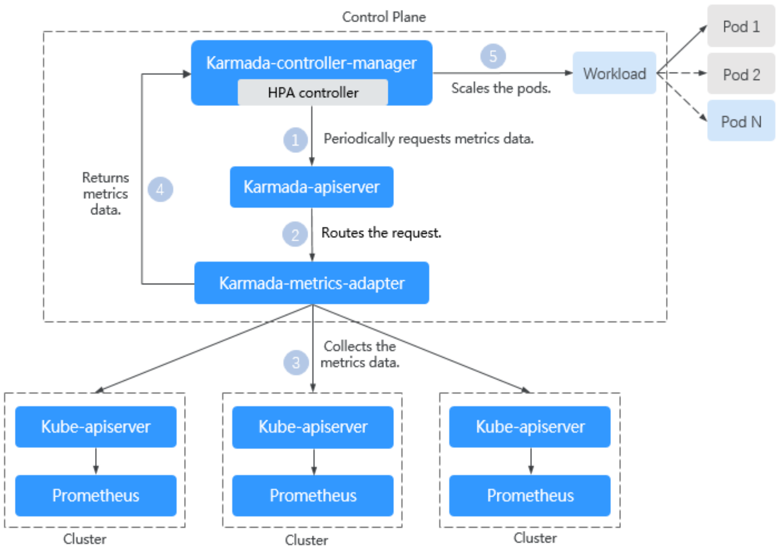
FederatedHPAs and scheduling policies can be used together to implement various functions. For example, after a FederatedHPA scales out pods in your workload, you can configure a scheduling policy to schedule the pods to clusters with more resources. This solves the resource limitation of a single cluster and improves the fault recovery capability.

How FederatedHPA Works

Figure 3-47 shows the working principle of FederatedHPA. The details are as follows:

1. The HPA controller periodically requests metrics data of a workload from either the system metrics API or the custom metrics API.
2. After receiving the metric query request, karmada-apiserver routes the request to karmada-metrics-adapter that was registered through its API.
3. After receiving the request, karmada-metrics-adapter collects the metrics data of the workload.
4. karmada-metrics-adapter returns **calculated metrics data** to the HPA controller.
5. The HPA controller **calculates the desired number of pods** based on the returned metrics data and maintains the **stability of workload scaling**.

Figure 3-47 Working principle of FederatedHPA



How Do I Calculate Metrics Data?

There are system metrics and custom metrics. Their calculation methods are as follows:

- System metrics

  There are two types of system metrics: CPU usage and memory usage. The system metrics can be queried and monitored through metrics API. For example, if you want to control the CPU usage of a workload at a reasonable level, you can create a FederatedHPA for the workload based on the CPU usage metric.

  NOTE

  Usage = CPUs or memory used by pods in a workload/Requested CPUs or memory

- Custom metrics

  You can create a FederatedHPA for a workload based on custom metrics such as requests per second and writes per second. The HPA controller then queries for these custom metrics from a series of APIs.

If you set multiple desired metric values when creating a FederatedHPA, the HPA controller evaluates each metric separately and uses the scaling algorithm to determine the new workload scale based on each one. The largest scale is selected for the autoscale operation.

How Do I Calculate the Desired Number of Pods?

The HPA controller operates on the scaling ratio between the desired metric value and current metric value and then uses that ratio to calculate the desired number of pods based on the current number of pods.

- Current number of pods = Number of pods in the **Ready** state in all clusters

When calculating the desired number of pods, the HPA controller chooses the largest recommendation based on the last five minutes to prevent subsequent autoscaling operations before the workload finishes responding to prior autoscaling operations.

- Desired number of pods = Current number of pods x (Current metric value/ Desired metric value)

For example, if the current CPU usage is 100% and the desired CPU usage is 50%, the desired number of pods is twice the current number of pods.

How Do I Ensure the Stability of Workload Scaling?

To ensure the stability of workload scaling, the HPA controller is designed to provide the following functions:

- Stabilization window

  When detecting that the metric data reaches the desired value (the scaling standard is met), the HPA controller continuously checks the metric data within stabilization window. If the result shows that the metric data continuously reaches the desired value, the HPA controller performs scaling. By default, the stabilization window is 0 seconds for a scale-out and 300 seconds for a scale-in. The values can be changed. In actual configuration, to avoid service jitter, a scale-out needs to be fast, and a scale-in needs to be slow.

- Tolerance

  Tolerance = abs (Current metric value/Desired metric value – 1)

  abs indicates an absolute value. If the metric value change is within the specified tolerance range, the scaling operation will not be triggered. The default value is 0.1 and cannot be changed.

For example, if you select the default settings when creating a FederatedHPA, a scale-in will be triggered when the metric value is more than 1.1 times the desired value and lasts for more than 300 seconds, and a scale-out will be triggered when the metric value is less than 0.9 times the desired value and lasts for more than 0 seconds.

3.13.3.2 Installing a Metric Collection Add-on

Before creating a FederatedHPA, you need to install the add-on that supports metrics APIs for a cluster to collect workload metrics. If you have installed the add-on, skip this step.

Selecting an Add-on

UCS provides two types of add-ons: Kubernetes Metrics Server and kube-prometheus-stack. The add-ons apply to different cluster and metric types. For details about how to select an add-on, see **Table 3-48**.

Table 3-48 Add-ons that provide metrics APIs

Applicable Cluster Type	Supported Metric Type	Add-on	Precautions
Huawei Cloud clusters	System metrics	Install Kubernetes Metrics Server or kube-prometheus-stack.	After installing kube-prometheus-stack, you need to register it as a service that provides the metrics API. For details, see **Providing Resource Metrics Through the Metrics API**.

Applicable Cluster Type	Supported Metric Type	Add-on	Precautions
	Custom metrics	Install kube-prometheus-stack.	• Before installing this add-on, check your Huawei Cloud cluster version. If the version is earlier than v1.19, upgrade the cluster version first. • When installing this add-on, you must select the server mode. In this mode, you can customize metrics. • After installing this add-on, aggregate custom metrics to the Kubernetes API server. For details, see **Aggregating Custom Metrics to the Kubernetes API Server**.
Non-Huawei Cloud clusters	System metrics	Install Kubernetes Metrics Server.	For details, see **Installing an Add-on**.
	Custom metrics	No add-on available.	To collect custom metrics of a non-Huawei Cloud cluster, you need to install **Prometheus Adapter** and configure a custom metric collection rule. Then create a FederatedHPA.

Installing an Add-on

After selecting an applicable add-on, install it for the cluster by referring to the precautions in **Table 3-48** and related documents.

CAUTION

Install the metric collection add-on for all clusters in a federation for which a scaling policy needs to be created. Otherwise, metric collection will be abnormal and the scaling policy will become invalid.

- For details about how to install Kubernetes Metrics Server, see **Kubernetes Metrics Server**.
- For details about how to install kube-prometheus-stack, see **kube-prometheus-stack**.

3.13.3.3 Creating a FederatedHPA to Scale Pods Based on Metric Changes

This section describes how you can create a FederatedHPA so that pods in workloads are automatically scaled in or out based on different metrics.

Before creating a FederatedHPA, you must have learnt the basic working principle and concepts of FederatedHPA (**How FederatedHPA Works**). To know the differences between the FederatedHPA and CronFederatedHPA, see **Overview**.

Constraints

FederatedHPA can be configured only for clusters 1.19 or later. To query the cluster version, log in to the UCS console, click the name of the fleet that the cluster is added to, and click **Container Clusters**.

Creating a FederatedHPA

Using the console

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Metric-based Policy** tab. Then click **Create Metric-based Policy** in the upper right corner.

Step 4 Configure parameters for the FederatedHPA.

Table 3-49 FederatedHPA parameters

Parameter	Description
Policy Name	Enter a name containing 4 to 63 characters for the FederatedHPA.
Namespace	Select the namespace for the workload for which you want to set automatic scaling. You can also create a namespace. For details, see Namespaces.
Applicable Workload	Select the name of the workload for which you want to set automatic scaling. You can also create a workload. For details, see Workloads.
Pod Range	Enter the minimum and maximum numbers of pods. When the FederatedHPA is triggered, the pods will be scaled within this range. ● **Min.**: Enter an integer ranging from 1 to 299. ● **Max.**: Enter a positive integer ranging from 1 to 1,500. The value must be greater than the minimum value.

Parameter	Description
Stabilization Window	The scaling operation is initiated only when the metric continuously reaches the desired value within stabilization window. By default, the stabilization window is 0 seconds for a scale-out and 300 seconds for a scale-in. For details about stabilization window, see **How Do I Ensure the Stability of Workload Scaling?**. ● **Scale-out**: Enter an integer ranging from 0 to 3,600, in seconds. ● **Scale-in**: Enter an integer ranging from 0 to 3,600, in seconds.
System rule	If you want to scale pods for a workload based on system metrics, you need to configure this rule. ● **Metric Name**: Select **CPU utilization** or **Memory utilization**. ● **Expected Value**: The scaling operation is triggered when the metric reaches the desired value.
Custom rule	If you want to scale pods for a workload based on custom metrics, you need to configure this rule. ● **Metric Name**: Select a name from the drop-down list. ● **Source**: Select the object type described by the custom metric from the drop-down list. Currently, only **Pod** is supported. ● **Expected Value**: The scaling operation is triggered when the metric reaches the desired value. CAUTION ● Custom rules can be created only for clusters 1.19 or later. ● Before using a custom rule, install the add-on that supports custom metric collection for the cluster. Ensure that the add-on can collect and report the custom metrics of the workloads. For details, see **Installing a Metric Collection Add-on**.

Step 5 Click **Create** in the lower right corner.

In the displayed policy list, you can view the policy details.

----End

Using kubectl

Step 1 Use kubectl to connect to the federation. For details, see Using kubectl to Connect to a Federation.

Step 2 Create and edit an **fhpa.yaml** file. For details about the key parameters, see **Table 3-50**.

vi fhpa.yaml

In this example, the FederatedHPA is named **hpa-example-hpa** and associated with the workload named **hpa-example**. The stabilization window is 0 seconds for a scale-out and 300 seconds for a scale-in. The maximum number of pods is 100 and the minimum number of pods is 2. There are two system metric rules: **memory** and **cpu**. The desired memory usage in **memory** is 50%, and the desired CPU usage in **cpu** is 60%.

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: FederatedHPA
metadata:
  name: hpa-example-hpa                        # FederatedHPA name
  namespace: default                       # Namespace where the workload resides
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-example                          # Workload name
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300                # The stabilization window is 300 seconds for a scale-in.
    scaleUp:
      stabilizationWindowSeconds: 0                # The stabilization window is 0 seconds for a scale-out.
  minReplicas: 2                           # The minimum number of pods is 2.
  maxReplicas: 100                          # The maximum number of pods is 100.
  metrics:
    - type: Resource
      resource:
        name: memory                       # Name of the first rule
        target:
          type: Utilization                # The metric type is resource usage.
          averageUtilization: 50              # Desired average resource usage
    - type: Resource
      resource:
        name: cpu                        # Name of the second rule
        target:
          type: Utilization                # The metric type is resource usage.
          averageUtilization: 60              # Desired average resource usage
```

Table 3-50 Key parameters

Parameter	Mandatory	Type	Description
stabilizationWindowSeconds	No	String	Stabilization window for a scale-in. Enter a positive integer ranging from 0 to 3,600, in seconds. If this parameter is not specified, the default value is **300**. **NOTE** The scaling operation is initiated only when the metric continuously reaches the desired value within stabilization window. For details about stabilization window, see **How Do I Ensure the Stability of Workload Scaling?**.
stabilizationWindowSeconds	No	String	Stabilization window for a scale-out. Enter a positive integer ranging from 0 to 3,600, in seconds. If this parameter is not specified, the default value is **0**.
minReplicas	Yes	String	Minimum number of pods that can be scaled in for a workload when the scaling policy is triggered. Enter a positive integer ranging from 1 to 299.

Parameter	Mandatory	Type	Description
maxReplicas	Yes	String	Maximum number of pods that can be scaled out for a workload when the scaling policy is triggered. Enter a positive integer ranging from 1 to 1,500 and ensure that the value is greater than the minimum value.
name	Yes	String	Rule name. For scaling operations based on system metrics, **memory** is used as the rule name of the memory usage, and **cpu** is used as the rule name of the CPU usage.
type	Yes	String	Metric type. ● **Value**: total number of pods ● **AverageValue**: Total number of pods/ Number of current pods ● **Utilization**: CPUs or memory used by pods in a workload/Requested CPUs or memory
averageUtilization	Yes	String	The scaling operation is triggered when the metric reaches the desired value.

Step 3 Create a FederatedHPA.

kubectl apply -f fhpa.yaml

If information similar to the following is displayed, the policy has been created:

```
FederatedHPA.autoscaling.karmada.io/hpa-example-hpa created
```

You can run the following commands to check the workload scaling:

- **kubectl get deployments**: checks the current number of pods in a workload.
- **kubectl describe federatedhpa hpa-example-hpa**: views scaling events (latest three records) of the FederatedHPA.

You can run the following commands to manage FederatedHPA **hpa-example-hpa** (replaced with the actual name):

- **kubectl get federatedhpa hpa-example-hpa**: obtains the FederatedHPA.
- **kubectl edit federatedhpa hpa-example-hpa**: updates the FederatedHPA.
- **kubectl delete federatedhpa hpa-example-hpa**: deletes the FederatedHPA.

----End

3.13.3.4 Configuring a FederatedHPA to Control the Scaling Rate

Why Do I Need to Control the Scaling Rate?

To limit the rate at which pods are scaled by the HPA controller, a scale-out needs to be fast, and a scale-in needs to be slow. However, if only the stabilization window is configured, the scaling rate cannot be limited after the stabilization window expires. To accurately and flexibly limit the scaling rate, you can configure the behavior section of the spec in the YAML file. In the behavior section, the scaling rate can be unique for each FederatedHPA, and different rates can be configured for scale-out and scale-in operations.

Procedure

The following describes behavior structures in common service scenarios. In other service scenarios, for example, if you want to perform slow scale-out or fast scale-in, you can set **scaleUp** and **scaleDown** under the **behavior** field by referring to the following description of each behavior structure.

- Scenario 1: Fast scale-out

  If you want to perform a fast scale-out at peak hours, you can set **Percent** to a large value.

  ```
  behavior:
    scaleUp:
      policies:
      - type: Percent
        value: 900
        periodSeconds: 60
  ```

  In this example, the value of **Percent** is **900**. This means the scaling rate increases tenfold (1 + 900%) in each scaling period. For example, if the number of pods in a workload starts from 1, the number of pods added every 60 seconds changes as follows: 1 > 10 > 100 > ... . Note that the number of pods after a scale-out cannot exceed the maximum number of pods configured in the FederatedHPA.

  Resource consumption of the Percent type fluctuates greatly. If you want the resource consumption to be controllable, use the Pods type with the absolute value.

  ```
  behavior:
    scaleDown:
      policies:
      - type: Pods
        value: 10
        periodSeconds: 60
  ```

  In this example, the value of **Pods** is **10**. This means 10 pods are added in each scaling period. For example, if the number of pods in a workload starts from 1, the number of pods added every 60 seconds changes as follows: 1 > 11 > 21 > ... . Note that the number of pods after a scale-out cannot exceed the maximum number of pods configured in the FederatedHPA.

- Scenario 2: Slow scale-in

  If you want to scale in pods in a workload more slowly after peak hours to improve application reliability, you can set **Pods** to a small value and **periodSeconds** to a large value.

  ```
  behavior:
    scaleDown:
  ```

```
  policies:
  - type: Pods
    value: 1
    periodSeconds: 600
```

In this example, the value of **Pods** is **1**, and the value of **periodSeconds** is **600**. This means the scale-in period is 600 seconds, and one pod is reduced for each scale-in. If the initial number of pods is 100, the number of pods to be scaled in every 600 seconds changes as follows: 100 > 99 > 98 > ... . In extreme cases, if you do not want the pods in a workload to be automatically scaled in, you can set **Percent** or **Pods** to **0**.

- Scenario 3: Default scaling rate

  If **behavior** is not configured, the default settings of the FederatedHPA are as follows:

```
behavior:
  scaleDown:
    stabilizationWindowSeconds: 300
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
  scaleUp:
    stabilizationWindowSeconds: 0
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
    - type: Pods
      value: 4
      periodSeconds: 15
```

  In the default configuration, the scaling period is 15 seconds. In each scaling period, a scale-out or scale-in is performed at a rate of twice (1 + 100%). 4 pods to be scaled each time.

3.13.3.5 Managing a FederatedHPA

This section describes how you can modify and delete a FederatedHPA.

CAUTION

If you modify or delete a FederatedHPA during workload scaling, the modification or deletion will take effect immediately.

Modifying a FederatedHPA

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Metric-based Policy** tab. Locate the policy and click **Edit** in the **Operation** column. Then modify the policy settings. For details about the parameters, see **Table 3-50**.

Step 4 Click **OK**.

----End

Deleting a FederatedHPA

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Metric-based Policy** tab. Select the policy you want to delete and choose **More** > **Delete** in the **Operation** column. If you want to delete multiple policies in batches, click **Delete** in the upper left. In the displayed dialog box, click **Yes**.

----End

3.13.4 CronFederatedHPA

3.13.4.1 How CronFederatedHPA Works

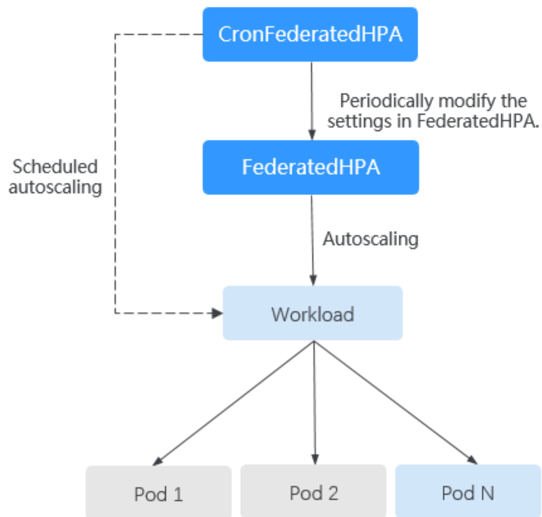
CronFederatedHPA is needed because FederatedHPA can only scale in or out pods for workloads based on metrics data. However, metric-based scaling brings in latency. CronFederatedHPA can automatically scale in or out pods for workloads at regular intervals.

You can configure a CronFederatedHPA for workloads whose resource usage changes periodically, so that pods can be added before predicated peak hours and reclaimed at off-peak hours.

How CronFederatedHPA Works

Figure 3-48 shows the working principle of CronFederatedHPA. When creating a CronFederatedHPA, you can specify a time to adjust the maximum and minimum numbers of pods in a FederatedHPA or directly specify the number of pods desired.

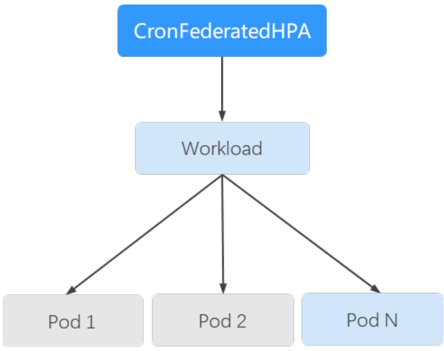
Figure 3-48 Working principle of CronFederatedHPA



Using CronFederatedHPA Separately

If CronFederatedHPA is separately used, it periodically adjusts the number of pods for workloads. After you set the effective time and desired number of pods in a CronFederatedHPA, pods will be periodically scaled after the CronFederatedHPA is in effect.

Figure 3-49 Using CronFederatedHPA separately



The detailed procedure is as follows:

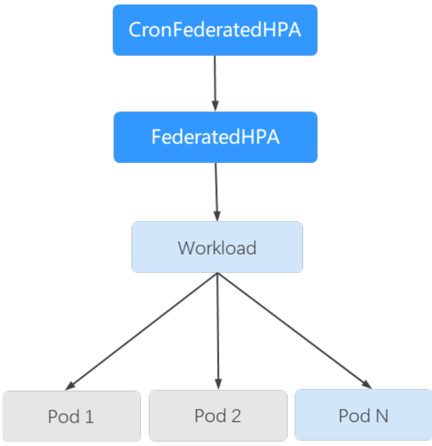
1. Create a CronFederatedHPA and set the effective time and desired number of pods.
   - Effective time: the time when the CronFederatedHPA takes effect.
   - Desired number of pods: the desired number of pods when the CronFederatedHPA takes effect.
2. When the CronFederatedHPA takes effect, the **number of existing pods** in the workload will be compared with the **desired number of pods** set in **1**. If the desired number is greater, pods are scaled out for the workload. If the desired number is smaller, pods are scaled in.

   Number of existing pods: the number of pods in the workload before the CronFederatedHPA takes effect.

Using Both CronFederatedHPA and FederatedHPA

If both FederatedHPA and CronFederatedHPA are used, CronFederatedHPA runs based on FederatedHPA and periodically adjusts the maximum and minimum numbers of pods in the FederatedHPA for scheduled scaling.

Figure 3-50 Using both FederatedHPA and CronFederatedHPA



The detailed procedure is as follows:

1. Create a CronFederatedHPA and set the effective time and desired number of pods.
   - Effective time: the time when the CronFederatedHPA takes effect.
   - Desired number of pods: the number of pods set in the CronFederatedHPA. When CronFederatedHPA takes effect, this number will be used as a reference for adjusting the maximum and minimum numbers of pods in the FederatedHPA. The maximum and minimum numbers can be used as starting points for adjusting the number of pods for a workload.
2. When the CronFederatedHPA takes effect, the **number of existing pods** of the workload, **maximum number of pods** and **minimum number of pods** in the FederatedHPA, and **desired number of pods** set in **1** will be compared to determine how much the maximum and minimum numbers of pods in the FederatedHPA will be adjusted. Then, the FederatedHPA scales in or out pods for the workload based on the adjusted maximum and minimum numbers of pods.
   - Number of existing pods: the number of pods in the workload before the CronFederatedHPA takes effect.
   - Maximum number of pods in the FederatedHPA: the maximum number of pods for a workload.
   - Minimum number of pods in the FederatedHPA: the minimum number of pods for a workload.

Figure 3-51 and **Table 3-51** show the possible scaling scenarios when both FederatedHPA and CronFederatedHPA are used. You can learn about how CronFederatedHPA takes effect on the FederatedHPA and workload based on the number of existing pods, maximum number of pods, minimum number of pods, and desired number of pods.

Figure 3-51 Scaling scenarios when both policies are used

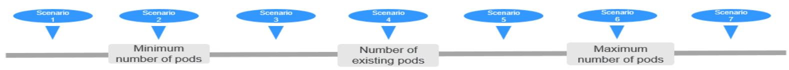


Table 3-51 Scaling scenarios when both policies are used

Scenario No.	Description	Desired Number of Pods (in a CronFederatedHPA)	Number of Existing Pods (in a Workload)	Minimum/Maximum Number of Pods (in a FederatedHPA)	Result
1	**Desired number of pods** < Minimum number of pods ≤ Number of existing pods ≤ Maximum number of pods	3	5	4/10	• The minimum number of pods in the FederatedHPA is changed to 3. • The number of existing pods of the workload is not changed.
2	**Desired number of pods** = Minimum number of pods ≤ Number of existing pods ≤ Maximum number of pods	4	5	4/10	• The minimum number of pods in the FederatedHPA is not changed. • The number of existing pods of the workload is not changed.
3	Minimum number of pods < **Desired number of pods** < Number of existing pods ≤ Maximum number of pods	5	6	4/10	• The minimum number of pods in the FederatedHPA is changed to 5. • The number of existing pods of the workload is not changed.

Scenario No.	Description	Desired Number of Pods (in a CronFederatedHPA)	Number of Existing Pods (in a Workload)	Minimum/ Maximum Number of Pods (in a FederatedHPA)	Result
4	Minimum number of pods < **Desired number of pods** = Number of existing pods ≤ Maximum number of pods	5	5	4/10	• The minimum number of pods in the FederatedHPA is changed to 5. • The number of existing pods of the workload is not changed.
5	Minimum number of pods ≤ Number of existing pods < **Desired number of pods** < Maximum number of pods	6	5	4/10	• The minimum number of pods in the FederatedHPA is changed to 6. • The number of existing pods of the workload is changed to 6.
6	Minimum number of pods ≤ Number of existing pods < **Desired number of pods** = Maximum number of pods	10	4	4/10	• The minimum number of pods in the FederatedHPA is changed to 10. • The number of existing pods of the workload is changed to 10.

Scenario No.	Description	Desired Number of Pods (in a CronFederatedHPA)	Number of Existing Pods (in a Workload)	Minimum/ Maximum Number of Pods (in a FederatedHPA)	Result
7	Minimum number of pods ≤ Number of existing pods ≤ Maximum number of pods < **Desired number of pods**	11	4	4/10	● The minimum and maximum numbers of pods in the FederatedHPA are both changed to 11. ● The number of existing pods of the workload is changed to 11.

3.13.4.2 Creating a CronFederatedHPA to Scale Pods at Regular Intervals

This section describes how you can create a CronFederatedHPA so that pods in workloads are automatically scaled in or out at regular intervals.

Before creating a CronFederatedHPA, you must have learnt the basic working principle and concepts of CronFederatedHPA (**How CronFederatedHPA Works**). To know the differences between the FederatedHPA and CronFederatedHPA, see **Overview**.

Constraints

CronFederatedHPA can be configured only for clusters 1.19 or later.

Creating a CronFederatedHPA

Using the console

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Scheduled Policies** tab. Then click **Create Scheduled Policy** in the upper right corner.

Step 4 Configure parameters for the CronFederatedHPA by referring to **Table 3-52**.

Table 3-52 Basic parameters

Parameter	Description
Policy Name	Enter a name for the CronFederatedHPA.
Namespace	Select the namespace for the workload for which you want to configure automatic scaling.
Object	Select **Workloads** or **Metric-based Policy**. ● **Workloads**: Select or create a workload you will associate the policy with. For details, see **Creating a Workload**. ● **Metric-based Policy**: Select an existing metric-based policy or click **Create Metric-based Policy** on the right to create one. For details, see **Creating a FederatedHPA**.

Step 5 Click **Add Rule** in **Policy Settings**. In the displayed dialog box, configure parameters by referring to **Table 3-53**.

Table 3-53 Parameters for adding a rule

Parameter	Description
Rule Name	Enter a name for the CronFederatedHPA.
Expected Copies	Enter the desired number of pods scaled when the CronFederatedHPA is triggered.
Triggered	Select **Hourly**, **Daily**, **Weekly**, **Monthly**, **Yearly**, or **Cron**. ● **Hourly**: a specific minute in an hour when the policy is executed. For example, if you select **5**, the policy is executed at the fifth minute of every hour. ● **Daily**: a specific minute every day when the policy is executed. ● **Weekly**: a specific minute on a day of each week when the policy is executed. ● **Monthly**: a specific minute on a day of each month when the policy is executed. ● **Yearly**: a specific minute on a day of a month in each year when the policy is executed. ● **Cron**: Cron expression syntax: Minute (0 to 59) Hour (0 to 23) A day in a month (1 to 31) Month (1 to 12) A day in a week (0 to 6) * * * * * For example, **0 0 13 * 5** indicates that a task is started at 00:00 on every Friday and the 13th day of each month.

Parameter	Description
Time Zone	Select Shanghai or Singapore.

Step 6 Click **OK** and then click **Create**.

In the displayed policy list, you can view the policy details.

----End

Using kubectl

Step 1 Use kubectl to connect to the federation. For details, see Using kubectl to Connect to a Federation.

Step 2 Create and edit a **cfhpa.yaml** file.

vi cfhpa.yaml

For details about the parameters in this file, see **Table 3-54**. In this example, the CronFederatedHPA named **cron-federated-hpa** is used for the **test** workload and contains two rules (**Scale-Up** and **Scale-Down**) for scheduled scaling. **Scale-Up** specifies that 10 pods are scaled out at 8:30 daily, and **Scale-Down** specifies that 5 pods are scaled in at 21:00 daily.

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: CronFederatedHPA
metadata:
  name: cron-federated-hpa          # CronFederatedHPA name
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment               # Select Deployment or FederatedHPA.
    name: test                    # Name of the workload or FederatedHPA
  rules:
  - name: "Scale-Up"               # Rule name
    schedule: 30 08 * * *          # Time when the policy is triggered
    targetReplicas: 10             # Desired number of pods, which is a non-negative integer
    timeZone: Asia/Shanghai          # Time zone
  - name: "Scale-Down"              # Rule name
    schedule: 0 21 * * *           # Time when the policy is triggered
    targetReplicas: 5              # Desired number of pods, which is a non-negative integer
    timeZone: Asia/Shanghai          # Time zone
```

Table 3-54 Key parameters

Parameter	Mandatory	Type	Description
kind	Yes	String	Select **Deployment** or **FederatedHPA**. ● **Deployment**: The CronFederatedHPA is used separately. ● **FederatedHPA**: Both FederatedHPA and CronFederatedHPA are used.
name	Yes	String	Enter the rule name of 1 to 32 characters in the CronFederatedHPA.

Parameter	Mandatory	Type	Description
schedule	Yes	String	Time when the policy is triggered. Cron expression syntax: Minute (0 to 59) Hour (0 to 23) A day in a month (1 to 31) Month (1 to 12) A day in a week (0 to 6) * * * * * For example, **0 0 13 * 5** indicates that a task is started at 00:00 on every Friday and the 13th day of each month.
targetReplicas	Yes	String	Enter the desired number of pods scaled when the CronFederatedHPA is triggered.
timeZone	Yes	String	Select Shanghai or Singapore. ● Shanghai: Asia/Shanghai ● Singapore: Asia/Singapore

Step 3 Create a CronFederatedHPA.

kubectl apply -f cfhpa.yaml

If information similar to the following is displayed, the policy has been created:

```
CronFederatedHPA.autoscaling.karmada.io/cron-federated-hpa created
```

You can run the following commands to check the workload scaling:

- **kubectl get deployments**: checks the current number of pods in a workload.
- **kubectl describe cronfederatedhpa cron-federated-hpa**: views scaling events (latest three records) of the CronFederatedHPA.

You can run the following commands to manage CronFederatedHPA **cron-federated-hpa** (replaced with the actual name):

- **kubectl get cronfederatedhpa cron-federated-hpa**: obtains the CronFederatedHPA.
- **kubectl edit cronfederatedhpa cron-federated-hpa**: updates the CronFederatedHPA.
- **kubectl delete cronfederatedhpa cron-federated-hpa**: deletes the CronFederatedHPA.

----End

3.13.4.3 Managing a CronFederatedHPA

This section describes how you can modify and delete a CronFederatedHPA.

Modifying a CronFederatedHPA

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Scheduled Policies** tab. Locate the policy and click **Edit** in the **Operation** column. Then delete or add a policy rule.

- To delete a policy rule, click **Delete** next to the rule.
- To add a rule, click **Add Rule** in **Policy Settings**. In the displayed dialog box, configure parameters and click **OK**. For details about the parameters, see **Table 3-54**.

Step 4 Click **OK**.

----End

Deleting a CronFederatedHPA

Step 1 Log in to the UCS console and choose **Fleets** in the navigation pane.

Step 2 Click the name of the fleet with federation enabled.

Step 3 Choose **Workload Scaling** in the navigation pane and click the **Scheduled Policies** tab. Select the policy you want to delete and choose **More** > **Delete** in the **Operation** column. If you want to delete multiple policies in batches, click **Delete** in the upper left. In the displayed dialog box, click **Yes**.

----End

3.14 Adding Labels and Taints to a Cluster

UCS allows you to add different labels to clusters to define clusters. By using these cluster labels, you can quickly understand the characteristics of each cluster. Taints enable a cluster to repel specific pods to prevent these pods from being scheduled to the cluster, achieving reasonable allocation of workloads on clusters.

Labels

You can add different labels to clusters to classify and manage clusters.

Taints

Taints are in the format of **Key=Value:Effect**. **Key** and **Value** are the labels of a taint. **Value** can be empty. **Effect** is used to describe the effect of taints. The following two options are supported for **Effect**:

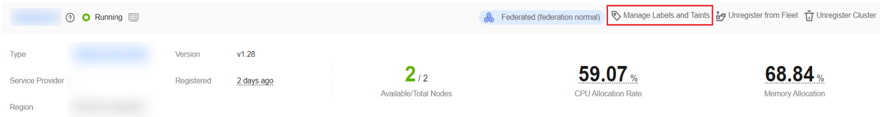
- **NoSchedule**: No pod will be able to schedule onto the cluster unless it has a matching toleration, but existing pods will not be evicted from the cluster.
- **NoExecute**: Pods that cannot tolerate this taint cannot be scheduled onto the cluster, and existing pods will be evicted from the cluster.

Managing Cluster Labels and Taints

Step 1 Log in to the UCS console.

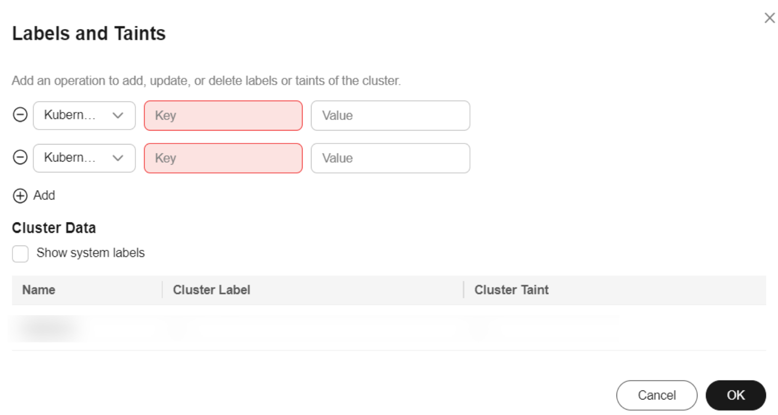
Step 2 Click the name of the fleet where the target cluster is located. In the navigation pane, choose **Container Clusters**, locate the target cluster, and click in the upper right corner to go to the **Manage Labels and Taints** page.

Figure 3-52 Label and taint management



Step 3 Click to add a node label or taint. You can add a maximum of 10 operations at a time.

Figure 3-53 Adding labels or taints



- Choose **Add** or **Delete**.
- Set the operation object to **Kubernetes Label** or **Taint**.
- Specify **Key** and **Value**.
- If you choose **Taint**, select a taint effect. For details, see **Taints**.

Step 4 Click **OK**.

----End

3.15 Cluster Federation RBAC Authorization

UCS cluster federation can implement refined permission management based on Huawei Cloud IAM. In addition, native Kubernetes RBAC resources can be created in the federation for refined management of federation access permissions.

Precautions

- The **permission management** of UCS and the current RBAC authorization of the cluster federation do not affect each other. When UCS APIs are called, the

UCS permission management takes effect. If the kubeconfig file is used to perform federation operations, the RBAC authorization takes effect.

- RBAC resources created in the cluster federation and member clusters are unaware of and do not affect each other. The RBAC permissions configured through the cluster federation entry take effect only when the federation is directly accessed. When a member cluster is directly accessed, only the RBAC permissions for the member cluster take effect.
- You need to assign permissions and roles (such as ClusterRole and ClusterRoleBinding) with caution for fine-grained authorization. Do not assign the permission to view resources to namespaces prefixed with Karmada-. Role and RoleBinding are recommended for assigning permissions to resources in specified namespaces.

Cluster Federation RBAC Authorization

The UCS cluster federation uses the native RBAC authentication mode of Kubernetes. You can create RBAC resources to assign federation access permissions to IAM users.

Step 1 Download and configure the kubeconfig file as an IAM user with the Tenant Administrator permission. For details, see Using kubectl to Connect to a Federation.

Step 2 Save the following content to the **list-deploy.yaml** file:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: list-deploy-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: list-deploy-role
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: <user-id>    # IAM user ID
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
    name: <group-id>   # IAM user group ID
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: list-deploy-role
  namespace: default
rules:
  - apiGroups:
      - apps
    resources:
      - deployments
    verbs:
      - list
      - get
```

<user-id> indicates the IAM user ID, and *<group-id>* indicates the IAM user group ID. For details about the fields in **RoleBinding** and **Role**, see **Using RBAC Authentication**.

Run the following command to create the resources:

```
kubectl apply -f list-deploy.yaml
```

The IAM user specified by *<user-id>* or IAM users in the group specified *<group-id>* can run the following command to view the Deployments in the default namespace:

```
kubectl get deploy -n default
```

----End

4 Image Repositories

UCS integrates Huawei Cloud SoftWare Repository for Containers (SWR), which provides easy, secure, and reliable management over container images throughout their lifecycles, facilitating the deployment of containerized applications.

SWR allows you to securely host and efficiently distribute images on the cloud to smoothly run your services in containers. You do not need to build or maintain image repositories.

Features

- Full lifecycle management of images

  SWR manages the full lifecycle of your container images, including push, pull, and deletion.

- Private image repository

  Private image repository and fine-grained permission management allow you to grant different access permissions, namely, read, write, and edit, to different users.

- Image Acceleration

  Acceleration technology developed by Huawei brings faster image pull for CCE clusters during high concurrency.

- Automatic deployment update through triggers

  Application deployment can be triggered automatically upon image tag update. You only need to set a trigger for the desired image. Every time the image tag is updated, the application deployed with this image will be automatically updated.

Constraints

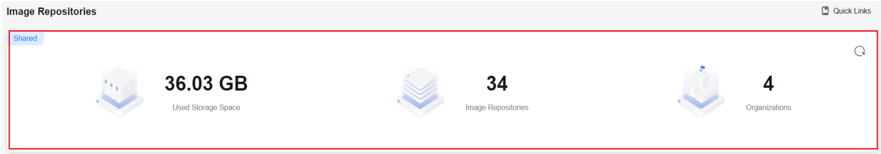
Attached clusters connected to UCS through a private network cannot download images from SWR. Ensure your clusters can access the public network.

Pushing the Image

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Image Repositories**.

Step 2 View the basic information about the image repository and click the image repository name to access SWR.

Figure 4-1 Image repositories



Step 3 Upload an image to SWR by referring to **Uploading an Image Through a Container Engine Client**.

----End

Using an Image

Clusters and federations managed by UCS allow you to create a workload by pulling an image from the image repository. The following uses the CCE cluster taken over by UCS as an example to shown you how to pull and use an image to create a workload:

Step 1 Log in to the cluster console.

Step 2 In the navigation pane, choose **Workloads** and click **Create from Image** in the upper right corner.

Step 3 In the **Basic Info** area, set workload parameters. Deployment is used as an example.

- **Workload Type**: Select **Deployment**.
- **Workload Name**: The value can be customized.
- **Pods**: Set this parameter based on service requirements.
- **Description**: Enter the description of the workload.
- **Time Zone Synchronization**: Specify whether to enable this function. After time zone synchronization is enabled, the container and node use the same time zone. The time zone synchronization function depends on the local disk mounted to the container. Do not modify or delete the time zone.

Step 4 In the **Container Settings** area, click **Select Image**.

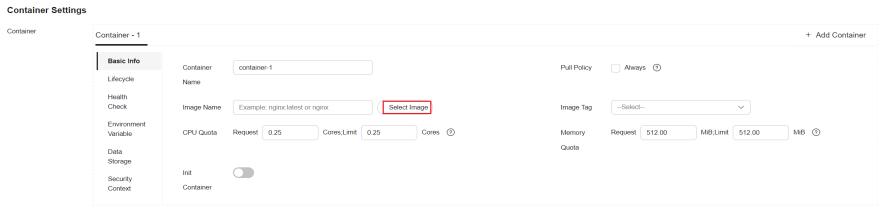
On the **My Images** tab page, select the target image and click **OK**.

NOTICE

- If the selected image is a public image, you do not need to select an **Image Access Credential**.
- If the selected image is a private image, you need to select an **Image Access Credential**. Otherwise, the image cannot be pulled.

  You can click **Create Secret** to create an image access credential. For details, see **Creating an Image Secret**.

Figure 4-2 Container settings



Step 5 Click **Create Workload**. For details about how to create a workload, see **Deployments**.

----End

Creating an Image Secret

When a Huawei Cloud cluster is being created, a secret named **default-secret** is generated by default, which contains an access credential of SWR. You do not need to create an image secret again.

When an attached cluster uses SWR private images, you need to create an image secret to pull SWR images. The procedure is as follows:

Step 1 Log in to the cluster console.

Step 2 In the navigation pane on the left, choose **ConfigMaps and Secrets**, and click the **Secrets** tab.

Step 3 Click **Create Secret** and set parameters.

Figure 4-3 Creating a secret

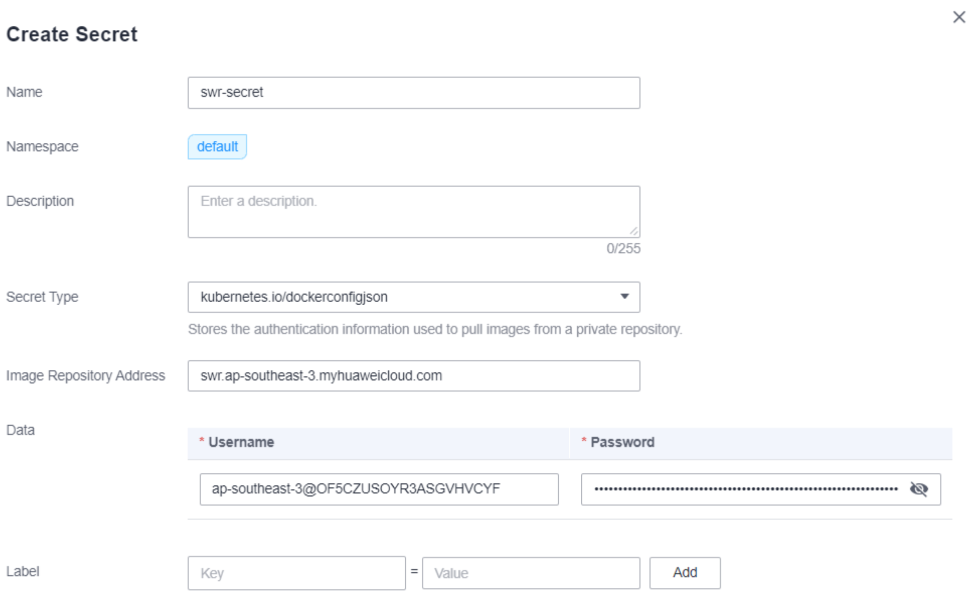


Table 4-1 Parameter description

Parameter	Description
Name	Name of the secret you create, which must be unique.
Namespace	Namespace to which the secret belongs. If you do not specify this parameter, the value **default** is used by default.
Description	Description of a secret.
Secret Type	Type of the new secret. **kubernetes.io/dockerconfigjson** stores the authentication information required for pulling images from a private repository.
Image Repository Address	The image repository address is **swr.*region*.myhuaweicloud.com**. For example, the image repository address of AP-Singapore is **swr.ap-southeast-3.myhuaweicloud.com**. For details about the regions where SWR is used, see **Regions and Endpoints**.
Data	Enter the username and password of the private image repository. Workload secret data can be used in containers. To obtain the username and password when using SWR, perform the following steps: 1. Click the username in the upper right corner, choose **My Credentials** > **Access Keys**, and click **Create Access Key**. You can obtain the AK and SK information from the **credentials.csv** file downloaded. The AK/SK file can be **downloaded only once**. Keep it secure. For more details about access keys, see**Access Keys**. 2. Log in to a Linux computer and run the following command to obtain the login key (*$AK* and *$SK* are the AK/SK obtained in the previous step.): **printf "*$AK*" \| openssl dgst -binary -sha256 -hmac "*$SK*" | od -An -vtx1 | sed 's/[\n]//g' | sed 'N;s/\n//'** 3. The username is ***Regional project name*@*AK***, for example, ap-southeast-3@***. The password is the login key obtained in **2**.
Label	Label of the secret. Enter a key-value pair and click **Add**.

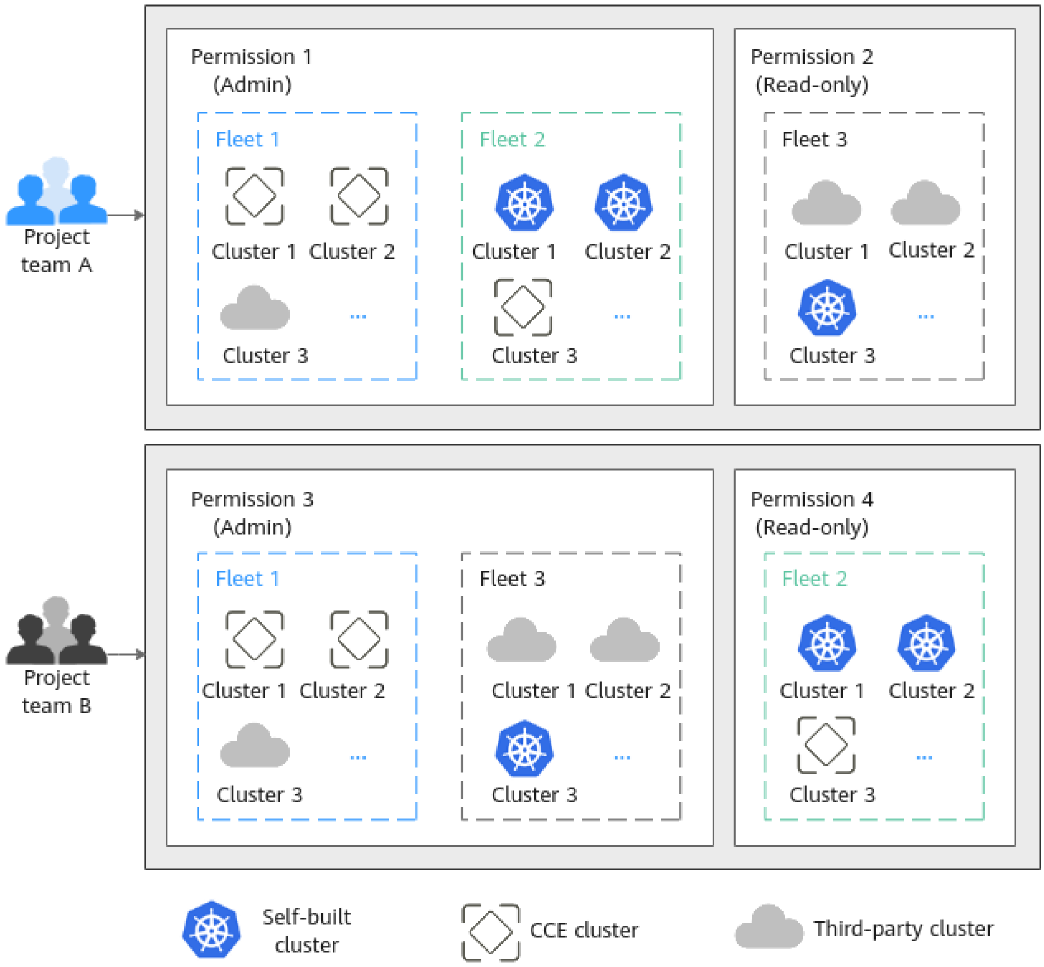
----End

5 Permissions

5.1 UCS Permissions

UCS allows you to grant cluster permissions to IAM users and user groups under your account, so that departments or projects can be isolated by permission policy or cluster group.

Figure 5-1 Permission design



UCS Permission Types

UCS provides refined permission management based on the role access control (RBAC) capability of IAM and Kubernetes. Permission control can be implemented by UCS service resource and Kubernetes resource in a cluster. The two permission types apply to different resource types and are granted using different methods.

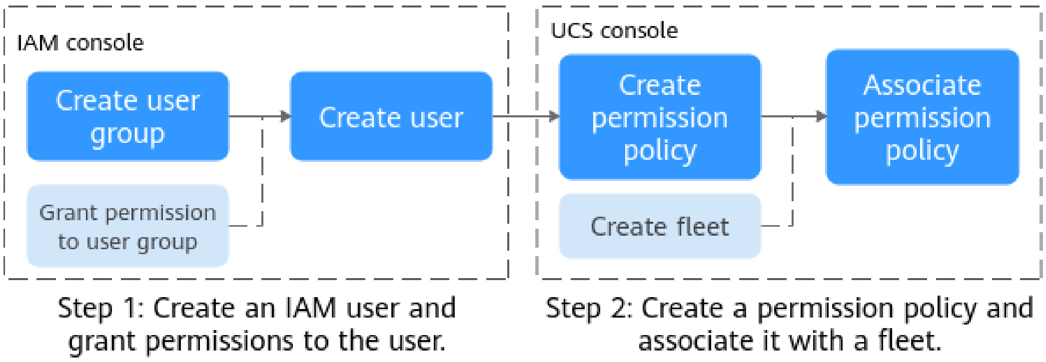
- **UCS resource permissions** are granted based on the system policies of IAM. UCS resources include container fleets, clusters, and federation instances. Administrators can grant different permissions to different user roles (such as development and O&M) to control their use of UCS resources.
- **Kubernetes resource permissions in a cluster** are granted based on the Kubernetes RBAC capability. Refined permissions can be granted to Kubernetes resource objects in a cluster. With permission setting, the permissions for performing operations on different Kubernetes resource objects (such as workloads, jobs, and services) will vary with users.

UCS permissions apply to three phases: creating and managing infrastructure resources in the first phase, that is, creating container fleets, registering clusters, and enabling cluster federation; using cluster Kubernetes resource objects (such as workloads and services) in the second phase; O&M infrastructure resources and Kubernetes resources in the third phase. In the first and third phases, UCS resource permissions are granted following the IAM system policies on the IAM console. In the second phase, Kubernetes resource permission policies are created by the administrator on the **Permissions** page of the UCS console, and are associated with specific fleets or clusters on the **Fleets** page.

Permission Management Flow

Figure 5-2 shows the permission management flow of a new IAM user.

Figure 5-2 Permission management flow

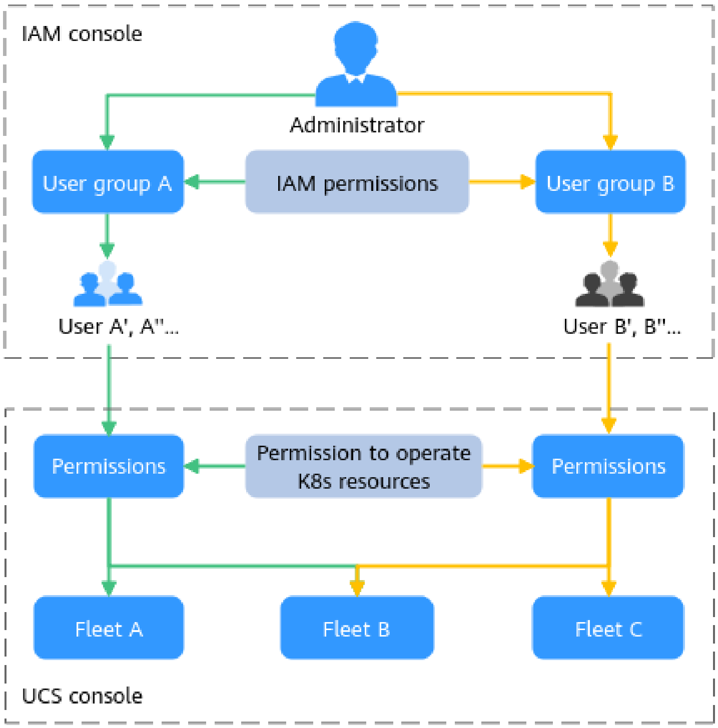


Basic Concepts

Figure 5-3 shows the relationships between the following basic concepts:

- **User:** You can use your administrator account to create IAM users and grant permissions on specific resources. Each IAM user has their own identity credentials (password and access keys) and uses cloud resources based on granted permissions.
- **User group:** You can use user groups to grant permissions to IAM users. IAM users added to a user group automatically obtain the permissions granted to the group. For example, after the administrator grants the **UCS FullAccess** permission to a user group, users in the user group have the administrator permissions of UCS. If a user is added to multiple user groups, the user inherits the permissions granted to all these groups.
- **Permission:** The UCS administrator defines the scope of operations performed by one or more users on Kubernetes resources in a cluster. UCS presets several common permissions, including **Admin**, **Viewer**, and **Developer** permissions, and supports custom permissions. For details, see **Creating a Permission Policy**.
- **Fleet:** A fleet contains multiple clusters. Administrators can use fleets to classify associated clusters. Fleets can also be used to manage multiple clusters based on permissions, security policies, configurations, and multi-cluster orchestration. Fleets and permissions are in a many-to-many relationship. That is, a permission policy can be associated with multiple fleets, and a fleet can be associated with multiple permission policies.

Figure 5-3 Permission relationship



Constraints

- An on-premises cluster can use the Huawei Cloud IAM token to access kube-apiserver, and does not identify the system policies (**UCS FullAccess**, **UCS CommonOperations**, **UCS CIAOperations**, and **UCS ReadOnlyAccess**) of UCS.
- Multi-cloud clusters can only be registered using a Huawei Cloud account. Cluster registration through IAM system policies is not supported.

5.2 UCS Resource Permissions (IAM Authorization)

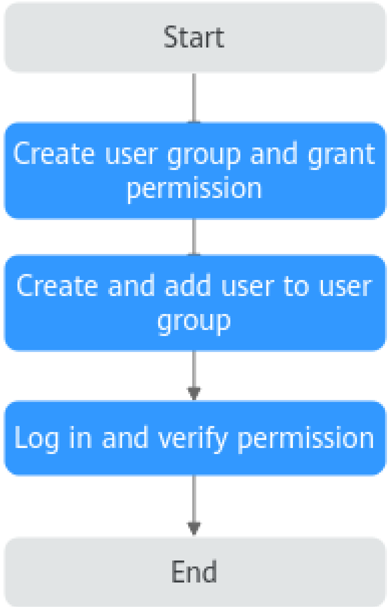
UCS resources include container fleets, clusters, and federation instances. Administrators can grant different permissions to different user roles (such as development and O&M) to control their use of UCS resources. UCS resource permissions are granted following the IAM system policies.

IAM grants permissions to users through user groups. Before granting permissions to a user group, read the **UCS system policies** that can be added to the user group and the **minimum permissions required by UCS**. To learn about the permission policies of other cloud services, see **System Permissions**.

Permission Granting Process

This section uses the **UCS ReadOnlyAccess** policy as an example to describe how to grant permissions to a user. **Figure 5-4** shows the process.

Figure 5-4 Process for granting UCS permissions to a user



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

   Create a user group on the IAM console as the administrator, and grant UCS permissions, for example, the **UCS ReadOnlyAccess** policy to the group.

   NOTE

   UCS is a global service deployed in all physical regions. When granting permissions, set the authorization scope to **All resources**.

2. **Create an IAM 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 the permissions.

   Log in to the console as an IAM user and verify the permissions (assume that the user has only the **UCS ReadOnlyAccess** policy).

   – Choose **UCS** from **Service List** of Huawei Cloud. In the left navigation pane, choose **Infrastructure** > **Fleets**. If a message is displayed indicating that you do not have the access permission when you create a fleet or register a cluster, the **UCS ReadOnlyAccess** permission has taken effect.

   – Choose another service (such as ECS) in **Service List** of Huawei Cloud. If a message is displayed indicating insufficient permissions, the **UCS ReadOnlyAccess** policy has taken effect.

System Policies

The preset UCS system policies of IAM include **UCS FullAccess**, **UCS CommonOperations**, **UCS CIAOperations**, and **UCS ReadOnlyAccess**.

- **UCS FullAccess**: UCS administrator with full permissions, including creating permission policies and security policies
- **UCS CommonOperations**: Common UCS user with permissions for creating workloads, distributing traffic, and other operations

- **UCS CIAOperations**: CIA administrator with full permissions in UCS
- **UCS ReadOnlyAccess**: Read-only permissions on UCS (except for CIA)

You can check a policy to learn about the actions supported by a system policy. An action is in the format of *Service name*:*Resource type*:*Operation*. The wildcard (*) is allowed, indicating all actions.

The following is an example of the **UCS FullAccess** policy. This policy contains all permissions on UCS, CCE, and SWR, and operation permissions on some resources of services such as AOM, SMN, and DNS.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ucs:*:*",
                "cce:*:*",
                "swr:*:*",
                "aom:*:get",
                "aom:*:list",
                "smn:*:list",
                "dns:*:get*",
                "dns:*:list*",
                "dns:*:get",
                "dns:*:list",
                "dns:recordset:create",
                "dns:recordset:delete",
                "dns:recordset:update",
                "dns:tag:get",
                "lts:*:get",
                "lts:*:list",
                "apm:*:get",
                "apm:*:list",
                "vpcep:epservices:*",
                "vpcep:connections:*",
                "vpcep:endpoints:*",
                "elb:*:get",
                "elb:*:list",
                "vpc:*:get",
                "vpc:*:list",
                "ief:*:get",
                "ief:*:list",
                "cgs:images:operate",
                "cgs:*:get",
                "cgs:*:list"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Minimum Permissions Required by UCS

Services on Huawei Cloud are interdependent, and UCS depends on other cloud services to implement some functions, such as image repository and domain name resolution. The preceding four system policies are often used together with roles or policies of other cloud services for refined permission granting. When granting permissions to IAM users, the administrator must comply with the principle of least privilege. **Table 5-1** lists the minimum permissions required by the administrator, operation, and read-only permissions of each UCS function.

NOTICE

- For the first login of your Huawei Cloud account to the UCS console, you need to grant permissions to the account. Then UCS will create an agency named **ucs_admin_trust** for you in IAM. Do not delete or modify the agency.
- If the user group of an IAM user is not granted any permissions, you cannot access the UCS console. See **Table 5-1**.

Table 5-1 Minimum permissions required by UCS

Description	Permission Type	Permission	Minimum Permission
Fleet	Admin	• Creating and deleting a fleet • Registering a Huawei Cloud cluster (CCE cluster and CCE Turbo cluster) , on-premises cluster, or attached cluster • Unregistering a cluster • Adding a cluster to or removing a cluster from a fleet • Associating permission policies with a cluster or fleet • Enabling cluster federation and performing federation management operations (such as creating a federated workload and creating domain name access)	UCS FullAccess
	Viewer	Querying clusters and fleets or their details	UCS ReadOnlyAccess
Huawei Cloud clusters	Admin	Read-write permissions on Huawei Cloud clusters and all Kubernetes resource objects (including nodes, workloads, jobs, and services)	UCS FullAccess + CCE Administrator
	Operation	Read-write permissions on Huawei Cloud clusters and most Kubernetes resource objects and read-only permissions on Kubernetes resource objects such as namespaces and resource quotas	UCS CommonOperations + CCE Administrator
	Viewer	Read-only permissions on Huawei Cloud clusters and all Kubernetes resource objects (including nodes, workloads, jobs, and services)	UCS ReadOnlyAccess + CCE Administrator

Description	Permission Type	Permission	Minimum Permission
On-premises/ Attached/ Multi-cloud clusters	Admin	Read-write permissions on on-premises/attached/multi-cloud clusters and all Kubernetes resource objects (including nodes, workloads, jobs, and services)	UCS FullAccess
	Operation	Read-write permissions on on-premises/attached/multi-cloud clusters and most Kubernetes resource objects and read-only permissions on Kubernetes resource objects such as namespaces and resource quotas	UCS CommonOperations + UCS RBAC (The list permission for namespaces is required.)
	Viewer	Read-only permissions on on-premises/attached/multi-cloud clusters and all Kubernetes resource objects (including nodes, workloads, jobs, and services)	UCS ReadOnlyAccess + UCS RBAC (The list permission for namespaces is required.)
Image repository	Admin	All permissions on SoftWare Repository for Container (SWR), including creating organizations, uploading images, viewing images or details, and downloading images	SWR Administrator
Permissions	Admin	• Creating and deleting a permission policy • Viewing permissions or details **NOTE** When creating a permission policy, you need to grant the **IAM ReadOnlyAccess** permission (read-only permissions on IAM) to IAM users to obtain the IAM user list.	UCS FullAccess + IAM ReadOnlyAccess
	Viewer	Viewing permissions or details	UCS ReadOnlyAccess + IAM ReadOnlyAccess
Policy Center	Admin	• Enabling the Policy Center • Creating and disabling a policy • Querying policies • Viewing policy implementation details	UCS FullAccess
	Viewer	For fleets and clusters with Policy Center enabled, users with this permission can view policies and policy implementation details.	UCS CommonOperations or UCS ReadOnlyAccess

Description	Permission Type	Permission	Minimum Permission
Traffic distribution	Admin	Operations such as creating a traffic policy, suspending and deleting a scheduling policy	(Recommended) UCS CommonOperations + DNS Administrator or UCS FullAccess + DNS Administrator
	Viewer	Viewing traffic policies or details	UCS ReadOnlyAccess + DNS Administrator
CIA	Admin	• Connecting clusters to a fleet or canceling cluster connection • Viewing monitoring data in multiple aspects, such as infrastructure and application workload	UCS CIAOperations

5.3 Kubernetes Resource Permissions in a Cluster (RBAC Authorization)

Kubernetes resource permissions in a cluster are granted based on the Kubernetes RBAC capability. The administrator can grant users operation permissions on specific Kubernetes resource objects in a cluster. The permissions take effect on the namespace of a fleet or on clusters that do not join the fleet.

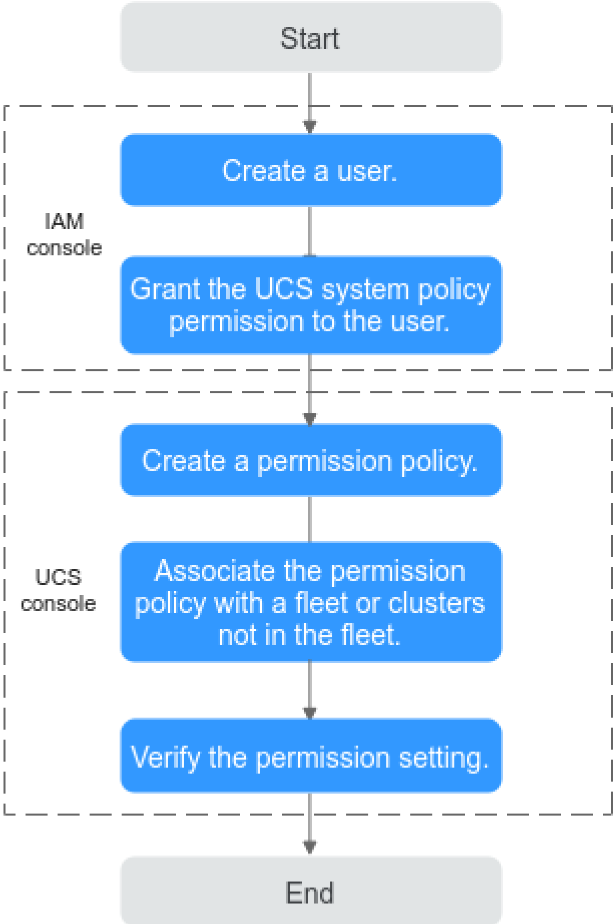
This section uses the read-only permission as an example to describe how to grant Kubernetes resource permissions to users. **Figure 5-5** shows the operation process.

NOTICE

The UCS cluster operation permission setting takes effect only for non–Huawei Cloud clusters. Operation permissions of Huawei Cloud clusters (CCE and CCE Turbo clusters) are subject to the IAM or CCE RBAC permissions.

Permission Granting Process

Figure 5-5 Process for granting Kubernetes resource permissions to a user



1. **Create a user**.

   The administrator creates a user on the IAM console.

2. **Grant the UCS system policy permission to the user**.

   Before granting the Kubernetes resource permissions, you must grant the UCS system policy permission to the user. In this example, the **UCS ReadOnlyAccess** policy (read-only permission on UCS) must be granted.

3. **Create a permission policy**.

   The administrator creates a permission policy on the UCS console. Select the **Viewer** permission type, which indicates read-only permissions on all Kubernetes resource objects.

4. **Associate the permission policy with a fleet or clusters not in the fleet**.

   Associate the permission policy with a fleet. During the association, you need to select the namespace to which the permission policy applies. You can also associate the permission policy with clusters not in the fleet.

5. **Verify the permission setting**.

   Log in to the console as the created user, and verify whether the read-only permission takes effect.

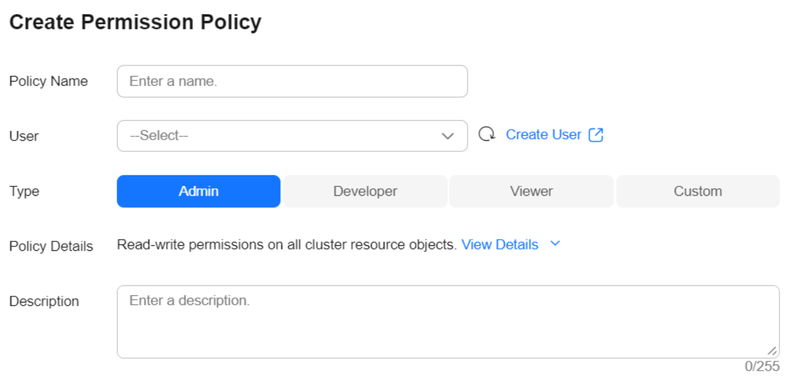
Creating a Permission Policy

Step 1 Log in to the UCS console. In the navigation pane, choose **Permissions**.

Step 2 Click **Create Permission Policy** in the upper right corner.

Step 3 Configure permission policy parameters.

Figure 5-6 Creating a permission policy



- **Policy Name**: Enter a name, starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **User**: Select the newly created username from the drop-down list. You can select multiple users. Assume that the R&D employees of a company have the same operation permission on resources. When creating a permission policy, you can select multiple users to grant permissions to all these users.

  This section uses the **readonly_user** user as an example.
- **Type**: **Admin**, **Viewer**, **Developer**, and **Custom** permissions are supported.

Table 5-2 Permission types

Permission Type	Description
Admin	Read-write permissions on all cluster resource objects.
Viewer	Read-only permissions on all cluster resource objects.
Developer	Read-write permissions on most cluster resource objects and read-only permissions on cluster resource objects such as namespaces and resource quotas.
Custom	Permissions are determined by the actions and resource objects you select.

- **Policy Details**: indicates the actions allowed on specific resources. The **Admin**, **Viewer**, and **Developer** permission types have been templated. You can click to view the details of a permission type. When **Type** is set to **Custom**, configure **Operation to perform** and **Resource Object**.

Operation to perform: You can add an operation type (for example, **deletecollection** indicates the deletion of multiple resources). The options are as follows:

- **get**: Retrieves a specific resource object by name.
- **list**: Retrieves all resource objects of a specific type in the namespace.
- **watch**: Responds to resource changes.
- **create**: Creates a resource.
- **update**: Updates a resource.
- **patch**: Updates resources partially.
- **delete**: Deletes a resource.

NOTE

All operations: **All**

Read-only: **get** + **list** + **watch**

Read-write: **get** + **list** + **watch** + **create** + **update** + **patch** + **delete**

Resource Object: Select **All** or **Resources to operate**. **All** includes existing resource objects and custom resource objects to be added. **Resources to operate** indicates the custom range of resource objects. UCS categorizes resource objects by workload, service, config and storage, authentication, authorization, policy, extend, and cluster.

If the desired resource object does not exist in system resources, you can add a custom resource object.

If the operation types vary according to resource objects (for example, you have the **create** and **delete** permissions on Deployments and the **get**, **list**, and **watch** permissions on secrets), you can click + to add multiple groups of permissions.

NOTE

For details about resource objects and operation types, see **Kubernetes API**.

- **Description**: Enter a description of the permission policy to be added.

Step 4 Click **OK**. After the permission policy is created, you need to associate the permission policy with a fleet or clusters not in the fleet so that you can perform operations on Kubernetes resources.

----End

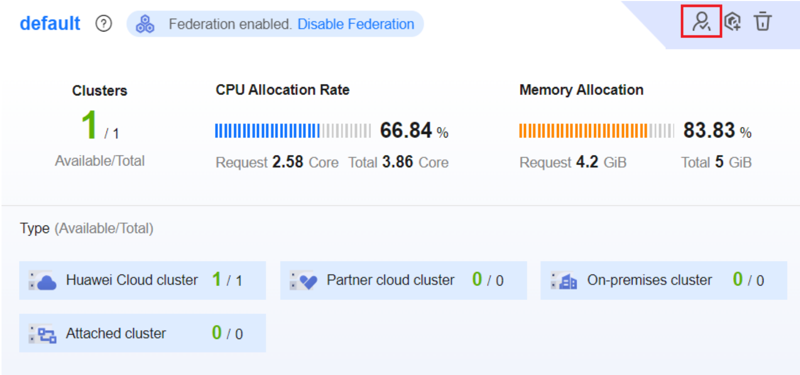
Associating the Permission Policy with a Fleet or Clusters Not in the Fleet

A fleet contains multiple clusters and can implement unified permission management for these clusters. After clusters join a fleet, you are advised to associate the permission policy with the fleet so that clusters in the fleet can have the same permissions.

Step 1 Log in to the UCS console. In the navigation pane, choose **Fleets**.

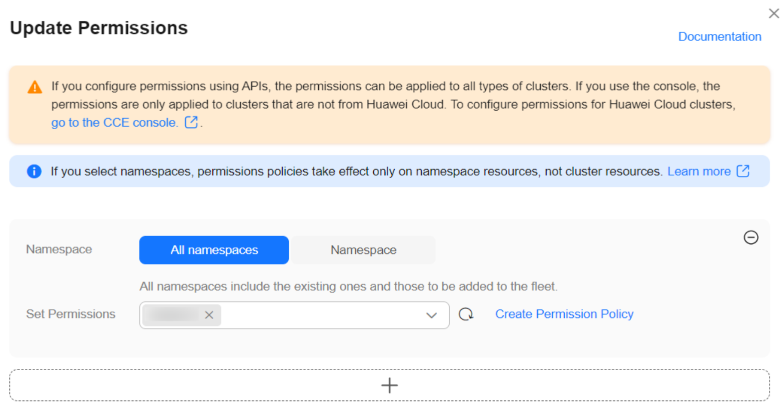
Step 2 In the card view of the destination fleet, click in the upper right corner.

Figure 5-7 Associating a permission policy with a fleet



Step 3 On the displayed page, click **Update Fleet Permissions** or **Set Permissions**. Then, associate the created permission policy with the namespace of the fleet.

Figure 5-8 Updating a permission policy



- **Namespace**: Select **All namespaces** or **Namespace**. **All namespaces** includes the existing namespace of the fleet and the namespace to be added to the fleet. **Namespace** indicates the custom range of namespaces. UCS provides several common namespaces, such as **default**, **kube-system**, and **kube-public**. You can also add a namespace, which should exist in the cluster.

  If you select namespaces, permission policies take effect only on namespace resources, not cluster resources. For details about namespace and cluster resources, see **Kubernetes Resource Objects**.
- **Set Permissions**: Select permissions from the drop-down list box. You can select multiple permissions at a time to batch grant permissions.

In this example, select **default** for namespace and the **readonly** permission.

If different namespaces are associated with different permission policies (for example, the **default** namespace is associated with the **readonly** permission policy and the **development** namespace is associated with the **develop** permission policy, you can click + to add multiple relationships of permission granting.

Step 4 Click **OK**.

If you need to update the permission policy of the fleet, select the namespace and permission again using the preceding method.

----End

Verifying the Permission Setting

Log in to the console as the newly created **readonly_user**user and check whether the permission takes effect. The following uses an attached cluster as an example.

- Go to the attached cluster of the fleet and choose **Resources** > **Workloads**. If you can view the workloads of the **default** namespace but a message is displayed indicating that you do not have the permission for viewing workloads of other namespaces, the read-only permission has taken effect.
- Go to the attached cluster of the fleet and choose **Resources** > **Workloads**. Switch to the **default** namespace, and click **Create Workload** in the upper right corner. If a message is displayed indicating that you do not have the permission, the read-only permission has taken effect.

5.4 Kubernetes Resource Objects

By their application scope, Kubernetes resource objects can be categorized into namespace objects or cluster objects.

Namespace Level

Namespace is an isolation mechanism of Kubernetes and is used to categorize, filter, and manage any resource object in a cluster.

If different resource objects are placed in different namespaces, they are isolated from each other. For example, run the following command to obtain all pods:

```
kubectl get pod
```

The pod has a namespace, which defaults to **default**. To specify a namespace, run the following command:

```
kubectl get pod -n default
```

To obtain pods in all namespaces, run the following command:

```
kubectl get pod --all-namespaces
```

In this way, you can view all pods in the cluster.

```
$ kubectl get pod --all-namespaces
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE
default       nginx-dd9796d66-5chbr                       1/1     Running   0          3d1h
default       nginx-dd9796d66-xl69p                       1/1     Running   0          15d
default       sa-example                                  1/1     Running   0          10d
kube-system   coredns-6fcd88c4c-k8rtf                     1/1     Running   0          48d
kube-system   coredns-6fcd88c4c-z46p4                     1/1     Running   0          48d
kube-system   everest-csi-controller-856f8bb679-42rgw     1/1     Running   1          48d
kube-system   everest-csi-controller-856f8bb679-xs6dz     1/1     Running   0          48d
kube-system   everest-csi-driver-mkpbv                    2/2     Running   0          48d
kube-system   everest-csi-driver-v754w                    2/2     Running   0          48d
kube-system   icagent-5p44q                               1/1     Running   0          48d
kube-system   icagent-jrlbl                               1/1     Running   0          48d
monitoring    alertmanager-alertmanager-0                 2/2     Running   0          29d
monitoring    cluster-problem-detector-7788f94f64-thp6s   1/1     Running   0          29d
```

```
monitoring    custom-metrics-apiserver-5f7dcf6d9-n5nrr        1/1     Running   0          19d
monitoring    event-exporter-6844c5c685-khf5t               1/1     Running   1          3d1h
monitoring    kube-state-metrics-8566d5f5c5-7kx7b           1/1     Running   0          29d
monitoring    node-exporter-7l4ml                           1/1     Running   0          29d
monitoring    node-exporter-gpxvl                           1/1     Running   0          29d
```

Pods are namespace objects. Most workload resources, Service resources, and config and storage are also namespace objects.

- **Workload resources**

  Pod: the smallest and simplest unit in the Kubernetes object model that you create or deploy.

  ReplicaSet: a backup controller in Kubernetes. It is used to control the managed pods so that the number of pod replicas remains the preset one.

  Deployment: declares the pod template and controls the pod running policy. It is applicable to the deployment of stateless applications.

  StatefulSet: manages stateful applications. Created pods have persistent identifiers created based on specifications.

  DaemonSet: used to deploy background programs in the resident cluster, for example, node log collection.

  Job: The job controller creates one or more pods. These pods run according to the running rules until the running is complete.

  CronJob: periodically runs a job based on a specified schedule.

- **Service resources**

  Service: Containers deployed in Kubernetes provide layer-7 network services using HTTP and HTTPS, and layer-4 network services using TCP and UDP. Services in Kubernetes are used to manage layer-4 network access in a cluster. Based on the four-layer network, Service exposes the container services in a cluster.

  Ingress: provides Layer-7 network services using HTTP and HTTPS and common Layer-7 network capabilities. An ingress is a set of rules that allow accessing Services in a cluster. You can configure forwarding rules to enable different URLs to access different Services in a cluster.

- **Config and storage resources**

  ConfigMap: key-value pair, which is used to decouple configurations from running images so that applications more portable.

  Secret: key-value pair, which is used to store sensitive information such as passwords, tokens, and keys to reduce the risk of direct exposure.

  Volume: A volume is essentially a directory that may contain some data. Containers in a pod can access the directory. A volume will no longer exist if the pod to which it is mounted does not exist. However, files in the volume may outlive the volume, depending on the volume type.

Cluster Level

A cluster resource has a much larger application scope than a namespace resource. It is visible to the entire cluster and can be invoked. It does not belong to a certain namespace. Therefore, the name of a resource object must be globally unique.

Cluster resources are visible in any namespaces. You do not need to specify a namespace when defining cluster resources.

Cluster resources include Namespace, Node, Role, RoleBinding, ClusterRole, and ClusterRoleBinding.

- **Namespace**: an isolation mechanism of Kubernetes and is used to categorize, filter, and manage any resource object in a cluster.

  To query all namespaces in a cluster, run the following command:

  ```
  kubectl get ns
  ```

- **Node**: A node is a basic element of a container cluster and can be a VM or physical machine. The components on a node include kubelet and kube-proxy. A node name must be globally unique.
- **Role**: defines a set of rules for accessing Kubernetes resources in a namespace.
- **RoleBinding**: defines the relationship between users and roles.
- **ClusterRole**: defines a set of rules for accessing Kubernetes resources in a cluster (including all namespaces).
- **ClusterRoleBinding**: defines the relationship between users and cluster roles.

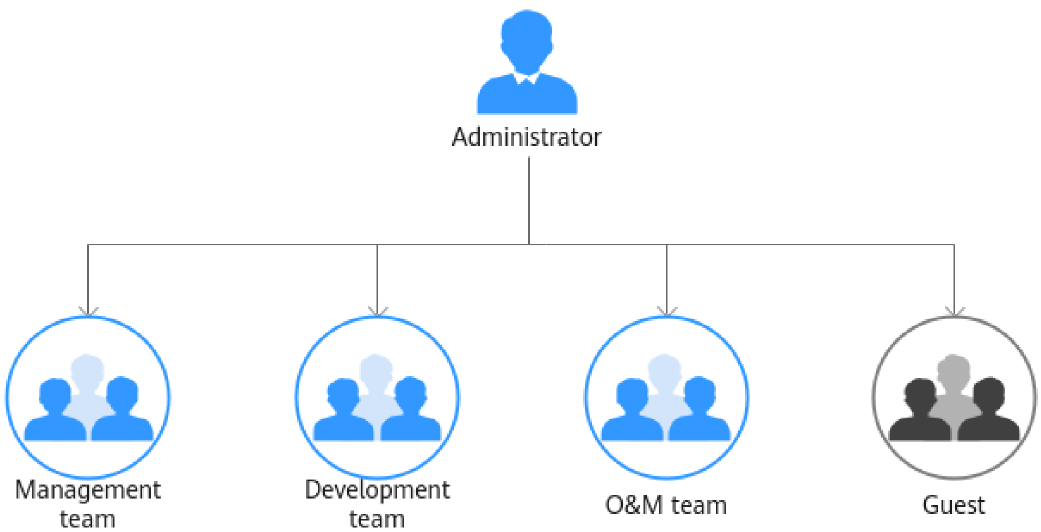
  NOTE

  Role and ClusterRole specify actions that can be performed on specific resources. RoleBinding and ClusterRoleBinding bind roles to specific users, user groups, or ServiceAccounts.

5.5 Example: Designing and Configuring Permissions for Users in a Company

A company uses Huawei Cloud UCS to manage multiple clusters. The company has multiple functional teams responsible for permission granting, resource management, application creation, traffic distribution, and O&M, respectively. Using the permissions management of IAM and UCS can achieve refined permission granting.

Figure 5-9 Organizational structure



- Management team: manages all resources of the company.
- Development team: develops services.
- O&M team: views and monitors the usage of all resources.
- Guest: a reserved read-only team that has only the permission for viewing resources.

Grant required permissions to different functional teams in the company according to **Table 5-3**.

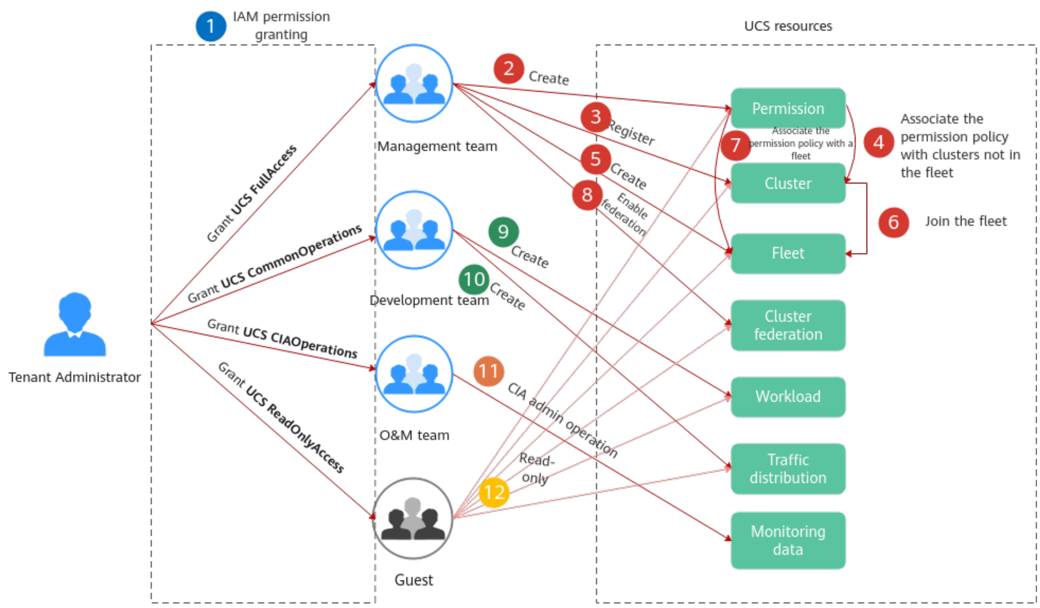
Table 5-3 Permissions

Functional Team	Policy to Be Granted	Permission Description
Management team	UCS FullAccess	UCS administrator with full permissions, including creating permissions policies and security policies
Development team	UCS CommonOperations	Common UCS user with permissions for creating workloads, distributing traffic, and other operations
O&M team	UCS CIAOperations	CIA administrator with full permissions in UCS
Guest	UCS ReadOnlyAccess	Read-only permissions on UCS (except for CIA)

Permission Design

The following figure shows the operations that can be performed by different functional teams on UCS resources.

Figure 5-10 Operations that can be performed on UCS resources

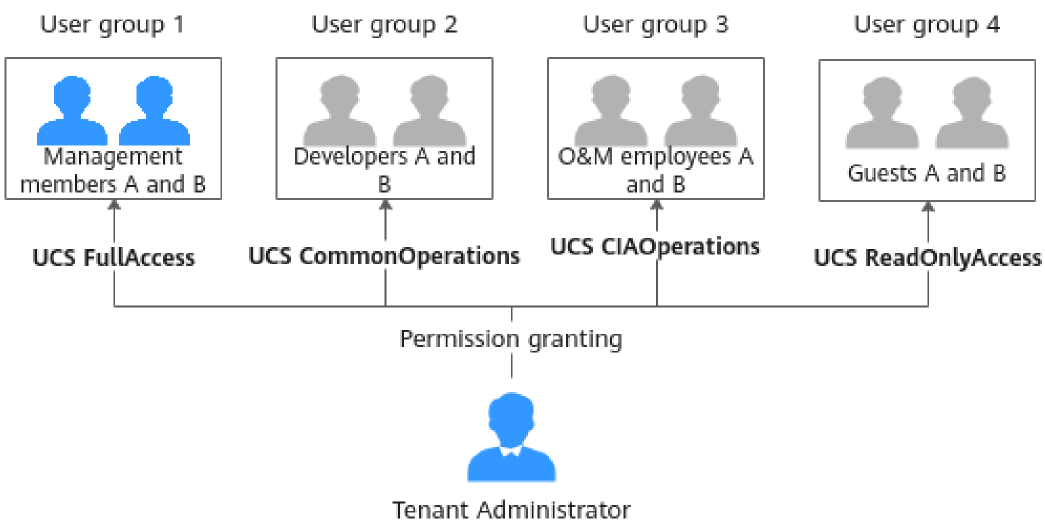


- 1: Tenant Administrator grants permissions to each functional team.
- 2 to 8: The management team with the **UCS FullAccess** permission is responsible for creating a fleet, registering a cluster, adding a cluster to the fleet, enabling cluster federation, and building the multi-cluster federation infrastructure. In addition, the management team creates permissions and associates them with the fleet or clusters that are not added to the fleet so that the development team has the corresponding operation permissions on specific Kubernetes resources.
- 9 and 10: The development team with the **UCS CommonOperations** permission performs operations such as creating workloads and distributing traffic.
- 11: The O&M team with the **UCS CIAOperations** permission performs monitoring and O&M.
- 12: Guests with the **UCS ReadOnlyAccess** permission can view resources such as clusters, fleets, and workloads.

Administrator: IAM Authorization

Tenant Administrator performs IAM authorization for each functional team by creating four user groups, granting the **UCS FullAccess**, **UCS CommonOperations**, **UCS CIAOperations**, and **UCS ReadOnlyAccess** permissions to these user groups, and adding users to each user group, as shown in **Figure 5-11**.

Figure 5-11 IAM authorization



For example, create the **dev** user group for the development team, grant the **UCS CommonOperations** permission to the user group, and add the **devuser1** and **devuser2** users.

Figure 5-12 Granting permissions

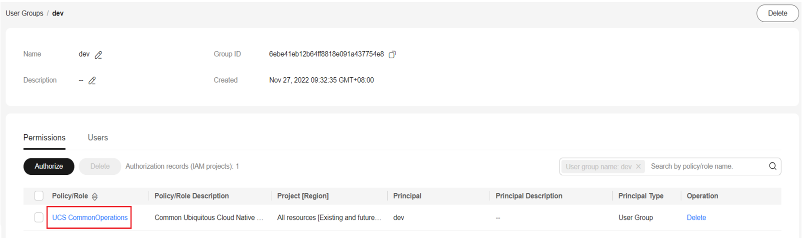
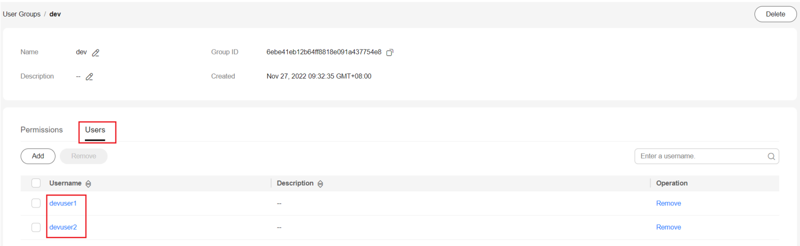


Figure 5-13 Managing users



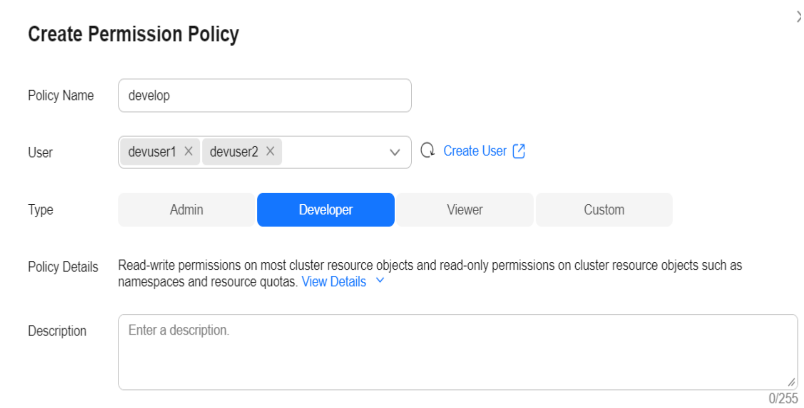
For details, see **UCS Resource Permissions (IAM Authorization)**. To use some UCS functions that depend on other cloud services, grant permissions to related cloud services. For example, the IAM user list is required for creating a permission policy, so both the **UCS FullAccess** and **VDC ReadOnlyAccess** permissions need to be granted to the management team.

Management Team: Building Infrastructure and Configuring Permission Policies

Step 1 Create a permission policy.

Create a development permission policy for developers.

Figure 5-14 Creating a development permission policy



Step 2 Create a fleet and associate the permission policy with the fleet.

A fleet contains multiple clusters and can implement unified permission management for these clusters. The management team associates the development permission created in the previous step with the fleet, so that clusters subsequently added to the fleet will have the permission. In this way, developers are allowed to perform operations on cluster resources (such as creating workloads) in the fleet. For details, see **Managing Fleets**.

Step 3 Register clusters and add them to the fleet.

UCS supports the registration of Huawei Cloud clusters, on-premises clusters, multi-cloud clusters, and attached clusters. The management team can select a cluster type as required. For details, see **Huawei Cloud Clusters**, **Overview**, **Overview**, or **Overview**.

Step 4 Enable cluster federation.

Enable it to enjoy unified orchestration of multiple clusters, cross-cluster auto scaling & service discovery, auto failover, etc. Enabling cluster federation for the fleet will federate the registered clusters in the fleet.

----End

Development Team: Creating Workloads and Distributing Traffic

After the management team builds the multi-cluster federation infrastructure, developers can use the infrastructure resources. For details, see **Workload Management** and **Traffic Distribution**.

O&M Team: Viewing and Monitoring Resource Usage

The O&M team can use the functions provided by CIA, such as intelligent analysis, dashboard, notification configuration, and 24/7 daemon, to monitor workload resources in real time, analyze application health, and complete other routine O&M tasks. For details, see **Container Intelligent Analysis**.

Guest: Viewing Resources

Guests (persons who have only the permission for viewing resources) can view resources such as clusters, fleets, and workloads.

6 Policy Center

6.1 Overview

Ensuring the consistency of configuration and security policies is challenging and is important to O&M efficiency. To solve this problem, UCS provides the policy center function implemented by the Gatekeeper based on the Open Policy Agent (OPA). This function helps you define and execute consistent policies in multiple clusters and unify the compliance status of resources.

You can create, manage, and monitor the implementation of policies across multiple clusters (fleets). In this way, you can ensure that all clusters comply with the same security and compliance requirements, thereby improving O&M efficiency. This centralized policy management makes it easier for you to cope with complex enterprise environments while ensuring that all resources are in compliance at any time, achieving higher O&M efficiency and stronger security.

The UCS Policy Center boasts the following advantages:

- Consistent policy management

  A set of security compliance policies are applied to multiple container fleets and clusters in a centralized and consistent manner.
- Assured resource security

  Resources are continuously audited to ensure that they meet security compliance requirements and do not violate policies.
- Global resource compliance view

  The global resource compliance overview helps protect and manage cluster resources.

6.2 Basic Concepts

Policy Definition

Before creating a policy instance, you need to define a policy definition, which describes both the Rego that enforces the constraint and the schema of the constraint. The schema of a policy definition allows an admin to fine-tune the behavior of a constraint, much like arguments to a function. The Rego code in a policy definition describes the specific logic of enforcement and implements different compliance rules based on your requirements. Policy definitions are flexible. Admins can adjust policy behaviors based on actual requirements when creating policy instances to meet compliance control requirements in different scenarios. For more information, see the **official documentation**.

Here is an example of a policy definition that requires all labels described by the constraint to be present:

```
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
      validation:
        # Schema for the `parameters` field
        openAPIV3Schema:
          type: object
          properties:
            labels:
              type: array
              items:
                type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredlabels
        violation[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
```

Policy Instances

Policy instances are used to inform Gatekeeper that the admin wants a ConstraintTemplate to be enforced, and how. For more information, see the **official documentation**.

The following is an example of a policy instance that uses the previously mentioned **K8sRequiredLabels** policy definition to ensure that the Gatekeeper enforces the specified label on all namespaces:

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  enforcementAction: deny
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["gatekeeper"]
```

In this example, the **K8sRequiredLabels** policy is used and the action for executing the policy is set to **deny**, which means that the Gatekeeper will deny requests that violate the policy. This policy is specified in **match** to apply only to namespace resources. In **parameters**, a label that must exist on the resource is specified. The example label is **gatekeeper**.

6.3 Enabling the Policy Center

When you use the policy center function for the first time, you need to enable it. You can choose to enable this function for a fleet or only for clusters that have not joined a fleet. After the policy center function is enabled, the system automatically installs the Gatekeeper add-on for the fleet or cluster you select.

Constraints

- Only **Huawei Cloud accounts** or users with the **UCS FullAccess** permission can enable the policy center function.
- Before enabling the policy center function for a non-Huawei Cloud cluster, ensure that the cluster can pull public network images.
- After the policy center function is enabled, the system installs the Gatekeeper add-on on the fleet or cluster. Note that the add-on occupies some cluster resources (as shown in **Table 6-1**). Therefore, ensure the cluster has sufficient resources. This will help ensure the smooth deployment of the policy center function while avoiding negative impacts on the performance of existing workloads.

Table 6-1 Resource usage of the Gatekeeper add-on

CPU	Mem
Requests: 100m * 3 Limits: 1000m * 3	Requests: 256Mi * 3 Limits: 512Mi * 3

NOTE

*** 3** indicates that there are three pods.

- When a fleet or cluster is being enabled, avoid performing any operations on the fleet or cluster. Performing operations during the enabling process may affect the enabling success.

Procedure

Step 1 Log in to the UCS console. In the navigation pane, choose **Policy Center**.

Step 2 Click **Enable**. The **Enable Policy Management** dialog box is displayed.

Step 3 Select a fleet or cluster from the drop-down list and click **OK** to return to the policy center.

You will see that policy management is being enabled. Wait for about 3 minutes.

If **The throttling threshold has been reached: policy ip over rate limit** is displayed when you enable the policy management function, traffic is limited because a large number of clusters are enabled. Wait for a while and try again.

----End

6.4 Creating and Managing Policy Instances

A policy instance is an instruction set used to guide the Gatekeeper to execute a specific policy definition and execution mode. They act as a collection of rules to help you enforce security policies and consistency in a Kubernetes cluster. This section describes how to create and manage policy instances.

Prerequisites

The policy center function has been enabled for a fleet or cluster.

Constraints

If you have deleted a policy instance from a cluster by running the **kubectl** command, you need to delete the policy instance on the console and create a policy instance again. In this way, the system delivers the new policy instance to the cluster.

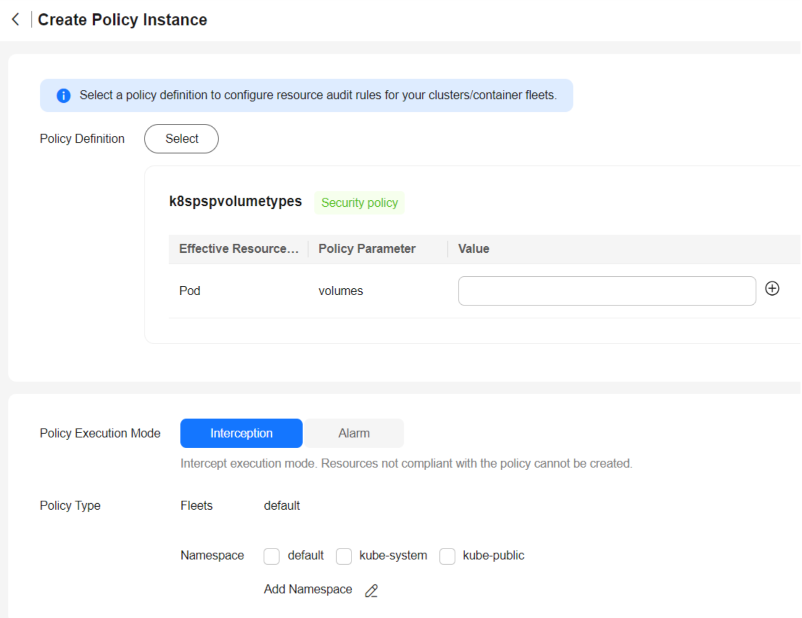
Creating a Policy Instance

Step 1 Log in to the UCS console. In the navigation pane, choose **Policy Center**.

Step 2 In the list, find the fleet or cluster for which the policy center function has been enabled and click **Create Policy Instance**.

Step 3 Set the following parameters:

Figure 6-1 Creating a policy instance



- **Policy Definition**: Select one from the 33 built-in policy definitions to configure resource audit rules for your clusters or fleets. Although custom policy definitions are not supported currently, these predefined policy definitions can basically meet your compliance and security requirements. For details about policy definition, see **Overview**.
- **Policy Execution Mode**: **Interception** and **Alarm** are supported. Interception indicates that resources that do not comply with the policy cannot be created. Alarm indicates that resources that do not comply with the policy can still be created.
- **Policy Type**: Select the namespace where the policy takes effect.

Step 4 Click **Create**. After the policy is created, the system automatically distributes the policy. If the distribution is successful, the policy instance takes effect in the cluster.

After the policy instance is successfully distributed, the action that complies with the policy instance can be executed in the cluster. If the action that does not comply with the policy instance is executed in the cluster, the action is rejected or an alarm event is reported.

----End

Modifying or Deleting a Policy Instance

As a platform engineer, you usually need to periodically review and update policy instances, or delete policy instances that are no longer used. To perform these operations, perform the following steps:

Step 1 Log in to the UCS console. In the navigation pane, choose **Policy Center**.

Step 2 In the list, click the name of the fleet or cluster for which the policy center function has been enabled. The details page is displayed.

Step 3 In the **Policy Implementation Details** area, click the **Policy Instances** tab.

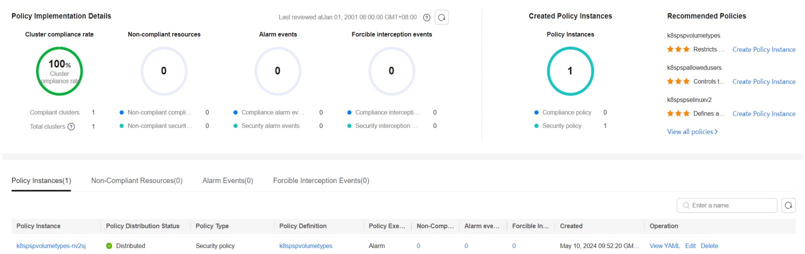
Step 4 Locate the target policy instance and click **Edit** or **Delete** in the **Operation** column to modify related parameters or delete the policy instance.

----End

Viewing the Policy Implementation Status

On the policy details page of a fleet or cluster, you can view the policy implementation details and status, as well as non-compliant resources, alarm events, and forcible interception events. You can evaluate cluster compliance based on the data and take measures in a timely manner.

Figure 6-2 Policy implementation details



NOTE

- Currently, it takes about 15 to 30 minutes to report non-compliant resource statistics.
- If non-compliant resources are not blocked or alarms are not generated after a policy instance is delivered, check whether the feature gate ValidatingAdmissionPolicy is enabled and whether the admission controllers ValidatingAdmissionWebhook and MutatingAdmissionWebhook are enabled. For details, see **What does each admission controller do?**

6.5 Example: Using Policy Center for Kubernetes Resource Compliance Governance

Assume that you are a platform engineer of a large enterprise. You are responsible for configuring and managing security policies for the entire infrastructure to ensure compliance of the cluster resources. With the UCS Policy Center, you can:

- Create a unified policy instance that contains the security and compliance regulations that all teams need to comply with. In this way, you can ensure that all teams follow the same standards when using cluster resources.
- Deploy policies automatically as the system automatically applies these policies to clusters, improving efficiency and accuracy.
- Monitor policy implementation and quickly detect and solve problems during policy implementation.

This section describes how to use Policy Center to implement compliance governance for Kubernetes resources. The process is as follows:

Enabling the Policy Center

Step 1 Log in to the UCS console. In the navigation pane, choose **Policy Center**.

Step 2 Click **Enable**. The **Enable Policy Management** dialog box is displayed.

Step 3 Select a fleet or cluster from the drop-down list and click **OK** to return to the policy center.

You will see that policy management is being enabled. Wait for about 3 minutes.

----End

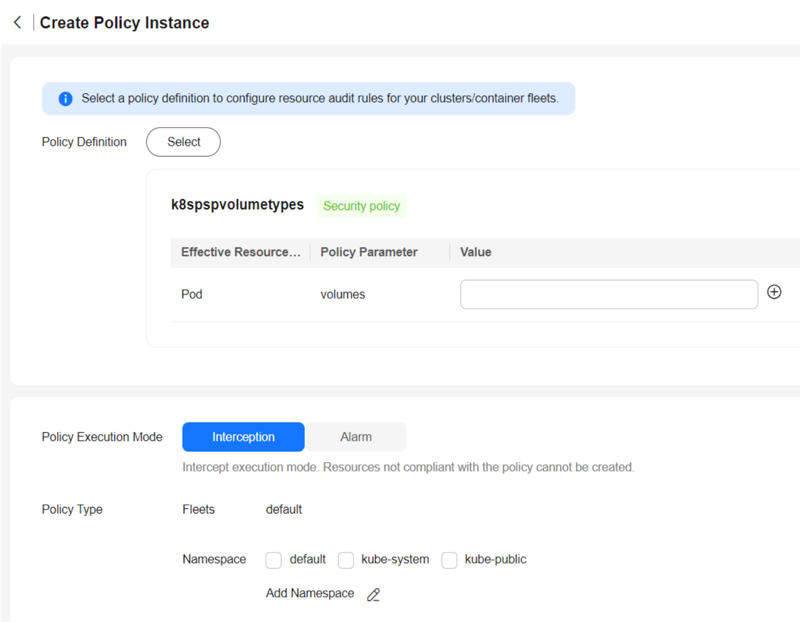
Creating a Policy Instance

Step 1 Log in to the UCS console. In the navigation pane, choose **Policy Center**.

Step 2 In the list, find the fleet or cluster for which the policy center function has been enabled and click **Create Policy Instance**.

Step 3 Set the following parameters:

Figure 6-3 Creating a policy instance



- **Policy Definition**: Select one from the 33 built-in policy definitions. This section uses **k8srequiredlabels** as an example. This policy definition requires resources to contain specified labels, with values matching the provided regular expressions. In this example, the label key is set to **owner**, and the regular expression is **^[a-zA-Z]+.agilebank.demo$**.
- **Policy Execution Mode**: **Interception** and **Alarm** are supported. Interception indicates that resources that do not comply with the policy cannot be created. Alarm indicates that resources that do not comply with the policy can still be created. This section uses **Interception** as an example.
- **Policy Type**: Select the namespace where the policy takes effect. This section uses **default** as an example.

Step 4 Click **Create**. After the policy is created, the system automatically distributes the policy. If the distribution is successful, the policy instance takes effect in the cluster.

----End

Verifying the Policy Instance

After the policy instance is successfully distributed, the action that complies with the policy instance can be executed in the cluster. If the action that does not comply with the policy instance is executed in the cluster, the action will be rejected (depending on the configured policy execution mode).

Create a pod in the cluster and define the label as **owner: user.agilebank.demo**. The pod complies with the policy instance can be created.

If the label defined in the policy instance is not included during pod creation, the pod fails to be created, and the corresponding record is generated on the **Non-Compliant Resources** tab page.

6.6 Policy Definition Library

6.6.1 Overview

UCS provides you with a preset policy definition library. With this library, you can create specific policy instances and delegate the task of defining policy instance details to individuals or teams with professional knowledge. This approach not only isolates concerns, but also separates the logic of policy instances from their definitions.

To help you better understand the working principle of a policy definition, each preset policy definition contains the following three parts: an example policy instance, which is used to show how to use the policy definition; a resource definition that violates the policy instance, which is used to describe the resource examples that do not meet the policy requirements; a resource definition that meets the policy instance, which is used to display resource examples that meet the policy requirements.

Each policy instance contains a **match** field, which defines the target object to which the policy instance is applied. The **match** field specifies the resource type, namespace, or other specific conditions to which the policy instance applies. This ensures that the policy instance takes effect only on the objects that meet these conditions.

Table 6-2 defines 16 security policies, which are used to ensure the security of clusters and resources. **Table 6-3** defines 17 compliance policies, which are used to meet different compliance requirements.

Table 6-2 Security policy definition

Policy Definition	Type	Level of Recommendation	Target Object	Parameter
k8spspvolumetypes	Security	L3	Pods	volumes: Array

Policy Definition	Type	Level of Recommendation	Target Object	Parameter
k8spspallowedusers	Security	L3	Pods	exemptImages: String array runAsUser ● rule: String ● ranges – min: Integer – max: Integer runAsGroup ● rule: String ● ranges – min: Integer – max: Integer supplementalGroups ● rule: String ● ranges – min: Integer – max: Integer fsGroup ● rule: String ● ranges – min: Integer – max: Integer
k8spspselinuxv2	Security	L3	Pods	allowedSELinuxOptions: Object array, including four string objects: level, role, type, and user. exemptImages: String array
k8spspseccomp	Security	L3	Pods	allowedLocalhostFiles: Array allowedProfiles: Array exemptImages: String array
k8spspreadonlyrootfilesystem	Security	L3	Pods	exemptImages: String array
k8spspprocmount	Security	L3	Pods	exemptImages: String array procMount: String

Policy Definition	Type	Level of Recommendation	Target Object	Parameter
k8spspprivilegedcontainer	Security	L3	Pods	exemptImages: String array
k8spsphostnetworkingports	Security	L3	Pods	exemptImages: String array hostNetwork ● max: Integer ● min: Integer
k8spsphostnamespace	Security	L3	Pods	None
k8spsphostfilesystem	Security	L3	Pods	allowedHostPaths ● pathPrefix: String
k8spspfsgroup	Security	L3	Pods	rule: The value is a string. MayRunAs, MustRunAs, and RunAsAny are supported. ranges ● max: Integer ● min: Integer
k8spspforbiddensysctls	Security	L3	Pods	allowedSysctls: Array forbiddenSysctls: Array
k8spspflexvolumes	Security	L3	Pods	allowedFlexVolumes: Array
k8spspcapabilities	Security	L3	Pods	allowedCapabilities: Array exemptImages: String array requiredDropCapabilities: Array
k8spspapparmor	Security	L3	Pods	allowedProfiles: Array exemptImages: String array
k8spspallowprivilegeescalationcontainer	Security	L3	Pods	exemptImages: String array

Table 6-3 Definition of compliance policies

Policy Definition	Type	Level of Recommendation	Target Object	Parameter
k8srequiredprobes	Compliance	L1	Pods	probes: Array probeTypes: Array
k8srequiredlabels	Compliance	L1	Deployment	labels ● key/allowedRegex: Key-value pair array message: String
k8srequiredannotations	Compliance	L1	Pods	annotations ● key/allowedRegex: Key-value pair array message: String
k8sreplicalimits	Compliance	L1	Deployment, ReplicaSet, and CronJob	ranges ● min_replicas: Integer ● max_replicas: Integer
noupdateserviceaccount	Compliance	L1	Pods	allowedGroups: Array allowedUsers: Array
k8simagedigests	Compliance	L1	Pods	exemptImages: String array
k8sexternalips	Compliance	L1	Service	allowedIPs: String array
k8sdisallowedtags	Compliance	L1	Pods	tags: String array exemptImages: String array
k8srequiredresources	Compliance	L1	Pods	exemptImages: String array limits ● cpu ● memory requests ● cpu ● memory
k8scontainerratios	Compliance	L1	Pods	ratio: String cpuRatio: String exemptImages: String array

Policy Definition	Type	Level of Recommendation	Target Object	Parameter
k8scontainerrequests	Compliance	L1	Pods	cpu: String memory: String exemptImages: String array
k8scontainerlimits	Compliance	L1	Pods	cpu: String memory: String exemptImages: String array
k8sblockwildcardingress	Compliance	L1	Ingress	None
k8sblocknodeport	Compliance	L1	Service	None
k8sblockloadbalancer	Compliance	L1	Pods	None
k8spspautomountserviceaccounttokenpod	Compliance	L1	Pods	None
k8sallowedrepos	Compliance	L1	Pods	repos: String array

6.6.2 k8spspvolumetypes

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
  volumes: Array

Function

This policy restricts the type of the **volumes** field in PodSecurityPolicy.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. The **volumes** field of **parameters** defines the allowed types.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPVolumeTypes
metadata:
  name: psp-volume-types
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    volumes:
    # - "*" # * may be used to allow all volume types
    - configMap
    - emptyDir
    - projected
    - secret
    - downwardAPI
    - persistentVolumeClaim
    #- hostPath #required for allowedHostPaths
    - flexVolume #required for allowedFlexVolumes
```

Resource Definition That Complies with the Policy

In the example, the types in volumes are within the preceding range and comply with the policy.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-volume-types-allowed
  labels:
    app: nginx-volume-types
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  - name: nginx2
    image: nginx
    volumeMounts:
    - mountPath: /cache2
      name: demo-vol
  volumes:
  - name: cache-volume
    emptyDir: {}
  - name: demo-vol
    emptyDir: {}
```

Resource Definition That Does Not Comply with the Policy

In the example, the type (hostPath) in **volumes** is not within the preceding range and does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-volume-types-disallowed
  labels:
```

```
    app: nginx-volume-types
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  - name: nginx2
    image: nginx
    volumeMounts:
    - mountPath: /cache2
      name: demo-vol
  volumes:
  - name: cache-volume
    hostPath:
      path: /tmp # directory location on host
  - name: demo-vol
    emptyDir: {}
```

6.6.3 k8spspallowedusers

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

```
exemptImages: String array
runAsUser:
  rule: String
  ranges:
    - min: Integer
      max: Integer
runAsGroup:
  rule: String
  ranges:
    - min: Integer
      max: Integer
supplementalGroups:
  rule: String
  ranges:
    - min: Integer
      max: Integer
fsGroup:
  rule: String
  ranges:
    - min: Integer
      max: Integer
```

Function

This policy restricts the **runAsUser**, **runAsGroup**, **supplementalGroups**, and **fsGroup** fields in PodSecurityPolicy.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **parameters** defines constraints on fields such as **runAsUser**, **runAsGroup**, **supplementalGroups**, and **fsGroup**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAllowedUsers
```

```
metadata:
  name: psp-pods-allowed-user-ranges
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    runAsUser:
      rule: MustRunAs # MustRunAsNonRoot # RunAsAny
      ranges:
        - min: 100
          max: 200
    runAsGroup:
      rule: MustRunAs # MayRunAs # RunAsAny
      ranges:
        - min: 100
          max: 200
    supplementalGroups:
      rule: MustRunAs # MayRunAs # RunAsAny
      ranges:
        - min: 100
          max: 200
    fsGroup:
      rule: MustRunAs # MayRunAs # RunAsAny
      ranges:
        - min: 100
          max: 200
```

Resource Definition That Complies with the Policy

In the example, parameters such as **runAsUser** are within the range and comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-users-allowed
  labels:
    app: nginx-users
spec:
  securityContext:
    supplementalGroups:
      - 199
    fsGroup: 199
  containers:
    - name: nginx
      image: nginx
      securityContext:
        runAsUser: 199
        runAsGroup: 199
```

Resource Definition That Does Not Comply with the Policy

In the example, parameters such as **runAsUser** are not within the range and do not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-users-disallowed
  labels:
    app: nginx-users
spec:
  securityContext:
    supplementalGroups:
      - 250
```



```
  fsGroup: 250
 containers:
  - name: nginx
   image: nginx
   securityContext:
    runAsUser: 250
    runAsGroup: 250
```

6.6.4 k8spspselinuxv2

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  allowedSELinuxOptions: Object array, including four string objects: level, role, type, and user.

  exemptImages: String array

Function

This policy restricts the SELinux configurations in a pod.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **allowedSELinuxOptions** in **parameters** defines the allowed parameters.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPSELinuxV2
metadata:
 name: psp-selinux-v2
spec:
 match:
  kinds:
   - apiGroups: [""]
    kinds: ["Pod"]
 parameters:
  allowedSELinuxOptions:
   - level: s0:c123,c456
    role: object_r
    type: svirt_sandbox_file_t
    user: system_u
```

Resource Definition That Complies with the Policy

In the example, the parameters of **seLinuxOptions** are displayed in the parameter list and complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-selinux-allowed
  labels:
    app: nginx-selinux
spec:
 containers:
```

```
- name: nginx
  image: nginx
  securityContext:
    seLinuxOptions:
      level: s0:c123,c456
      role: object_r
      type: svirt_sandbox_file_t
      user: system_u
```

Resource Definition That Does Not Comply with the Policy

In the example, the parameters of **seLinuxOptions** are not displayed in the parameter list and does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx-selinux-disallowed
    labels:
        app: nginx-selinux
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      seLinuxOptions:
        level: s1:c234,c567
        user: sysadm_u
        role: sysadm_r
        type: svirt_lxc_net_t
```

6.6.5 k8spspseccomp

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
  allowedLocalhostFiles: Array
  allowedProfiles: Array
  exemptImages: String array

Function

This policy restricts the **seccomp.security.alpha.kubernetes.io/allowedProfileNames** annotation in PodSecurityPolicy.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **allowedProfiles** in **parameters** defines the allowed annotations.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPSeccomp
metadata:
  name: psp-seccomp
spec:
```

```
match:
  kinds:
    - apiGroups: [""]
      kinds: ["Pod"]
parameters:
  allowedProfiles:
  - runtime/default
  - docker/default
```

Resource Definition That Complies with the Policy

In the example, the value of the **container.seccomp.security.alpha.kubernetes.io/nginx** annotation is in the specified value list and complies with the policy definition.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-seccomp-allowed
  annotations:
    container.seccomp.security.alpha.kubernetes.io/nginx: runtime/default
  labels:
    app: nginx-seccomp
spec:
  containers:
  - name: nginx
    image: nginx
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of the **container.seccomp.security.alpha.kubernetes.io/nginx** annotation is not in the configured value list and does not comply with the policy definition.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-seccomp-disallowed
  annotations:
    container.seccomp.security.alpha.kubernetes.io/nginx: unconfined
  labels:
    app: nginx-seccomp
spec:
  containers:
  - name: nginx
    image: nginx
```

6.6.6 k8spspreadonlyrootfilesystem

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  exemptImages: String array

Function

This policy restricts the **readOnlyRootFilesystem** field in PodSecurityPolicy.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPReadOnlyRootFilesystem
metadata:
  name: psp-readonlyrootfilesystem
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

Resource Definition That Complies with the Policy

In the example, the value of the **readOnlyRootFilesystem** field is **true**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-readonlyrootfilesystem-allowed
  labels:
    app: nginx-readonlyrootfilesystem
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      readOnlyRootFilesystem: true
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of the **readOnlyRootFilesystem** field is **false**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-readonlyrootfilesystem-disallowed
  labels:
    app: nginx-readonlyrootfilesystem
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      readOnlyRootFilesystem: false
```

6.6.7 k8spspprocmount

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  exemptImages: String array

  procMount: String

Function

This policy restricts the **allowedProcMountTypes** field in PodSecurityPolicy.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. In **parameters**, the value of **procMount** is set to **Default**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPProcMount
metadata:
  name: psp-proc-mount
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    procMount: Default
```

Resource Definition That Complies with the Policy

In the example, **procMount** in the **securityContext** field is **Default**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-proc-mount-disallowed
  labels:
    app: nginx-proc-mount
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      procMount: Default
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of **procMount** in the **securityContext** field is **Unmasked**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-proc-mount-disallowed
  labels:
    app: nginx-proc-mount
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      procMount: Unmasked
```

6.6.8 k8spspprivilegedcontainer

Basic Information

- Policy type: security

- Recommended level: L3
- Effective resource type: Pod
- Parameter
  exemptImages: String array

Function

The **privileged** field in PodSecurityPolicy cannot be set to **true**.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPPrivilegedContainer
metadata:
  name: psp-privileged-container
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    excludedNamespaces: ["kube-system"]
```

Resource Definition That Complies with the Policy

In the example, the value of **privileged** is set to **false**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privileged-allowed
  labels:
    app: nginx-privileged
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: false
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of **privileged** is set to **true**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privileged-disallowed
  labels:
    app: nginx-privileged
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
```

6.6.9 k8spsphostnetworkingports

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
```
exemptImages: String array
hostNetwork:
  max: Integer
  min: Integer
```

Function

The **hostNetwork** and **hostPorts** fields in PodSecurityPolicy are restricted.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. If **hostNetwork** in **parameters** is set to **true**, the used port must be within the specified port range.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostNetworkingPorts
metadata:
  name: psp-host-network-ports
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    hostNetwork: bool
    min: 80
    max: 9000
```

Resource Definition That Complies with the Policy

In the example, **hostNetwork** is set to **false**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-networking-ports-allowed
  labels:
    app: nginx-host-networking-ports
spec:
  hostNetwork: false
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 9000
      hostPort: 80
```


Resource Definition That Does Not Comply with the Policy

In the example, **hostNetwork** is set to **true**, but the port number is not in the specified range, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-networking-ports-disallowed
  labels:
    app: nginx-host-networking-ports
spec:
  hostNetwork: true
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 9001
      hostPort: 9001
```

6.6.10 k8spsphostnamespace

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter: None

Function

The **hostPID** and **hostIPC** fields in PodSecurityPolicy are restricted.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostNamespace
metadata:
  name: psp-host-namespace
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

Resource Definition That Complies with the Policy

In the example, the values of **hostPID** and **hostIPC** are **false**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-namespace-allowed
  labels:
    app: nginx-host-namespace
spec:
  hostPID: false
```

```
  hostIPC: false
  containers:
  - name: nginx
    image: nginx
```

Resource Definition That Does Not Comply with the Policy

In the example, the values of **hostPID** and **hostIPC** are **true**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-namespace-disallowed
  labels:
    app: nginx-host-namespace
spec:
  hostPID: true
  hostIPC: true
  containers:
  - name: nginx
    image: nginx
```

6.6.11 k8spsphostfilesystem

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
```
allowedHostPaths:
  readOnly: Boolean
  pathPrefix: String
```

Function

The parameters of the **hostPath** field in PodSecurityPolicy are restricted.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **allowedHostPaths** in **parameters** specifies the value of the field.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostFilesystem
metadata:
  name: psp-host-filesystem
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedHostPaths:
    - readOnly: true
      pathPrefix: "/foo"
```

Resource Definition That Complies with the Policy

In the example, **pathPrefix** in **hostPath** starts with **/foo**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem-disallowed
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - mountPath: /cache
          name: cache-volume
          readOnly: true
  volumes:
    - name: cache-volume
      hostPath:
        path: /foo/bar
```

Resource Definition That Does Not Comply with the Policy

In the example, **pathPrefix** in **hostPath** starts with **/tmp**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem-disallowed
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
      readOnly: true
  volumes:
  - name: cache-volume
    hostPath:
      path: /tmp # directory location on host
```

6.6.12 k8spspfsgroup

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

```
rule: String. MayRunAs, MustRunAs, and RunAsAny are supported.
ranges
  max: Integer
  min: Integer
```

Function

This policy ensures that the value of the **fsGroup** field in PodSecurityPolicy is within a specified range.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPFSGroup
metadata:
  name: psp-fsgroup
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    rule: "MayRunAs" #"MustRunAs" #"MayRunAs", "RunAsAny"
    ranges:
    - min: 1
      max: 1000
```

Resource Definition That Complies with the Policy

In the example, the value of **fsGroup** is set to **500**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: fsgroup-disallowed
spec:
  securityContext:
    fsGroup: 500 # directory will have group ID 500
  volumes:
    - name: fsgroup-demo-vol
      emptyDir: {}
  containers:
    - name: fsgroup-demo
      image: busybox
      command: ["sh", "-c", "sleep 1h"]
      volumeMounts:
        - name: fsgroup-demo-vol
          mountPath: /data/demo
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of **fsGroup** is set to **2000**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: fsgroup-disallowed
spec:
  securityContext:
    fsGroup: 2000   # directory will have group ID 2000
  volumes:
  - name: fsgroup-demo-vol
    emptyDir: {}
  containers:
```



```
- name: fsgroup-demo
  image: busybox
  command: [ "sh", "-c", "sleep 1h" ]
  volumeMounts:
  - name: fsgroup-demo-vol
    mountPath: /data/demo
```

6.6.13 k8spspforbiddensysctls

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  allowedSysctls: Array

  forbiddenSysctls: Array

Function

This policy specifies the names that are not allowed in the **sysctls** field in PodSecurityPolicy.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. **forbiddenSysctls** in **parameters** defines the names that are not allowed in **sysctls**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPForbiddenSysctls
metadata:
  name: psp-forbidden-sysctls
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    forbiddenSysctls:
    # - "*" # * may be used to forbid all sysctls
    - kernel.*
```

Resource Definition That Complies with the Policy

In the example, the name of **sysctls** complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-forbidden-sysctls-disallowed
  labels:
    app: nginx-forbidden-sysctls
spec:
  containers:
    - name: nginx
      image: nginx
  securityContext:
    sysctls:
```

```
    - name: net.core.somaxconn
      value: "1024"
```

Resource Definition That Does Not Comply with the Policy

In the example, the name (**kernel.msgmax**) of **sysctls** does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-forbidden-sysctls-disallowed
  labels:
    app: nginx-forbidden-sysctls
spec:
  containers:
    - name: nginx
      image: nginx
  securityContext:
    sysctls:
      - name: kernel.msgmax
        value: "65536"
      - name: net.core.somaxconn
        value: "1024"
```

6.6.14 k8spspflexvolumes

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
  allowedFlexVolumes: Array

Function

This policy restricts the **allowedFlexVolumes** field type in PodSecurityPolicy.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. The **allowedFlexVolumes** field in **parameters** defines the allowed driver types.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPFlexVolumes
metadata:
  name: psp-flexvolume-drivers
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedFlexVolumes: #[]
    - driver: "example/lvm"
    - driver: "example/cifs"
```

Resource Definition That Complies with the Policy

In the example, the type in **flexVolume** is within the preceding range and complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-flexvolume-driver-allowed
  labels:
    app: nginx-flexvolume-driver
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /test
      name: test-volume
      readOnly: true
  volumes:
  - name: test-volume
    flexVolume:
      driver: "example/lvm"
```

Resource Definition That Does Not Comply with the Policy

In the example, the type in **flexVolume** is not within the preceding range and does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-flexvolume-driver-disallowed
  labels:
    app: nginx-flexvolume-driver
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /test
      name: test-volume
      readOnly: true
  volumes:
  - name: test-volume
    flexVolume:
      driver: "example/testdriver" #"example/lvm"
```

6.6.15 k8spspcapabilities

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  allowedCapabilities: Array

  exemptImages: String array

  requiredDropCapabilities: Array

Function

The **allowedCapabilities** and **requiredDropCapabilities** fields in PodSecurityPolicy are restricted.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. The **allowedCapabilities** and **requiredDropCapabilities** lists are defined in **parameters**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPCapabilities
metadata:
  name: capabilities-demo
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    allowedCapabilities: ["something"]
    requiredDropCapabilities: ["must_drop"]
```

Resource Definition That Complies with the Policy

In this example, the **capabilities** parameters comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      securityContext:
        capabilities:
          add: ["something"]
          drop: ["must_drop", "another_one"]
      resources:
        limits:
          cpu: "100m"
          memory: "30Mi"
```

Resource Definition That Does Not Comply with the Policy

In this example, the **capabilities** parameters do not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
    owner: me.agilebank.demo
```

```
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      securityContext:
        capabilities:
          add: ["disallowedcapability"]
      resources:
        limits:
          cpu: "100m"
          memory: "30Mi"
```

6.6.16 k8spspapparmor

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter
  allowedProfiles: Array
  exemptImages: String array

Function

This policy restricts the **AppArmor** fields.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. The **allowedProfiles** field of **parameters** defines the allowed values.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAppArmor
metadata:
  name: psp-apparmor
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedProfiles:
    - runtime/default
```

Resource Definition That Complies with the Policy

In the example, the value of **apparmor** is within the allowed range defined above, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor-allowed
  annotations:
```

```
    # apparmor.security.beta.kubernetes.io/pod: unconfined # runtime/default
    container.apparmor.security.beta.kubernetes.io/nginx: runtime/default
  labels:
    app: nginx-apparmor
spec:
  containers:
  - name: nginx
    image: nginx
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of **apparmor** is not within the allowed range defined above, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor-disallowed
  annotations:
    # apparmor.security.beta.kubernetes.io/pod: unconfined # runtime/default
    container.apparmor.security.beta.kubernetes.io/nginx: unconfined
  labels:
    app: nginx-apparmor
spec:
  containers:
  - name: nginx
    image: nginx
```

6.6.17 k8spspallowprivilegeescalationcontainer

Basic Information

- Policy type: security
- Recommended level: L3
- Effective resource type: Pod
- Parameter

  exemptImages: String array

Function

This policy sets the value of the **allowPrivilegeEscalation** field in PodSecurityPolicy to **false**.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAllowPrivilegeEscalationContainer
metadata:
  name: psp-allow-privilege-escalation-container
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

Resource Definition That Complies with the Policy

In the example, the value of **allowPrivilegeEscalation** is **false**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privilege-escalation-allowed
  labels:
    app: nginx-privilege-escalation
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      allowPrivilegeEscalation: false
```

Resource Definition That Does Not Comply with the Policy

In the example, the value of **allowPrivilegeEscalation** is not **false**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privilege-escalation-disallowed
  labels:
    app: nginx-privilege-escalation
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      allowPrivilegeEscalation: true
```

6.6.18 k8srequiredprobes

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  probes: Array

  probeTypes: Array

Function

Pods must have readiness or liveness probes.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. The **parameters** area displays **probes** and **probeTypes**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredProbes
metadata:
```

```
  name: must-have-probes
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    probes: ["readinessProbe", "livenessProbe"]
    probeTypes: ["tcpSocket", "httpGet", "exec"]
```

Resource Definition That Complies with the Policy

The pod contains **livenessProbe** and **readinessProbe**, and the probe type is **tcpSocket**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod1
spec:
  containers:
  - name: tomcat
    image: tomcat
    ports:
    - containerPort: 8080
    livenessProbe:
      tcpSocket:
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 10
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Resource Definition That Does Not Comply with the Policy

The pod contains **livenessProbe**, but **probeType** is not defined, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod1
spec:
  containers:
  - name: nginx-1
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    livenessProbe:
      # tcpSocket:
      #   port: 80
      # initialDelaySeconds: 5
      # periodSeconds: 10
    volumeMounts:
    - mountPath: /tmp/cache
      name: cache-volume
  - name: tomcat
    image: tomcat
    ports:
    - containerPort: 8080
    readinessProbe:
```

```
    tcpSocket:
      port: 8080
    initialDelaySeconds: 5
    periodSeconds: 10
volumes:
- name: cache-volume
  emptyDir: {}
```

6.6.19 k8srequiredlabels

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: *
- Parameter

```
labels: array of key-value pairs, key/allowedRegex
  key: a8r.io/owner
    # Matches email address or github user
    allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
```

Function

The resource must contain the specified label whose value matches the provided regular expression.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. **parameters** specifies the restrictions for **message** and **labels**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: all-must-have-owner
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    message: "All namespaces must have an `owner` label that points to your company username"
    labels:
      - key: owner
        allowedRegex: "^[a-zA-Z]+.agilebank.demo$"
```

Resource Definition That Complies with the Policy

The example contains the label defined in the policy instance, which complies with the policy instance.

```
apiVersion: v1
kind: Namespace
metadata:
  name: allowed-namespace
  labels:
    owner: user.agilebank.demo
```


Resource Definition That Does Not Comply with the Policy

The example does not contain the label defined in the policy instance, which does not comply with the policy instance.

```
apiVersion: v1
kind: Namespace
metadata:
  name: disallowed-namespace
```

6.6.20 k8srequiredannotations

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: *
- Parameter
```
annotations: array of key-value pairs, key/allowedRegex
  key: a8r.io/owner
    # Matches email address or github user
    allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
```

Function

The resource must contain the specified annotations, and the value must match the provided regular expression.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. **Parameters** specifies the **message** and **annotations** constraints.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredAnnotations
metadata:
  name: all-must-have-certain-set-of-annotations
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    message: "All services must have a `a8r.io/owner` and `a8r.io/runbook` annotations."
    annotations:
      - key: a8r.io/owner
        # Matches email address or github user
        allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
      - key: a8r.io/runbook
        # Matches urls including or not http/https
        allowedRegex: ^(http:\/\/www\.|https:\/\/www\.|http:\/\/|https:\/\/)?[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]
{2,5}(:[0-9]{1,5})?(\/.*)?$
```

Resource Definition That Complies with the Policy

The annotations in the example comply with the policy instance.

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: allowed-service
  annotations:
    a8r.io/owner: "dev-team-alfa@contoso.com"
    a8r.io/runbook: "https://confluence.contoso.com/dev-team-alfa/runbooks"
spec:
  ports:
  - name: http
    port: 80
    targetPort: 8080
  selector:
    app: foo
```

Resource Definition That Does Not Comply with the Policy

In the example, no value is configured for annotations, which does not comply with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-service
spec:
  ports:
  - name: http
    port: 80
    targetPort: 8080
  selector:
    app: foo
```

6.6.21 k8sreplicalimits

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: *
- Parameter

  ```
  ranges:
    min_replicas: Integer
    max_replicas: Integer
  ```

Function

Objects (such as Deployments and ReplicaSets) with the **spec.replicas** field must be within the defined range.

Policy Example

The following policy instance shows the resource types for which the policy definition takes effect. The value of **parameters** ranges from **3** to **50**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sReplicaLimits
metadata:
  name: replica-limits
spec:
  match:
    kinds:
      - apiGroups: ["apps"]
        kinds: ["Deployment"]
```

```
parameters:
  ranges:
  - min_replicas: 3
    max_replicas: 50
```

Resource Definition That Complies with the Policy

replicas is set to **3**, which complies with the policy instance.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: allowed-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Resource Definition That Does Not Comply with the Policy

replicas is set to **100**, which does not comply with the policy instance.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: disallowed-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 100
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

6.6.22 noupdateserviceaccount

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: *
- Parameter
  allowedGroups: Array

allowedUsers: Array

Function

The resources that are not in the whitelist are rejected to update ServiceAccount.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **parameters** defines the allowed group list **allowedGroups** and allowed user list **allowedUsers**.

```
# IMPORTANT: Before deploying this policy, make sure you allow-list any groups
# or users that need to deploy workloads to kube-system, such as cluster-
# lifecycle controllers, addon managers, etc. Such controllers may need to
# update service account names during automated rollouts (e.g. of refactored
# configurations). You can allow-list them with the allowedGroups and
# allowedUsers properties of the NoUpdateServiceAccount Constraint.
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: NoUpdateServiceAccount
metadata:
  name: no-update-kube-system-service-account
spec:
  match:
    namespaces: ["kube-system"]
    kinds:
    - apiGroups: [""]
      kinds:
      # You can optionally add "Pod" here, but it is unnecessary because
      # Pod service account immutability is enforced by the Kubernetes API.
      - "ReplicationController"
    - apiGroups: ["apps"]
      kinds:
      - "ReplicaSet"
      - "Deployment"
      - "StatefulSet"
      - "DaemonSet"
    - apiGroups: ["batch"]
      kinds:
      # You can optionally add "Job" here, but it is unnecessary because
      # Job service account immutability is enforced by the Kubernetes API.
      - "CronJob"
  parameters:
    allowedGroups: []
    allowedUsers: []
```

Resource Definition That Complies with the Policy

The ServiceAccount is not updated, which complies with the policy instance.

```
# Note: The gator tests currently require exactly one object per example file.
# Since this is an update-triggered policy, at least two objects are technically
# required to demonstrate it. Due to the gator requirement, we only have one
# object below. The policy should allow changing everything but the
# serviceAccountName field.
kind: Deployment
apiVersion: apps/v1
metadata:
  name: policy-test
  namespace: kube-system
  labels:
    app: policy-test
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: policy-test-deploy
  template:
    metadata:
      labels:
        app: policy-test-deploy
    spec:
      # Changing anything except this field should be allowed by the policy.
      serviceAccountName: policy-test-sa-1
      containers:
      - name: policy-test
        image: ubuntu
        command:
        - /bin/bash
        - -c
        - sleep 99999
```

6.6.23 k8simagedigests

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  exemptImages: String array

Function

The container image must contain **digest**.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sImageDigests
metadata:
  name: container-image-must-have-digest
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
```

Resource Definition That Complies with the Policy

The container image contains **digest**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/
opa:0.9.2@sha256:04ff8fce2afd1a3bc26260348e5b290e8d945b1fad4b4c16d22834c2f3a1814a
```

Resource Definition That Does Not Comply with the Policy

The container image does not contain **digest**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
```

6.6.24 k8sexternalips

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Service
- Parameter
  allowedIPs: String array

Function

The external IP of the Service must be an allowed IP address.

Policy Example

The external IP of the Service can only be the IP address defined in **allowedIPs**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sExternalIPs
metadata:
  name: external-ips
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    allowedIPs:
      - "203.0.113.0"
```

Resource Definition That Complies with the Policy

The IP addresses in **externalIPs** are those in the allowed IP address list, which complies with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: allowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
```



```
    port: 80
    targetPort: 8080
  externalIPs:
    - 203.0.113.0
```

Resource Definition That Does Not Comply with the Policy

The IP addresses in **externalIPs** are not in the allowed IP address list, which does not comply with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8080
  externalIPs:
    - 1.1.1.1
```

6.6.25 k8sdisallowedtags

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  tags: String array

  exemptImages: String array

Function

This policy restricts the container image tag.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect. **parameters** indicates that the container image tag cannot be **latest**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sDisallowedTags
metadata:
  name: container-image-must-not-have-latest-tag
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    tags: ["latest"]
    exemptImages: ["openpolicyagent/opa-exp:latest", "openpolicyagent/opa-exp2:latest"]
```


Resource Definition That Complies with the Policy

The container image tag is not **latest**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
```

Resource Definition That Does Not Comply with the Policy

The container image tag is **latest**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed-2
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:latest
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
```

6.6.26 k8sdisallowanonymous

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: RoleBinding and ClusterRoleBinding
- Parameter

  allowedRoles: String array

Function

ClusterRole and Role that are not in the whitelist cannot be associated with **system:anonymous User** and **system:unauthenticated Group**.

Policy Example

The policy instance shows that **ClusterRole** and **Role** resources can be associated only with roles defined in **allowedRoles**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sDisallowAnonymous
metadata:
  name: no-anonymous
spec:
  match:
```

```
  kinds:
    - apiGroups: ["rbac.authorization.k8s.io"]
      kinds: ["ClusterRoleBinding"]
    - apiGroups: ["rbac.authorization.k8s.io"]
      kinds: ["RoleBinding"]
 parameters:
  allowedRoles:
    - cluster-role-1
```

Resource Definition That Complies with the Policy

ClusterRole is associated with **cluster-role-1 Role** and complies with the policy instance.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-role-1
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:unauthenticated
```

Resource Definition That Does Not Comply with the Policy

ClusterRole is associated with **cluster-role-2 Role**, which does not comply with the policy instance.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-2
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-role-2
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:unauthenticated
```

6.6.27 k8srequiredresources

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

```
exemptImages: String array
limits
  cpu
  memory
```

```
requests
  cpu
  memory
```

Function

This policy restricts container resource usage.

Policy Example

The memory **Limit**, CPU, and memory **Request** must be configured.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredResources
metadata:
  name: container-must-have-cpu-requests-memory-limits-and-requests
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    limits:
      - memory
    requests:
      - cpu
      - memory
```

Resource Definition That Complies with the Policy

The configured memory **Limit**, CPU, and memory **Request** comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          cpu: "100m"
          memory: "1Gi"
        requests:
          cpu: "100m"
          memory: "1Gi"
```

Resource Definition That Does Not Comply with the Policy

The memory **Limit**, CPU, and memory **Request** are not configured, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
```

```
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          memory: "2Gi"k8sexternalips
```

6.6.28 k8scontainerratios

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Service
- Parameter

  ratio: String

  cpuRatio: String

  exemptImages: String array

Function

The external IP of the Service must be an allowed IP address.

Policy Example

The external IP of the Service can only be the IP address defined in **allowedIPs**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sExternalIPs
metadata:
  name: external-ips
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    allowedIPs:
      - "203.0.113.0"
```

Resource Definition That Complies with the Policy

The IP addresses in **externalIPs** are those in the allowed IP address list, which complies with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: allowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
    - name: http
```

```
    protocol: TCP
    port: 80
    targetPort: 8080
  externalIPs:
   - 203.0.113.0
```

Resource Definition That Does Not Comply with the Policy

The IP addresses in **externalIPs** are not in the allowed IP address list, which does not comply with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8080
  externalIPs:
    - 1.1.1.1
```

6.6.29 k8scontainerrequests

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  cpu: String

  memory: String

  exemptImages: String array

Function

This policy requires the CPU and memory **Request** be set and less than the configured maximum value.

Policy Example

This policy instance shows the **Request** configuration of CPU and memory.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sContainerRequests
metadata:
  name: container-must-have-requests
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    cpu: "200m"
    memory: "1Gi"
```

Resource Definition That Complies with the Policy

Request values of the CPU and memory are less than the configured maximum value, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        requests:
          cpu: "100m"
          memory: "1Gi"
```

Resource Definition That Does Not Comply with the Policy

The memory **Request** is greater than the maximum value, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        requests:
          cpu: "100m"
          memory: "2Gi"
```

6.6.30 k8scontainerlimits

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  cpu: String

  memory: String

  exemptImages: String array

Function

The CPU and memory **Limit** must be set for the container and must be less than the maximum values.

Policy Example

The example shows that the maximum CPU usage of the matched object is 200 MB and the maximum memory usage is 1 GB.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sContainerLimits
metadata:
  name: container-must-have-limits
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    cpu: "200m"
    memory: "1Gi"
```

Resource Definition That Complies with the Policy

Limit of the CPU and memory complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          cpu: "100m"
          memory: "1Gi"
```

Resource Definition That Does Not Comply with the Policy

The memory **Limit** exceeds the maximum value, which does not comply with the policy instance.

6.6.31 k8sblockwildcardingress

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Ingress
- Parameter: None

Function

Do not configure a blank or wildcard host name for the Ingress.

Policy Example

The following example shows the effective type of the policy definition.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockWildcardIngress
metadata:
  name: block-wildcard-ingress
spec:
  match:
    kinds:
      - apiGroups: ["extensions", "networking.k8s.io"]
        kinds: ["Ingress"]
```

Resource Definition That Complies with the Policy

The host name configured for the Ingress is not blank or wildcard, which complies with the policy instance.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: non-wildcard-ingress
spec:
  rules:
  - host: 'myservice.example.com'
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

Resource Definition That Does Not Comply with the Policy

The host name configured for the Ingress is blank, which does not comply with the policy instance.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wildcard-ingress
spec:
  rules:
  - host: ''
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wildcard-ingress
```

```
spec:
  rules:
  # Omitted host field counts as a wildcard too
  - http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

The host name configured for the Ingress contains a wildcard (*), which does not comply with the policy instance.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wildcard-ingress
spec:
  rules:
  - host: '*.example.com'
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

6.6.32 k8sblocknodeport

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Service
- Parameter: None

Function

NodePort Services are not allowed.

Policy Example

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockNodePort
metadata:
  name: block-node-port
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
```

Resource Definition That Complies with the Policy

The service type is not **Nodeport**, which complies with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

Resource Definition That Does Not Comply with the Policy

The service type is **Nodeport**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

6.6.33 k8sblockloadbalancer

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Service
- Parameter: None

Function

LoadBalancer Services are not allowed.

Policy Example

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockLoadBalancer
metadata:
  name: block-load-balancer
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
    excludedNamespaces:
      - "ingress-nginx-private"
      - "ingress-nginx-public"
```

Resource Definition That Complies with the Policy

The service type is not **LoadBalancer**, which complies with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-allowed
spec:
```



```
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 80
```

Resource Definition That Does Not Comply with the Policy

The service type is **LoadBalancer**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

6.6.34 k8sblockendpointeditdefaultrole

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: ClusterRole
- Parameter: None

Function

By default, many Kubernetes predefines a **ClusterRole** named **system:aggregate-to-edit**. The **k8sblockendpointeditdefaultrole** policy prohibits the **ClusterRole** from performing create, patch, and update operations on endpoints.

Policy Example

The following policy instance shows the types of resources for which the policy definition takes effect.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockEndpointEditDefaultRole
metadata:
  name: block-endpoint-edit-default-role
spec:
  match:
    kinds:
      - apiGroups: ["rbac.authorization.k8s.io"]
        kinds: ["ClusterRole"]
```

Resource Definition That Complies with the Policy

In the example, the effective object of **ClusterRole** does not contain **endpoints**, which complies with the policy instance.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
```

```
  creationTimestamp: null
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
  name: system:aggregate-to-edit
rules:
- apiGroups:
  - ""
  resources:
  - pods/attach
  - pods/exec
  - secrets
  - services/proxy
  verbs:
  - get
  - list
  - watch
```

Resource Definition That Does Not Comply with the Policy

In the example, the effective object of **ClusterRole** contains **endpoints**, which does not comply with the policy instance.

```
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: null
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
  name: system:aggregate-to-edit
rules:
- apiGroups:
  - apps
  resources:
  - endpoints
  verbs:
  - create
  - delete
  - deletecollection
  - patch
  - update
```

6.6.35 k8spspautomountserviceaccounttokenpod

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter: None

Function

The **automountServiceAccountToken** field cannot be set to **true**.

Policy Example

The example declares that the **automountServiceAccountToken** field cannot be set to **true**.


```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAutomountServiceAccountTokenPod
metadata:
  name: psp-automount-serviceaccount-token-pod
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    excludedNamespaces: ["kube-system"]
```

Resource Definition That Complies with the Policy

The **automountServiceAccountToken** field of the pod is set to **false**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-automountserviceaccounttoken-allowed
  labels:
    app: nginx-not-automountserviceaccounttoken
spec:
  automountServiceAccountToken: false
  containers:
  - name: nginx
    image: nginx
```

Resource Definition That Does Not Comply with the Policy

The **automountServiceAccountToken** field of the pod is set to **true**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-automountserviceaccounttoken-disallowed
  labels:
    app: nginx-automountserviceaccounttoken
spec:
  automountServiceAccountToken: true
  containers:
  - name: nginx
    image: nginx
```

6.6.36 k8sallowedrepos

Basic Information

- Policy type: compliance
- Recommended level: L1
- Effective resource type: Pod
- Parameter

  repos: String array

Function

The container image must start with a string in a specified string list.

Policy Example

The following policy instance specifies that the container image must start with **openpolicyagent/**.

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sAllowedRepos
metadata:
  name: repo-is-openpolicyagent
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    repos:
      - "openpolicyagent/"
```

Resource Definition That Complies with the Policy

The container image starts with **openpolicyagent/**, which complies with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
```

Resource Definition That Does Not Comply with the Policy

The container image starts with **nginx**, which does not comply with the policy instance.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-disallowed
spec:
  containers:
    - name: nginx
      image: nginx
```

7 Configuration Management

Scenarios

Automatic application delivery facilitates application deployment in distributed clusters. UCS configuration management provides the core capability of automatically deploying application configurations from repository resources to Kubernetes clusters. The repository is configured by using **Kustomize organizations and custom resource sets**. Configurations can be distributed and managed across namespaces, clusters, and fleets for Huawei Cloud clusters, multi-cloud clusters, on-premises clusters, and attached clusters. Real-time status observation and message notification are performed for services deployed in each cluster to ensure that application issues can be quickly identified and located, achieving the service level objective (SLO) for end users of customer service apps.

NOTE

Kustomize is a Kubernetes application configuration management tool. It provides a simple and flexible method to generate Kubernetes resources and allows using different ways to configure these resources in different environments. Kustomize also provides hooks that allow performing operations before and after generating resources, such as updating other files based on the generated resources. Kustomize uses a format called Kustomization to describe the configuration of an application. The file is usually named **Kustomization.yaml** and created in the root directory of the application.

7.1 GitOps

Overview

GitOps is a deployment template that uses the Git repository to manage applications. The Git repository is the only source for deploying applications in Kubernetes clusters to achieve continuous application deployment and multi-cluster GitOps delivery, meeting requirements such as high-availability application deployment and distribution of system components across clusters. GitOps assumes that each infrastructure is represented as a file in a storage system with versioning functions, and there is an automated process that seamlessly synchronizes modified applications to the operating environment.

This idea can be better implemented based on declarative APIs and control loops in the Kubernetes ecosystem. This system builds on declarative specifications leading to eventual convergence and consistency.

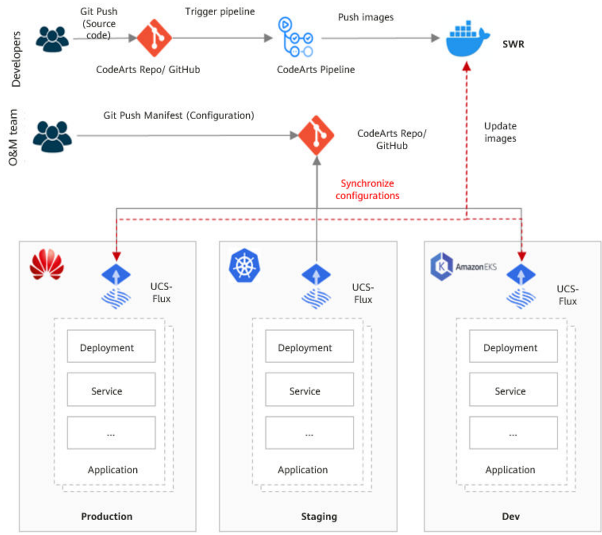
Constraints

Before enabling the configuration management function for a non-Huawei Cloud cluster, ensure that the cluster can pull public network images.

Implementation

- Based on the Git workflow, development and O&M personnel can extend the existing process from application development to deployment, application lifecycle management, and infrastructure configuration. Thanks to the instant availability, customers do not need to maintain the GitOps tool.
- The GitOps plug-in combines the built-in Kustomize with base/overlay artifact organization modes and HelmRelease with valuesFrom/valuesFiles capabilities to meet customers' differentiated configuration management requirements.
- The latest artifact configuration information in the Git repository is synchronized to multiple clusters. Version management and permission control are performed on application release. Release rollback, version iteration control, and audit and tracing are implemented.
- The required infrastructure status is automatically applied to the infrastructure without any manual intervention. The infrastructure is continuously monitored to ensure that it complies with the configuration in the Git repository and works properly.

Figure 7-1 GitOps implementation



Advantages

- Easy usage: Git is easy to be accepted by developers and easy to integrate without extra learning costs.
- High security: Developers do not need any Kubernetes cluster permission for using GitOps and only need the Git repository permission, ensuring cluster security and reliability.
- High reliability: Version management is implemented for the delivery lists of native Kubernetes resources, Helm Chart resources, and Kustomize resources, facilitating application deployment, incremental changes, and application configuration rollback.

- Continuous application deployment: The application statuses in the Kubernetes cluster and Git repository are automatically synchronized to ensure consistency.

Benefits

- Version management is implemented for the delivery lists of native Kubernetes resources, Helm Chart resources, and Kustomize resources, facilitating application deployment, incremental changes, and application configuration rollback.
- Refined differentiated configurations across clusters and environments:
  - The delivery template of the same application component is reused (for example, one connection pool template of the database for multiple business lines) and serves as the best practice template.
  - Operations are more flexible, such as label/string/version number replacement, dynamic parameter embedding, and patching.

7.2 Creating a Configuration Set

Context

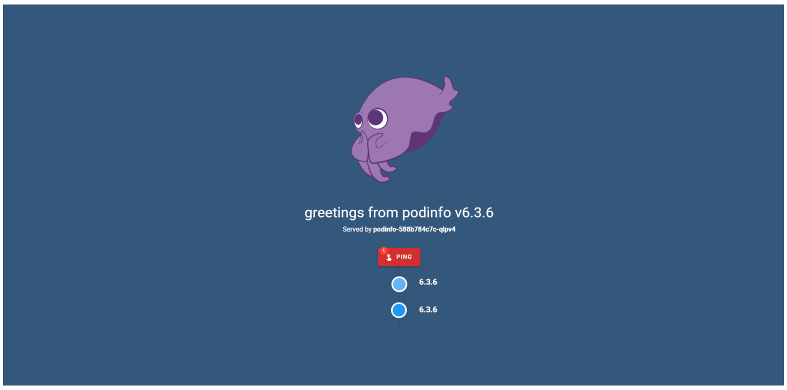
Podinfo is a tiny web application that showcases best practices of running microservices in Kubernetes. It is used for testing and workshops. This chapter uses the podinfo source code as an example to describe how to create a configuration set.

To deliver software more quickly and stably and reduce subsequent maintenance workload, the podinfo source code is stored in the **GitHub repository** and deployed in the cluster by creating a configuration set. GitOps is used for automated software deployment. For details, see **Procedure**.

NOTICE

- When creating a podinfo source code repository, register a GitHub account and fork all podinfo code to your GitHub repository.
- When defining the delivery resource list file in the GitHub repository, ensure that the file does not contain sensitive information (such as the database connection key). Sensitive information must be stored in environment variables or encrypted secrets.

Figure 7-2 Podinfo page



Procedure

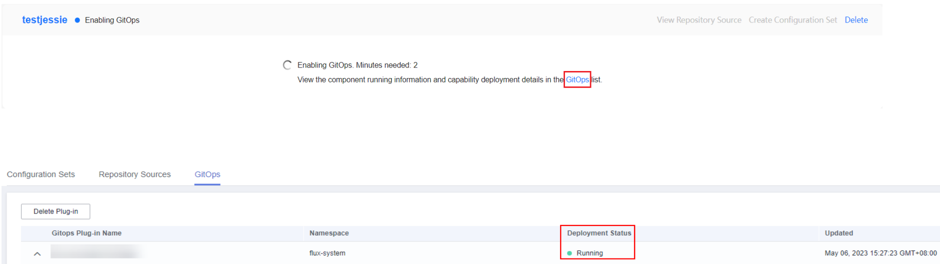
Step 1 Log in to Huawei Cloud Console.

Step 2 Choose **Ubiquitous Cloud Native Service** from **Service List**. In the navigation pane on the left, select **Configuration Management**.

Step 3 Click **Add Cluster** in the upper right corner, select the cluster for which you want to enable configuration management, and click **OK**.

Step 4 In the **Clusters with GitOps Enabled** area, click the cluster name. Then, click the **GitOps** tab and check whether the GitOps plug-in (name: *{Cluster name}*-**FluxPlugin**) has been deployed. If the plug-in deployment status is **Running**, the plug-in has been deployed.

Figure 7-3 Cluster overview



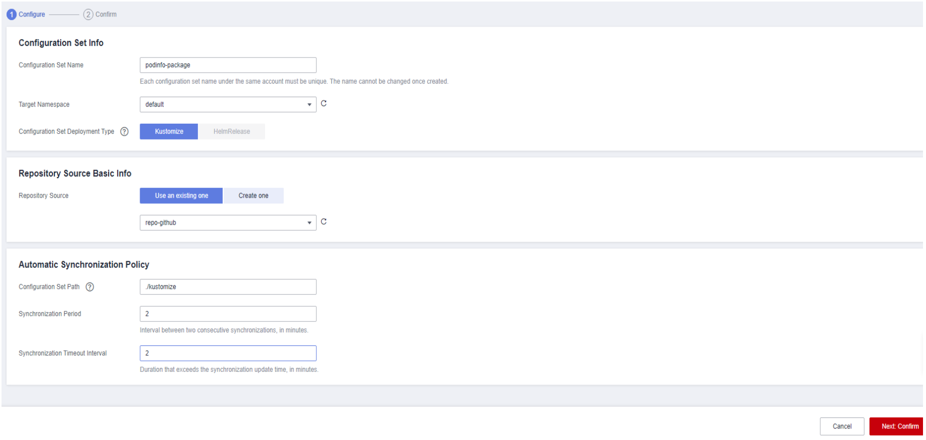
Step 5 Click the **Configuration Sets** tab and click **Create Configuration Set**.

Step 6 Select a repository source. If a repository source already exists, **use the existing repository source**. If you need to **create a repository source**, create one.

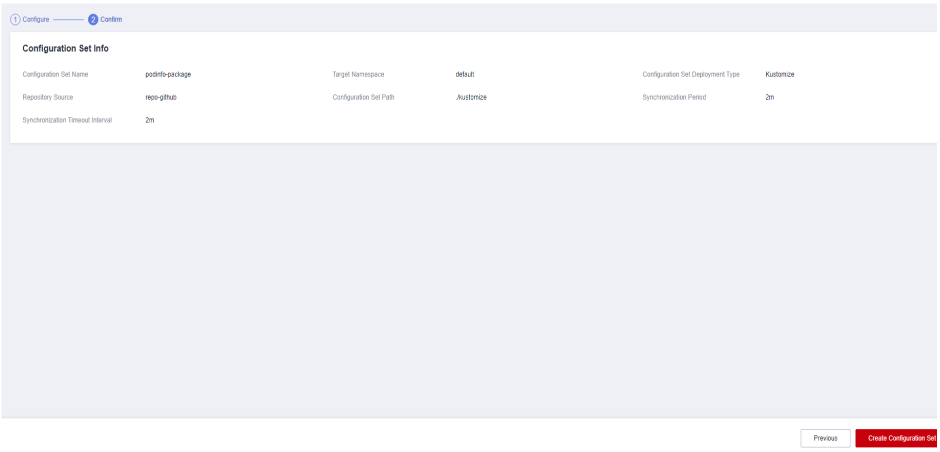
----End

Using the Existing Repository Source

Step 1 Enter the configuration set name, select the target namespace, select **Use an existing one**, and select an existing repository. Enter the configuration set path (top-level path of the configuration set to be synchronized in the repository source) under **Automatic Synchronization Policy**. Then click **Next: Confirm**.



Step 2 After confirming that the configuration information is correct, click Create **Configuration Set**. If the configuration information is incorrect, click **Previous** to modify it.



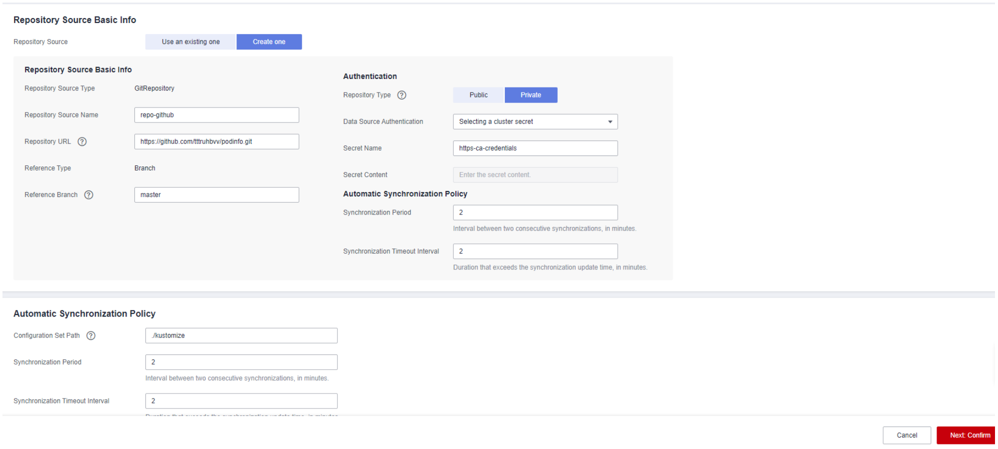
----End

Creating a Repository Source

Step 1 Click **Create one** and enter the repository source name and URL.

Step 2 Enter the code library branch that needs to be synchronized.

Step 3 Select a mode for **Data Source Authentication** and enter the secret.



NOTE

- Public repositories provide read-only permissions with no need for identity authentication.
- If you select a private repository, you can select **Selecting a cluster secret** or **Providing authentication information (SSH)** for **Data Source Authentication**. Both modes require that the configured secret pass the identity authentication.
- For details about how to create a repository secret, see **Keys**.

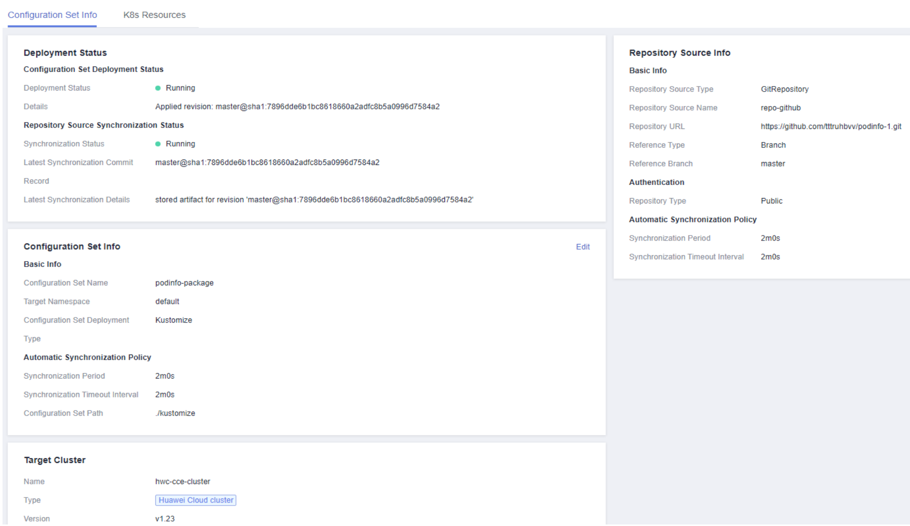
Step 4 After the repository source is created, enter the configuration set path under **Automatic Synchronization Policy** and click **Next: Confirm**.

Step 5 After confirming that the configuration information is correct, click Create **Configuration Set**. If the configuration information is incorrect, click **Previous** to modify it.

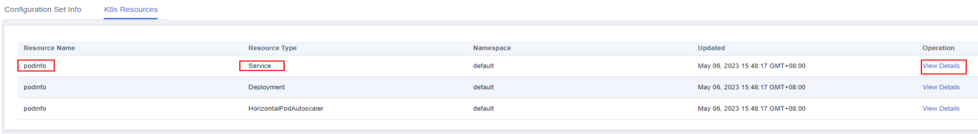
----End

Viewing Configuration Set Information

Step 1 Click the cluster name to go to the configuration management page. Click the configuration set name to view the configuration set information.



Step 2 Click the **K8s Resources** tab to view the resources of the configuration set. Click **View Details** in the **Operation** column to view the details.



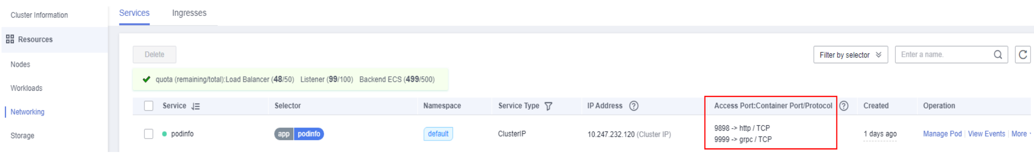
----End

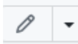
7.3 Modifying the Source Code

Modifying an Application Service

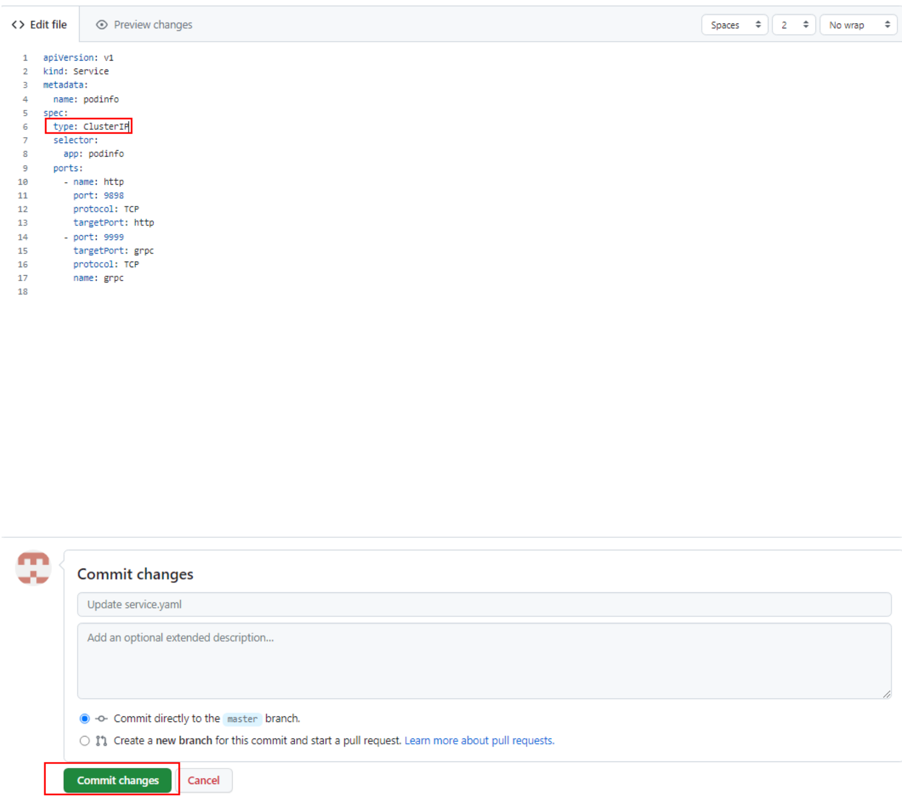
As shown in **Figure 7-4**, you need to change the **Service Type** of the podinfo service in the cluster from **ClusterIP** to **NodePort** and expose the port to the live network.

Figure 7-4 Services

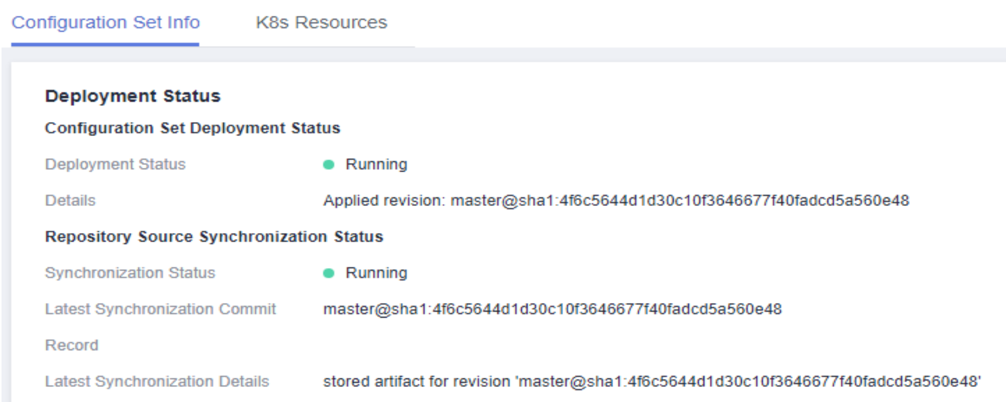


Step 1 Go to the source code repository of the configuration set, find and open the **service.yaml** file in the **podinfo/kustomize** directory according to the repository source information. Click , change **type: ClusterIP** to **type: NodePort**, and click 

to save the change.



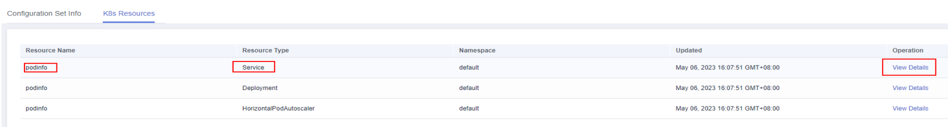
Step 2 Log in to the management console. Choose **Ubiquitous Cloud Native Service** from **Service List** on the left and select **Configuration Management** to go to the cluster where the configuration set to be modified resides. Click the configuration set name to view the configuration set information. If **Synchronization Status** under **Repository Source Synchronization Status** is **Running**, the repository source code has been synchronized. Click **K8s Resources**.



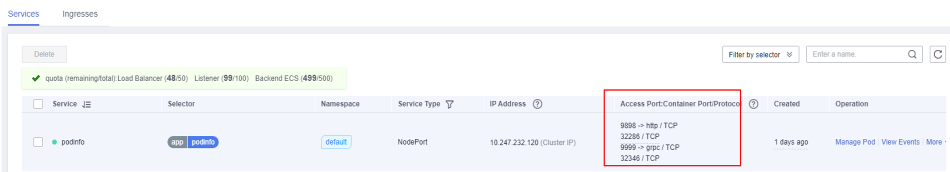
NOTE

After the repository source code is modified, the cluster needs to pull and deploy the application again. This may take several minutes.

Step 3 On the **K8s Resources** page, select the podinfo resource whose **Resource Type** is **Service** and click **View Details** in the **Operation** column.



Step 4 On the **Services** page, view the port number of podinfo in the column **Access Port:Container Port/Protocol**.



Step 5 Enter *Cluster EIP.HTTP port number* (32286 in this case) to access the Service page.

----End

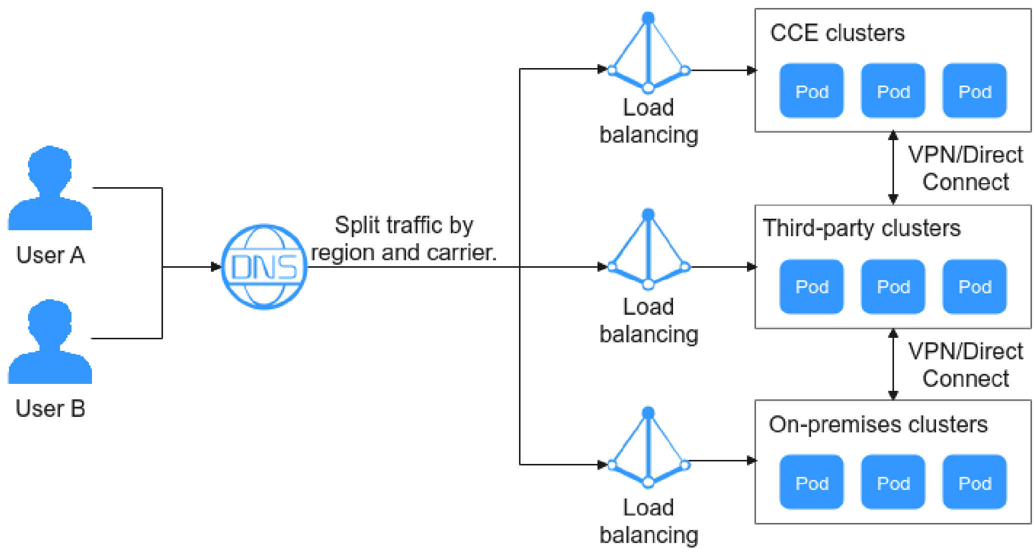
8 Traffic Distribution

8.1 Overview

UCS distributes requests globally according to user locations and service policies across clouds and clusters, implementing intelligent traffic distribution and scheduling. It also schedules application access traffic across domains in real time on demand.

With Domain Name Service (DNS), user requests to the same domain name can be responded to by different backend clusters, according to the users' carrier and region. Such traffic splitting reduces the latency in cross-domain and cross-network access.

Figure 8-1 Traffic management



Prerequisites

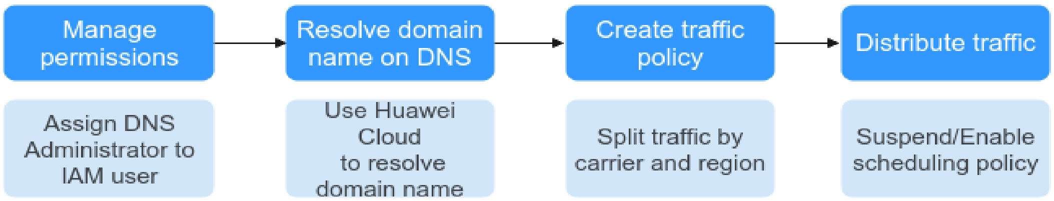
- To manage traffic, IAM users must have the **DNS Administrator** permission.
- You must have a public zone. If not, you need to **buy one**.

- Your public zone has been submitted for ICP license. If not, you need to apply for a license at the **Huawei Cloud ICP License Service**.
- Your public zone can be resolved. If not, you need to **create a DNS record set**.

Procedure

Figure 8-2 shows the process of traffic distribution.

Figure 8-2 Process of traffic distribution



8.2 Creating a Traffic Policy

Step 1 Log in to the UCS console. In the navigation pane, choose **Traffic Distribution**.

Step 2 On the **Traffic Distribution** page, click **Create Traffic Policy**.

Step 3 On the page displayed, enter the domain name and add at least one scheduling policy. To create traffic policies for multiple domain names, repeat **Step 3** to **Step 5**.

- **Domain name**: The domain name prefix can be customized. The suffix is the public zone that has been licensed and has created a Huawei Cloud DNS record set.

  The prefix consists of multiple strings separated by periods (.) and only allows letters, digits, and hyphens (-). Do not start or end with a hyphen (-). The maximum characters of a string are 63, and the maximum total characters of a domain name is 254.

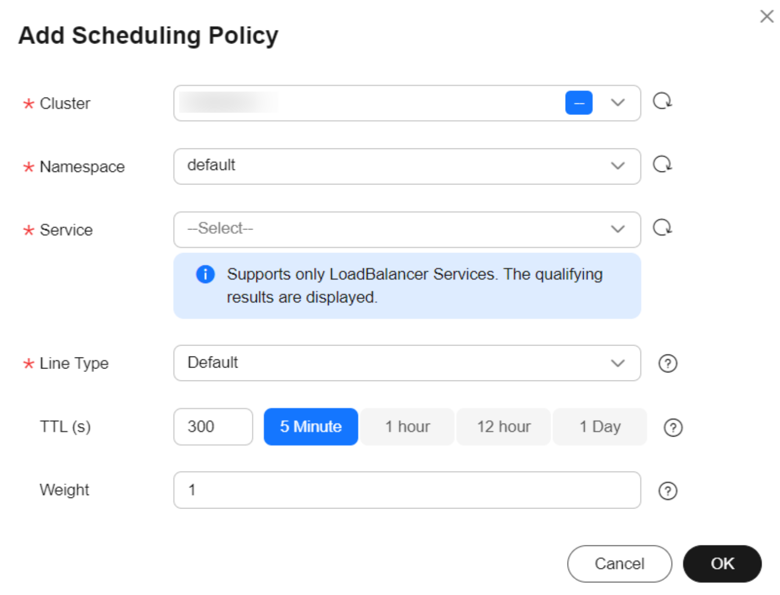
  NOTE

  – If there is no subdomain name, leave the domain name prefix blank.
  – The domain name suffix is the public zone that has been resolved in DNS. You can manage domain names on the DNS console. For details, see **Public Zone Management**.

- **Scheduling Policy**: Traffic can be scheduled based on user locations and service policies. For details, see **Step 4**.

Step 4 Click + to add a scheduling policy and click **OK**, as shown in **Figure 8-3**. To add different scheduling policies for the same domain name, repeat this step. You can also add more scheduling policies later.

Figure 8-3 Adding a scheduling policy



- **Cluster**: Select a cluster in Running state. All clusters taken over by UCS are displayed.
- **Namespace**: namespace that the Service belongs to. The default value is **default**.
- **Service**: Select a Service. Only LoadBalancer Services can be selected.
- **Line Type**:
  - **Default**: (mandatory) returns the default resolution result if no line is matched.
  - **ISP line**: routes visitors to the optimal address based on the carrier networks they use. Defaults to **China Telecom/Default regions**. You can specify a carrier and region down to province.
  - **Region line**: routes visitors to the optimal address based on their geographic locations. The value defaults to **Chinese Mainland/Default regions**. You can select a global region. For **Chinese Mainland**, the region granularity is province. For **Global**, the region granularity is country/region.

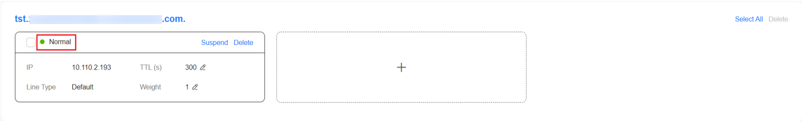
NOTICE

You need to create a **Default** scheduling policy as the default resolution, and then add a custom scheduling policy. If no default line record set is added for the domain name, access to regions beyond the specified line will fail.

- **TTL**: specifies cache duration of the record set on a local DNS server. The default value is **300s/5 minutes**. If your service address changes frequently, set TTL to a smaller value.
- **Weight**: If a resolution line in a domain name contains multiple record sets of the same type, you can set different weights to each record set. For details, see **Configuring Weighted Routing**.

Step 5 Click **Create**. The traffic policy is successfully created.

Figure 8-4 Creating a traffic policy successfully



----End

8.3 Managing Traffic Policies

Suspending a Scheduling Policy

In unexpected scenarios such as traffic switchover, you can suspend an existing scheduling policy and enable it after the fault is rectified. This section describes how to suspend a scheduling policy. Enabling a scheduling policy is the same as suspending a scheduling policy.

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Traffic Distribution**.

Step 2 Locate the scheduling policy and click **Suspend** in the upper right corner.

Step 3 In the dialog box displayed, click **Yes**. The scheduling policy is suspended.

----End

Deleting a Scheduling Policy

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Traffic Distribution**.

Step 2 Locate the scheduling policy and click **Delete** in the upper right corner.

If you want to delete multiple scheduling policies, select them in the upper left corner of the policy box and click **Delete** in the upper right corner of the page.

Step 3 In the dialog box displayed, confirm the deletion. Deleted scheduling policies cannot be restored.

NOTE

Do not close this dialog box or refresh the page during deletion, which may cause residual resources. The dialog box is auto closed upon successful deletion.

----End

9 Observability

9.1 Container Intelligent Analysis

9.1.1 Overview

Container Intelligent Analysis (CIA) is a next-generation O&M platform for cloud native containers. It monitors applications and resources in real time, collects metrics and events to analyze application health, and visualizes multi-dimensional data. Compatible with mainstream open source components, it supports quick fault locating.

Functions

- **Container Insights** comprehensively monitors Kubernetes native containers, provides the resource overview of clusters, nodes, and workloads, and displays node resource usage, workload resource consumption, and CPU/memory metrics in the past hour for the health and load of clusters.
- **Health Diagnosis** periodically checks the health statuses of clusters, including the resource usage of clusters and nodes as well as running statuses of workloads and pods.
- **Dashboard** displays different graphs such as line graphs and digit graphs on the same screen, which lets you view comprehensive monitoring data.

Advantages

- CIA is deeply integrated with Prometheus, a mature monitoring project of the Cloud Native Computing Foundation (CNCF), and complies with the OpenTracing and OpenTelemetry specifications. It brings in observability for your cloud native applications by collecting, storing, and visually presenting O&M data, such as key metrics and events.
- It provides full-stack monitoring from cloud native infrastructure resources to applications, enabling users to clearly perceive the infrastructure and application load status anytime and anywhere.

- It monitors Kubernetes clusters and container pods, provides end-to-end tracing and visualization for services, and provides cluster health diagnosis capabilities, greatly shortening the fault analysis and locating time.
- It provides ready-to-use add-ons, data collection, and dashboard-based monitoring. Compared with monitoring products developed based on open source technologies, it is more competitive in reliability, availability, and deployment.

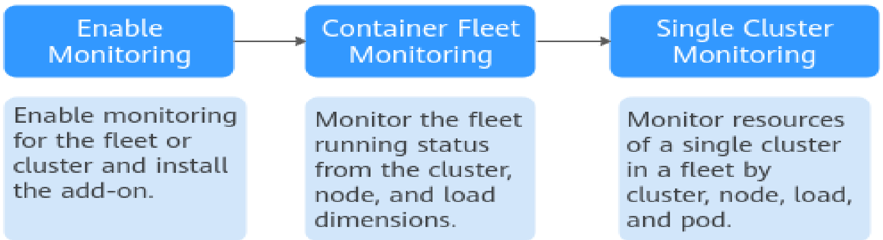
Constraints

- Only **Huawei Cloud accounts** or users with the **UCS FullAccess** or **UCS CIAOperations** (recommended) permission can perform container analysis operations.
- Currently, metrics and events of on-premises clusters and attached clusters can be reported to AOM 2.0 and LTS only in CN North-Beijing4. CIA can be enabled for Huawei Cloud clusters only in CN North-Beijing4 and CN East-Shanghai1.

Procedure

The following figure shows the process for using CIA.

Figure 9-1 Process for using CIA



9.1.2 Enabling Cluster Monitoring

9.1.2.1 Overview

You can enable monitoring for a cluster to ensure that the cluster is in the real-time protection state.

For details, see **kube-prometheus-stack** and **Cloud Native Logging Add-on**.

Currently, CIA can monitor Huawei Cloud clusters, attached clusters, on-premises clusters, multi-cloud clusters . When monitoring is enabled, the parameter settings of each cluster are different. This section describes how to enable monitoring for the five types of clusters.

- **Enabling Monitoring for Huawei Cloud Clusters**
- **Enabling Monitoring for On-premises Clusters**
- **Enabling Monitoring for Attached Clusters**
- **Enabling Monitoring for Multi-Cloud Clusters**

Add-on Status Description

Table 9-1 describes the status of kube-prometheus-stack and log-agent. Some statuses affect cluster monitoring enabling, monitoring configuration modification, and monitoring disabling. For details, see constraints in subsequent sections.

Table 9-1 Add-on status description

Add-on Status	Description
Not installed	The add-on is not installed.
Running	All add-on instances are in the running status and the add-on is working.
Installing	The add-on is being installed.
Upgrading	The add-on is being upgraded.
Rolling back	The add-on is being rolled back.
Rollback failed	The add-on rollback failed. You can retry the rollback, or uninstall it and try again.
Deleting	The add-on is being deleted.
Partially ready	Only some instances are in the running status, and the add-on is partially available.
Not available	Add-on abnormal and unavailable. Click the add-on name to view exceptions.
Installation failed	Install add-on failed. Uninstall it and try again.
Upgrade failed	Upgrade add-on failed. Upgrade it again or uninstall it and try again.
Deletion failed	Delete add-on failed. Try again.
Unknown	The add-on is in the unknown state. Reinstall it and try again.

9.1.2.2 Enabling Monitoring for Huawei Cloud Clusters

This section describes how to enable monitoring for Huawei Cloud clusters.

Constraints

Before enabling monitoring for Huawei Cloud clusters, kube-prometheus-stack may have been installed. If the add-on is in the **Installing**, **Upgrading**, **Deleting**, or **Rolling back** status, monitoring cannot be enabled. For details about the add-on status, see **Add-on Status Description**.

Prerequisites

A Huawei Cloud cluster has been registered with UCS. For details, see **Huawei Cloud Clusters**.

Procedure

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Container Intelligent Analysis**.

Step 2 Select a fleet or a cluster not in the fleet, and click **Enable Monitoring**.

Figure 9-2 Selecting a fleet or a cluster not in the fleet

Step 3 Select a Huawei Cloud cluster.

Step 4 Click **Next: Configure Connection** to complete the metric collection settings.

Specifications

- **Deployment Mode**: The **Agent** and **Server** modes are supported. The **Agent** mode occupies fewer cluster resources and provides the Prometheus metric collection capability for the cluster. However, the HPA and health diagnosis functions based on custom Prometheus statements are not supported. The **Server** mode provides the Prometheus metric collection capability for clusters and supports HPA and health diagnosis based on custom Prometheus statements. This mode depends on PVC and consumes a large amount of memory.
- **Add-on Specifications**: If **Deployment Mode** is set to **Agent**, the default add-on specifications are used. If **Deployment Mode** is set to **Server**, the add-on specifications include **Demo (≤ 100 containers)**, **Small (≤ 2,000 containers)**, **Medium (≤ 5,000 containers)**, and **Large (> 5,000 containers)**. Different specifications have different requirements on cluster resources, such as CPUs and memory. For details about the resource quotas of different add-on specifications, see **Resource Quota Requirements of Different Specifications**..

Parameters

- **Interconnection Mode**: Currently, only AOM can be interconnected.
- **AOM Instance**: Container monitoring reports metrics to AOM in a unified manner. Therefore, you need to select an AOM Prometheus for CCE instance. The default metrics are collected for free but custom metrics are billed by AOM.

- **Collection Period**: period for Prometheus to collect and report metrics. The value ranges from 10 to 120 seconds. The default value is 15 seconds.

  Storage: (Required when **Deployment Mode** is set to **Server**) Used for temporary storage (PVC) of Prometheus data. By default, Huawei Cloud clusters use PVCs of the csi-disk-topology storage type. If an available PVC (pvc-prometheus-server) exists in the namespace **monitoring**, it can be used as the storage source.
  - **EVS Disk Type**: You can select **High I/O**, **Ultra-high I/O**, or **Common I/O**.
  - **Capacity**: capacity specified when a PVC is created or the maximum storage limit when the pod storage is selected.

  NOTICE

  Using EVS disks for add-on storage will incur extra fees. For details, see **Product Pricing Details**.

For details about the add-on, see **kube-prometheus-stack**.

Step 5 Click **Confirm**. The **Container Insights** > **Clusters** page is displayed. The access status of the cluster is **Installing**.

After monitoring is enabled for the cluster, metrics such as the CPU usage and CPU allocation rate of the cluster are displayed in the list, indicating that the cluster is monitored by CIA.

NOTE

If monitoring fails to be enabled for the cluster, rectify the fault by referring to **FAQs**.

----End

9.1.2.3 Enabling Monitoring for On-premises Clusters

This section describes how to enable monitoring for on-premises clusters.

Prerequisites

An on-premises cluster has been registered with UCS. For details, see **Overview**.

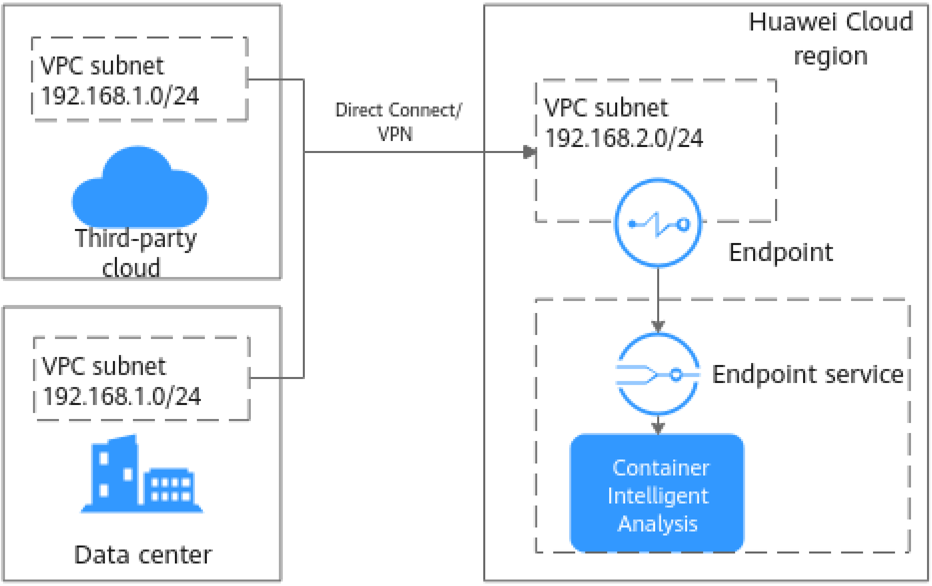
Preparing the Network Environment

There are two options, public network and private network, for data access of an on-premises cluster.

- The public network features flexibility, cost-effectiveness, and easy access. If network quality is not a concern and simpler access is preferred, public network access is a good choice.

  This option is only available for clusters that can access the public network.
- The private network features high speed, low latency, and security. After you connect the on-premises network to the cloud network over Direct Connect or VPN, you can use a VPC endpoint to access CIA over the private network.

Figure 9-3 Private network access diagram



Before enabling this function, you need to prepare a VPC and connect the network environment of the on-premises data center to the VPC. The VPC subnet CIDR block cannot overlap with the CIDR block used by the on-premises data center. Otherwise, the cluster cannot be connected. For example, if the VPC subnet used by the on-premises data center is 192.168.1.0/24, the subnet 192.168.1.0/24 cannot be used in the Huawei Cloud VPC.

Use either of the following methods to connect the network:

– VPN: See **Connecting an On-Premises Data Center to a VPC Through a VPN**.

– Direct Connect: See **Accessing a VPC over a Single Connection Through Static Routes** or **Accessing a VPC over a Single Connection Through BGP Routes**.

Enabling Monitoring

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Container Intelligent Analysis**.

Step 2 Select a fleet or a cluster not in the fleet, and click **Enable Monitoring**.

Figure 9-4 Selecting a fleet or a cluster not in the fleet

Step 3 Select an on-premises cluster.

Step 4 Click **Next: Configure Connection** to complete the network settings.

- **Data Access**: Select **Public access** or **Private access**.
- **Data Reported To**: Select the region where data is reported. The region must be the same as that of the VPC connected to the on-premises cloud network.
- **Project**: If the IAM project function is enabled, you also need to select a project.
- **Private access**: This parameter is mandatory when **Data Access** is set to **Private access**.

  To connect to the data reporting and receiving point of CIA, you can create a VPC endpoint in the VPC that has been connected to the on-premises network. You can select an existing private network access point. If you create a private network access point, you will be billed 0.1 CNY/hour for VPCEP resources.

  When you create a private network access point, a VPC endpoint and a DNS private domain name will be generated. Ensure that the Huawei Cloud account has corresponding resource quotas. In addition, ensure that the subnet selected on the page has available IP addresses.

Step 5 Complete metric collection settings.

Specifications

- **Deployment Mode**: The **Agent** and **Server** modes are supported. The **Agent** mode occupies fewer cluster resources and provides the Prometheus metric collection capability for the cluster. However, the HPA and health diagnosis functions based on custom Prometheus statements are not supported. The **Server** mode provides the Prometheus metric collection capability for clusters and supports HPA and health diagnosis based on custom Prometheus statements. This mode depends on PVC and consumes a large amount of memory.
- **Add-on Specifications**: If **Deployment Mode** is set to **Agent**, the default add-on specifications are used. If **Deployment Mode** is set to **Server**, the add-on specifications include **Demo (≤ 100 containers)**, **Small (≤ 2,000 containers)**, **Medium (≤ 5,000 containers)**, and **Large (> 5,000 containers)**. Different specifications have different requirements on cluster resources, such as CPUs and memory. For details about the resource quotas of different add-on specifications, see **Resource Quota Requirements of Different Specifications**..

Parameters

- **Interconnection Mode**: Currently, only AOM can be interconnected.
- **AOM Instance**: Container monitoring reports metrics to AOM in a unified manner. Therefore, you need to select an AOM Prometheus for CCE instance. The default metrics are collected for free but custom metrics are billed by AOM. For details, see **AOM Billing**.
- **Collection Period**: period for Prometheus to collect and report metrics. The value ranges from 10 to 60 seconds. The default value is 15 seconds.
- **Storage**: used to temporarily store Prometheus data. This parameter is mandatory when **Deployment Mode** is set to **Server**. The on-premises cluster

supports the CSI-Local storage type. A local volume represents a local disk of a node that is provided to a pod through a PVC. With local volumes, a pod using a local volume is always scheduled to the same node. Ensure that the scheduling policy of the pod does not conflict with that of the target node.

- **Storage Type**: Select **CSI-Local**.
- **Capacity**: capacity specified when the PVC is created. This capacity is for reference only. The actual capacity is the available capacity of the disk where the local directory is located.
- **Node**: node to which Prometheus will be scheduled. Ensure that Prometheus can be scheduled to this node.
- **Node Path**: directory for storing data on Prometheus. Enter an absolute path. The path will be automatically created on the target node.

For details about the add-on, see **kube-prometheus-stack**.

Step 6 Click **Confirm**. The **Container Insights** > **Clusters** page is displayed. The access status of the cluster is **Installing**.

After monitoring is enabled for the cluster, metrics such as the CPU usage and CPU allocation rate of the cluster are displayed in the list, indicating that the cluster is monitored by CIA.

NOTE

If monitoring fails to be enabled for the cluster, rectify the fault by referring to **FAQs**.

----End

9.1.2.4 Enabling Monitoring for Attached Clusters

This section describes how to enable monitoring for attached clusters.

Prerequisites

An attached cluster has been registered with UCS. For details, see **Overview**.

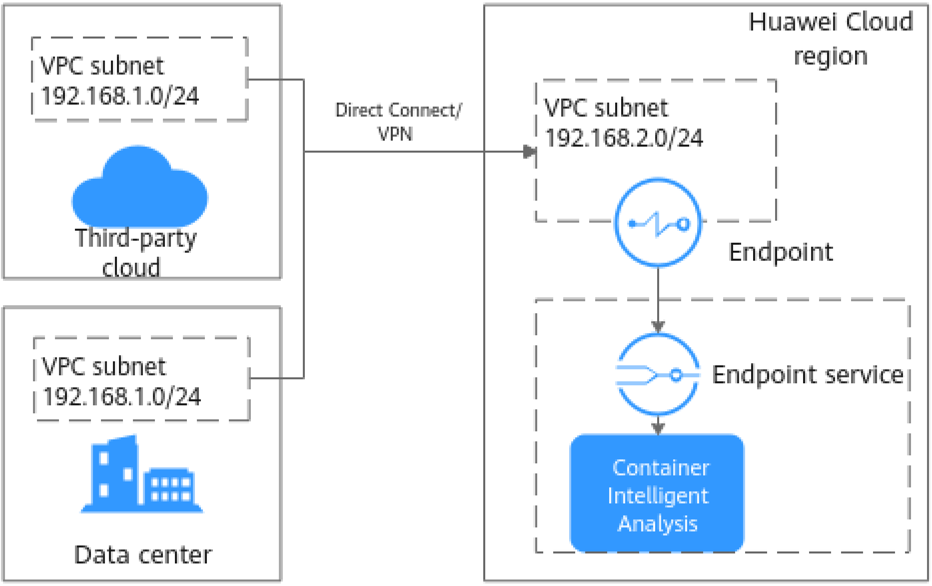
Preparing the Network Environment

There are two options, public network and private network, for data access of an attached cluster.

- The public network features flexibility, cost-effectiveness, and easy access. If network quality is not a concern and simpler access is preferred, public network access is a good choice.

  This option is only available for clusters that can access the public network.
- The private network features high speed, low latency, and security. After you connect the private network of a third-party cloud to the cloud network over Direct Connect or VPN, you can use a VPC endpoint to access CIA over the private network.

Figure 9-5 Private network access diagram



Before enabling this function, you need to prepare a VPC and connect the network environment of the third-party cloud vendor to the VPC. The VPC subnet CIDR block cannot overlap with the network CIDR block used by the third-party cloud. Otherwise, the cluster cannot be connected. For example, if the VPC subnet used by the third-party cloud is 192.168.1.0/24, the subnet 192.168.1.0/24 cannot be used in the Huawei Cloud VPC.

Use either of the following methods to connect the network:

- VPN: See **Connecting an On-Premises Data Center to a VPC Through a VPN**.
- Direct Connect: See **Accessing a VPC over a Single Connection Through Static Routes** or **Accessing a VPC over a Single Connection Through BGP Routes**.

Enabling Monitoring

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Container Intelligent Analysis**.

Step 2 Select a fleet or a cluster not in the fleet, and click **Enable Monitoring**.

Figure 9-6 Selecting a fleet or a cluster not in the fleet

Step 3 Select an attached cluster.

Step 4 Click **Next: Configure Connection** to complete the network settings.

- **Data Access**: Select **Public access** or **Private access**.
- **Data Reported To**: Select the region where data is reported. The region must be the same as that of the VPC connected to the third-party cloud network.
- **Project**: If the IAM project function is enabled, you also need to select a project.
- **Private access**: This parameter is mandatory when **Data Access** is set to **Private access**.

  To connect to the data reporting and receiving point of CIA, you can create a VPC endpoint in the VPC that has been connected to the private network of the third-party cloud. You can select an existing private network access point. If you create a private network access point, you will be billed 0.1 CNY/hour for VPCEP resources.

  When you create a private network access point, a VPC endpoint and a DNS private domain name will be generated. Ensure that the Huawei Cloud account has corresponding resource quotas. In addition, ensure that the subnet selected on the page has available IP addresses.

Step 5 Complete metric collection settings.

Specifications

- **Deployment Mode**: The **Agent** and **Server** modes are supported. The **Agent** mode occupies fewer cluster resources and provides the Prometheus metric collection capability for the cluster. However, the HPA and health diagnosis functions based on custom Prometheus statements are not supported. The **Server** mode provides the Prometheus metric collection capability for clusters and supports HPA and health diagnosis based on custom Prometheus statements. This mode depends on PVC and consumes a large amount of memory.
- **Add-on Specifications**: If **Deployment Mode** is set to **Agent**, the default add-on specifications are used. If **Deployment Mode** is set to **Server**, the add-on specifications include **Demo (≤ 100 containers)**, **Small (≤ 2,000 containers)**, **Medium (≤ 5,000 containers)**, and **Large (> 5,000 containers)**. Different specifications have different requirements on cluster resources, such as CPUs and memory. For details about the resource quotas of different add-on specifications, see **Resource Quota Requirements of Different Specifications**..

Parameters

- **Interconnection Mode**: Currently, only AOM can be interconnected.
- **AOM Instance**: Container monitoring reports metrics to AOM in a unified manner. Therefore, you need to select an AOM Prometheus for CCE instance. The default metrics are collected for free but custom metrics are billed by AOM. For details, see **AOM Billing**.
- **Collection Period**: period for Prometheus to collect and report metrics. The value ranges from 10 to 60 seconds. The default value is 15 seconds.
- **Storage**: used to temporarily store Prometheus data. This parameter is mandatory when **Deployment Mode** is set to **Server**.

– **Storage Type**: Attached clusters support **emptyDir** and **local-storage**.

  If **emptyDir** is used, Prometheus data will be stored in the pod. Ensure that the storage volume mounted to the container on the node scheduled by prometheus-server-0 is no less than the entered capacity.

  If **local-storage** is used, the monitoring namespace (if it does not exist) and PVs and PVCs of the local-storage type will be created in your cluster. Ensure that the entered directory exists on the specified node and the path capacity is sufficient.

– **Capacity**: capacity specified when a PVC is created or the maximum storage limit when the pod storage is selected.

For details about the add-on, see **kube-prometheus-stack**.

Step 6 Click **Confirm**. The **Container Insights** > **Clusters** page is displayed. The access status of the cluster is **Installing**.

After monitoring is enabled for the cluster, metrics such as the CPU usage and CPU allocation rate of the cluster are displayed in the list, indicating that the cluster is monitored by CIA.

NOTE

If monitoring fails to be enabled for the cluster, rectify the fault by referring to **FAQs**.

----End

9.1.2.5 Enabling Monitoring for Multi-Cloud Clusters

This section describes how to enable monitoring for multi-cloud clusters.

Prerequisites

A multi-cloud cluster has been registered with UCS. For details, see **Overview**.

Preparing the Network Environment

The data access mode of a multi-cloud cluster supports public network access, which is flexible, inexpensive, and easy. The cluster must be able to access public networks. If network quality is not a concern and simpler access is preferred, public network access is a good choice.

This option is only available for clusters that can access public networks.

Enabling Monitoring

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Container Intelligent Analysis**.

Step 2 Select a fleet or a cluster not in the fleet, and click **Enable Monitoring**.

Figure 9-7 Selecting a fleet or a cluster not in the fleet

Step 3 Select a multi-cloud cluster.

Step 4 Click **Next: Configure Connection** to complete the network settings.

- **Data Access**: Select **Public access**.
- **Data Reported To**: Select the region where data is reported.
- **Project**: If the IAM project function is enabled, you also need to select a project.

Step 5 Complete metric collection settings.

Specifications

- **Deployment Mode**: The **Agent** and **Server** modes are supported. The **Agent** mode occupies fewer cluster resources and provides the Prometheus metric collection capability for the cluster. However, the HPA and health diagnosis functions based on custom Prometheus statements are not supported. The **Server** mode provides the Prometheus metric collection capability for clusters and supports HPA and health diagnosis based on custom Prometheus statements. This mode depends on PVC and consumes a large amount of memory.
- **Add-on Specifications**: If **Deployment Mode** is set to **Agent**, the default add-on specifications are used. If **Deployment Mode** is set to **Server**, the add-on specifications include **Demo (≤ 100 containers)**, **Small (≤ 2,000 containers)**, **Medium (≤ 5,000 containers)**, and **Large (> 5,000 containers)**. Different specifications have different requirements on cluster resources, such as CPUs and memory. For details about the resource quotas of different add-on specifications, see **Resource Quota Requirements of Different Specifications**..

Parameters

- **Interconnection Mode**: Currently, only AOM can be interconnected.
- **AOM Instance**: Container monitoring reports metrics to AOM in a unified manner. Therefore, you need to select an AOM Prometheus for CCE instance. The default metrics are collected for free but custom metrics are billed by AOM. For details, see **AOM Billing**.
- **Collection Period**: period for Prometheus to collect and report metrics. The value ranges from 10 to 60 seconds. The default value is 15 seconds.
- **Storage**: used to temporarily store Prometheus data. This parameter is mandatory when **Deployment Mode** is set to **Server**.
  – **Storage Type**: Multi-cloud clusters support **emptyDir** and **local-storage**.

If **emptyDir** is used, Prometheus data will be stored in the pod. Ensure that the storage volume mounted to the container on the node scheduled by prometheus-server-0 is no less than the entered capacity.

If **local-storage** is used, the monitoring namespace (if it does not exist) and PVs and PVCs of the local-storage type will be created in your cluster. Ensure that the entered directory exists on the specified node and the path capacity is sufficient.

– **Capacity**: capacity specified when a PVC is created or the maximum storage limit when the pod storage is selected.

For details about the add-on, see **kube-prometheus-stack**.

Step 6 Click **Confirm**. The **Container Insights** > **Clusters** page is displayed. The access status of the cluster is **Installing**.

After monitoring is enabled for the cluster, metrics such as the CPU usage and CPU allocation rate of the cluster are displayed in the list, indicating that the cluster is monitored by CIA.

NOTE

If monitoring fails to be enabled for the cluster, rectify the fault by referring to **FAQs**.

----End

9.1.2.6 Modifying Monitoring Settings

After monitoring is enabled for a cluster, you can modify monitoring settings, including the network settings, metric collection settings, and event collection settings.

NOTE

When **Event Collection Settings** is toggled off, the system deletes the **log-agent** add-on.

Constraints

If the **kube-prometheus-stack** add-on is in **Installing**, **Upgrading**, **Deleting**, **Rolling back**, **Rollback failed**, **Not available**, **Installation failed**, **Deletion failed**, or **Unknown** state, the cluster monitoring configurations cannot be modified.

Procedure

Step 1 Log in to the UCS console. In the navigation pane on the left, choose **Container Intelligent Analysis**.

Step 2 Select a fleet or a cluster that is not added to the fleet.

Figure 9-8 Selecting a fleet or a cluster not in the fleet

Step 3 Choose **Container Insights** > **Clusters** to view the clusters with monitoring enabled. Locate the cluster for which you want to modify the configurations and click **Modify Access Configuration** in the **Operation** column.

Step 4 Click **Confirm**.

----End

9.1.2.7 Disabling Monitoring

This section describes how to disable cluster monitoring.

Constraints

Before disabling monitoring, read the following precautions carefully to prevent data loss or additional fees.

- Monitoring cannot be disabled when the **kube-prometheus-stack** add-on is in **Installing**, **Upgrading**, **Deleting**, or **Rolling back** state.
- Disable monitoring when the **kube-prometheus-stack** add-on is in **Running**, **Partially ready**, or **Installation failed** state. For Huawei Cloud clusters, the system updates the **kube-prometheus-stack** add-on to disable the data reporting function. For on-premises and attached clusters, the system uninstalls the **kube-prometheus-stack** add-on.
- If the **kube-prometheus-stack** add-on is in **Rollback failed**, **Not available**, **Installation failed**, **Deletion failed**, or **Unknown** state, disabling monitoring will uninstall the **kube-prometheus-stack** add-on.
- For on-premises and attached clusters accessed through private networks, when monitoring is disabled, the system checks whether the private network access point (VPCEP and DNS private domain name created when monitoring is enabled) is being used by other clusters. If not, the private network access point will be deleted.
- Huawei Cloud clusters use PVCs of the **csi-disk-topology** storage type to temporarily store add-on data. After cluster monitoring is disabled, PVCs in the **monitoring** namespace cannot be automatically deleted. To avoid unexpected expenditures, go to the CCE console and manually delete the PVCs. (You need to uninstall the **kube-prometheus-stack** add-on first.)

Procedure

Step 1 Select a fleet or a cluster that is not added to the fleet.

Figure 9-9 Selecting a fleet or a cluster not in the fleet

Step 2 Choose **Container Insights** > **Clusters** to view the clusters with monitoring enabled. Locate the cluster for which you want to disable monitoring and click **Cancel Monitoring** in the **Operation** column.

Step 3 In the confirmation dialog box, click **OK** to disable monitoring for the cluster.

----End

9.1.3 Container Insights

9.1.3.1 Overview

Container Insights comprehensively monitors Kubernetes native containers, provides the resource overview of clusters, nodes, and workloads, and displays node resource usage, workload resource consumption, and CPU/memory metrics in the past hour for the health and load of clusters.

9.1.3.2 Viewing Fleet Information

You can select a fleet to view the clusters with monitoring enabled, as well as nodes and workloads in these clusters.

NOTE

To view the clusters not in the fleet and their nodes and workloads, choose **Others** > **Clusters Not in Fleet** on the **Container Insights** tab.

Viewing Cluster Information in a Fleet

Navigation Path

Step 1 Log in to the UCS console. In the navigation pane, choose **Container Intelligent Analysis**. Then select a fleet.

Step 2 Choose **Container Insights** > **Clusters** to view the clusters with monitoring enabled. The list displays metrics such as the CPU usage, CPU allocation rate, memory usage, and memory allocation rate.

----End

Tab Overview

On the **Clusters** tab, you can view information about all clusters in a fleet, such as the status, type, region, CPU usage, CPU allocation rate, memory usage, memory allocation rate, and the numbers of normal and total nodes of each cluster. You can also enable or disable monitoring for a cluster and **modify cluster monitoring settings**.

Module	Description
Cluster Statistics	This module displays information about all clusters in a fleet, such as the cluster name, risk level, status, type, region, CPU usage, CPU allocation rate, memory usage, and memory allocation rate. You can click the search box above the list, select a property type, and enter a keyword to search for the desired cluster.
Cluster Risk Overview	This module displays 24/7 health inspection results of clusters so you can quickly diagnose cluster risks and Kubernetes warning events and address abnormal items following the provided suggestions. **NOTE** Only the latest 100 Kubernetes warning events are displayed. To view more events, go to the **Events** tab of the cluster. The Kubernetes events of attached clusters are not included.
Usage Statistics	By default, the average CPU threshold and average memory threshold in the last 1 hour, last 8 hours, and last 24 hours are displayed for you to quickly identify resource usages. **NOTE** You can hover over a chart to view the monitoring data in each minute.
Resource Health Overview	This module displays top 5 clusters by CPU usage, memory usage, node quantity, and pod quantity. You can click **Allocatable** to view the allocatable memory or CPU and click **Abnormal** to view the number of abnormal clusters.
Resource Stocktaking	This module displays the proportions of clusters in the fleet by cluster version, carrier, type, and region. You can click the cluster version, carrier name, or on-premises cluster to view the proportions of other types of clusters in the fleet.

Viewing Node Information in a Fleet

Navigation Path

Step 1 Log in to the UCS console. In the navigation pane, choose **Container Intelligent Analysis**. Then select a fleet.

Step 2 Choose **Container Insights** > **Nodes**.

----End

Tab Overview

On the **Nodes** tab, you can view information about nodes in all clusters with monitoring enabled, as well as node risk statistics and resource usages.

Table 9-2 Modules on the Nodes tab

Module	Description
Node Overview	This module displays the name, status, CPU usage, memory usage, cluster, node IP address, and region of each node. You can click the search box above the list, select a property type, and enter a keyword to search for the desired node.
Node Risk Overview	This module displays Kubernetes warning events that occur on nodes in the clusters with monitoring enabled. For each event, you can view the event name and type, cluster name, resource type, resource name, event content, occurrence time, and number of occurrences. You can click the search box above the list, select a property type, and enter a keyword to search for the desired event. You can also click to sort events. **NOTE** Only the latest 100 Kubernetes warning events are displayed. To view more events, go to the **Events** tab of the cluster.
Resource Health Overview	This module displays top 5 nodes by CPU usage and memory usage. You can click **Allocatable** to view the allocatable memory or CPU. You can click any node next to the chart of **Top 5 Nodes by CPU Usage** to hide its data in the chart and view only the data of other nodes.

Viewing Workload Information in a Fleet

Navigation Path

The **Workloads** tab displays information about all workloads in the clusters with monitoring enabled. On this tab, you can view the workload list, risk overview, and resource health overview.

Step 1 Log in to the UCS console. In the navigation pane, choose **Container Intelligent Analysis**. Then select a fleet.

Step 2 Choose **Container Insights** > **Workloads** to view all workloads in the clusters with monitoring enabled. The workload list displays metrics such as the workload name, status, number of pods, CPU usage, and memory usage. In the upper right corner of the list, you can filter desired workloads by workload type.

----End

Module	Description
Workload Overview	This module displays information about all workloads in the clusters with monitoring enabled, such as their status, the numbers of normal and total pods, namespace, cluster, CPU usage, and memory usage. You can click the search box above the list, select a property type, and enter a keyword to search for the desired workload. You can also click to sort workloads.

Module	Description
Risk Overview	This module displays Kubernetes warning events that occur on nodes in the clusters with monitoring enabled. For each event, you can view the event name and type, cluster name, resource type, resource name, event content, occurrence time, and number of occurrences. You can click the search box above the list, select a property type, and enter a keyword to search for the desired event. You can also click to sort events. You can select search criteria in the upper right corner to filter workloads in the list. **NOTE** Only the latest 100 Kubernetes warning events are displayed. To view more events, go to the **Events** tab of the cluster.
Resource Health Overview	This module displays top 5 workloads by CPU usage, memory usage, number of restarts, and number of abnormal pods in the fleet. You can click **Average Usage** to view the allocatable memory or CPU. You can click any workload next to the chart of **Top 5 Workloads by CPU Usage** to hide its data in the chart and view only the data of other workloads.

9.1.3.3 Viewing Cluster Information

Navigation Path

Choose **Container Insights** > **Clusters** and click the cluster name in **Cluster Statistics**. The displayed page consists of the following tabs:

- **Clusters**: For details, see **Viewing Cluster Details**.
- **Nodes**: For details, see **Viewing Node Details**.
- **Workloads**: For details, see **Viewing Workload Details**.
- **Pods**: For details, see **Viewing Pod Details**.
- **Events**: For details, see **Viewing Event Details**.

Viewing Cluster Details

The cluster details page provides monitoring data of a single cluster, including the resource overview, top resource consumption statistics, and usage statistics. Cluster monitoring allows you to view the resource usage and trend of a cluster in a timely manner and quickly handle potential risks for smooth cluster running.

You can hover over a chart to view the monitoring data in each minute.

Figure 9-10 Cluster details page

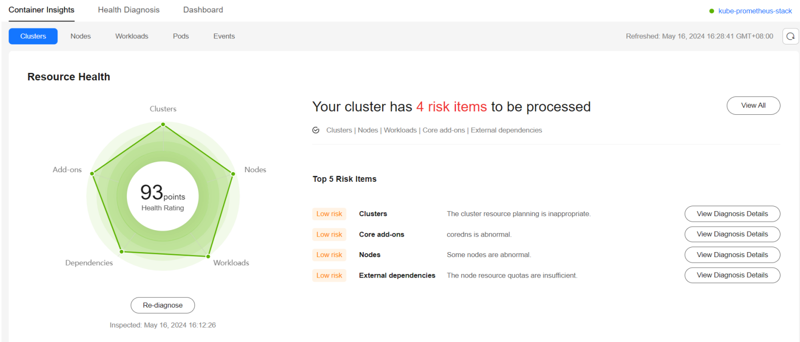


Table 9-3 Modules on the cluster details page

Module	Description
Resource Health	Resource health is evaluated from several dimensions, such as the health score, number of risk items to be processed, risk level, and proportion of diagnosed risk items for master nodes, clusters, worker nodes, workloads, and external dependencies. (Abnormal data is displayed in red.) For more diagnosis results, go to the **Health Diagnosis** tab. **NOTICE** You can view the resource health status of a cluster only when kube-prometheus-stack is deployed in server mode in the cluster.
Resource Overview	This module displays the proportion of abnormal resources in nodes, workloads, and pods and the total number of namespaces. In addition, the exception proportion of control plane components and master nodes, total QPS of the API server, and request error rate of the API server are also included. As the API service provider of the cluster, if the API server on the control plane is abnormal, the entire cluster may fail to be accessed and workloads that depend on the API server may fail to run normally. To help you quickly identify and fix problems, this module provides the total QPS and request error rate metrics of the API server.
Top Resource Consumption Statistics	This module displays statistics collected by UCS on top 5 nodes, Deployments, StatefulSets, and pods by CPU and memory usage, helping you identify high resource consumption. **NOTE** • CPU usage Workload CPU usage = Average CPU usage in each pod of the workload Pod CPU usage = Used CPU cores/Sum of CPU limits of containers in the pod (If CPU limits are not specified, all node CPU cores are used.) • Memory usage Workload memory usage = Average memory usage in each pod of the workload Pod memory usage = Used physical memory/Sum of memory limits of containers in the pod (If memory limits are not specified, all node memory is used.)

Module	Description
Data Plane Monitoring	By default, the resource usage is collected from each dimension in the last hour, last 8 hours, and last 24 hours. To view more monitoring information, click **View All Metrics** to access the **Dashboard** tab. For details, see **Dashboard**.

9.1.3.4 Viewing Node Information

To monitor the resource usage of nodes, go to the **Nodes** tab. This tab provides information about all nodes in a cluster and monitoring data of a node, such as the CPU usage, memory usage, network inbound rate, network outbound rate, disk read rate, and disk write rate.

Navigation Path

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Container Intelligent Analysis**. Choose **Container Insights** > **Clusters**, click the cluster name in **Cluster Statistics**, and click the **Nodes** tab.

This tab displays information about all workloads. To view the monitoring data of a workload, click the workload name to access its **Overview** tab and switch to the **Pods** or **Monitoring** tab.

----End

Viewing the Node List

The node list displays the name, status, IP address, number of pods (allocated/total), CPU request/limit/usage, and memory request/limit/usage of each node.

You can search for the desired node by name, status, private IP address, or public IP address. You can click  in the upper right corner of the list to export data of all nodes or selected nodes. The exported file is in .xlsx format, and the file name contains the timestamp.

Node Overview displays the name, status, CPU usage, memory usage, cluster, IP address, and region of each node. You can click the search box above the list, select a property type, and enter a keyword to search for the desired node.

If the CPU limit or memory limit of a node exceeds 100%, the node resources are overcommitted and the sum of workload limits (maximum available values) of the node exceeds the node specifications. If a workload occupies too many resources, the node may be abnormal.

Viewing Node Details

In the node list, click the node name to to access its **Overview** page and switch to the **Pods** or **Monitoring** tab.

Table 9-4 Modules on the node details page

Module	Description
Overview	You can click the node name to access this tab. On this tab, you can view: ● **Resource Overview**: displays the node status and number of pods as well as abnormal events. ● **Node Monitoring**: displays the monitoring data in the last hour, last 8 hours, last 24 hours, and custom period, including the CPU usage, memory usage, and network inbound/outbound rate. ● **Pod Usage Trend**: displays top 5 pods by used CPU and memory in the last hour, last 8 hours, last 24 hours, and custom period.
Pods	This tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, CPU usage, and memory usage of each pod. You can search for the desired pod by name, status, namespace, IP address, or node. You can click  in the upper right corner of the list to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp. You can click the name of a pod to view its detailed monitoring data. For more information, see **Viewing Pod Information**.
Monitoring	This tab displays the resource usage of the node in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access the **Dashboard** tab. For details, see **Dashboard**.

9.1.3.5 Viewing Workload Information

To monitor the resource usage of workloads, go to the **Workloads** tab. This tab provides information about all workloads in a cluster and monitoring data of a workload, such as the CPU usage, memory usage, network inbound rate, network outbound rate, and disk usage.

Navigation Path

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Container Intelligent Analysis**. Choose **Container Insights** > **Clusters**, click the cluster name in **Cluster Statistics**, and click the **Workloads** tab.

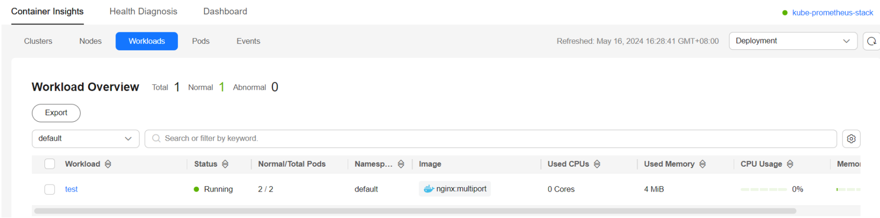
This tab displays information about all workloads. To view the monitoring data of a workload, click the workload name to access its **Overview** tab and switch to the **Pods** or **Monitoring** tab.

----End

Viewing the Workload List

The workload list displays the name, status, number of pods (normal/all), namespace, image name, CPU usage, and memory usage of each workload.

Figure 9-11 Workload list page



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

You can click  in the upper right corner of the list to export data of all workloads or selected workloads. The exported file is in .xlsx format, and the file name contains the timestamp.

Viewing Workload Details

In the workload list, click the workload name to to access its **Overview** page and switch to the **Pods** or **Monitoring** tab.

Table 9-5 Modules on the workload details page

Module	Description
Overview	You can click the workload name to access this tab. This tab consists of the following: ● **Resource Overview**: displays the workload status and the numbers of abnormal and total pods, as well as abnormal events. ● **Workload Monitoring**: displays the CPU usage, memory usage, network inbound rate, and network outbound rate. ● **Pod Usage Trend**: You can switch the metrics in the upper left corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each pod of the workload. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper right corner to view the top 5 data in descending or ascending order.

Module	Description
Pods	This tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, CPU usage, and memory usage of each pod. You can search for the desired pod by name, status, namespace, IP address, or node. You can click  in the upper right corner of the list to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp. You can click the name of a pod to view its detailed monitoring data. For more information, see **Viewing Pod Information**.
Monitoring	This tab displays the resource usage of the workload in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access the **Dashboard** tab. For details, see **Dashboard**.

9.1.3.6 Viewing Pod Information

To monitor the resource usage of pods, go to the **Pods** tab. This tab provides information about all pods in a cluster and monitoring data of a pod, such as the CPU usage, memory usage, network inbound rate, network outbound rate, and disk usage.

NOTE

Container groups, pods and instances are the same concept.

Navigation Path

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Container Intelligent Analysis**. On the **Container Insights** > **Clusters** tab, click the target cluster name in **Cluster Statistics** and click the **Pods** tab.

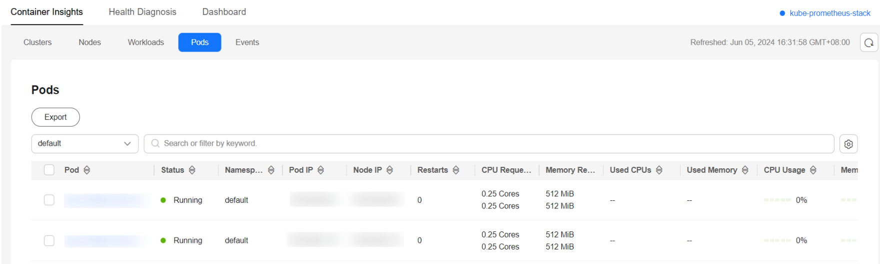
This tab displays information about all pods. To view the monitoring data of a pod, click the pod name to access its **Overview** page and switch to the **Containers** or **Monitoring** tab.

----End

Viewing Pods

The **Pods** tab lists the name, status, namespace, IP address, node, number of restarts, CPU request/limit, memory request/limit, CPU usage, and memory usage of each pod.

Figure 9-12 Pods



You can select a namespace and the name, status, IP address,or node above the list to quickly search for the desired pod.

You can click  in the upper right corner of the list to export data of all pods or selected pods. The exported file is in .xlsx format, and the file name contains the timestamp.

Viewing Pod Details

In the pod list, click the pod name to to access its **Overview** page and switch to the **Containers** or **Monitoring** tab.

Table 9-6 Modules on the pod details page

Module	Description
Overview	You can click the pod name to access this tab. This tab consists of the following: ● **Resource Overview**: displays the pod status and the numbers of abnormal and total containers, as well as abnormal events. ● **Container Monitoring**: displays the CPU usage, memory usage, network inbound rate, and network outbound rate. ● **Container Usage Trend**: You can switch the metrics in the upper left corner of the chart to view the CPU usage, used CPUs, memory usage, and used memory of each container in the pod. You can also click **Top 5 (Descending)** or **Top 5 (Ascending)** in the upper right corner to view the top 5 data in descending or ascending order.
Containers	This tab contains details such as the name, status, namespace, number of restarts, and image of each container. You can search for the desired container by name, status, or namespace. You can click  in the upper right corner of the list to export data of all containers or selected containers. The exported file is in .xlsx format, and the file name contains the timestamp.

Module	Description
Monitoring	This tab displays the resource usage of the pod in each dimension in the last 1 hour, last 8 hours, last 24 hours, or a custom period. To view more monitoring information, click **View Dashboard** to access the **Dashboard** tab. For details, see **Dashboard**.

9.1.3.7 Viewing Event Information

Kubernetes events show the cluster running status and resource scheduling status, helping O&M personnel observe resource changes and locate faults. To monitor events in a cluster, choose **Container Insights** > **Events**. You need to install log-agent in the cluster. log-agent can collect Kubernetes events and display them on the **Container Insights** > **Events** tab.

Navigation Path

Step 1 Log in to the UCS console.

Step 2 In the navigation pane, choose **Container Intelligent Analysis**. Choose **Container Insights** > **Clusters**, click the cluster name in **Cluster Statistics**, and click the **Events** tab.

----End

Viewing Event Details

The event details page has two tabs: **Overview** and **Events**. On the **Overview** tab, you can view the total number, trend, and sorting of events in the cluster. On the **Events** tab, you can view event details, such as the event name, type, content, and information about the resource that triggers the event.

Table 9-7 Tabs on the event details page

Tab	Description
Overview	By default, the **Overview** tab displays the event statistics of all namespaces in the cluster. You can also select a namespace from the drop-down list in the upper right corner to view its event data. ● **Total Events**: displays the distribution of normal and warning events in a doughnut chart. ● **Top 5 Warning Events by Resource**: displays the resource information corresponding to the number of top 5 warning events. ● **Warning Events by Resource Type**: displays the comparison between the number of warning events and the number of warning events in the last 24 hours. ● **Warning Event Trend (24 Hours)**: displays the trend of the number of warning events in the last 24 hours. ● **Normal Event Trend (24 Hours)**: displays the trend of the number of normal events in the last 24 hours. ● **Top 10 Events in 24 Hours**: displays the names of top 10 events in the last 24 hours.
Events	The **Events** tab displays cluster event details in a unit of time, including the event name, type, content, and information about the resource that triggers the event. **Searching for Events** The **Events** tab displays the event information of a specified resource that is searched out based on certain conditions, including the trend and details of normal and warning events. In this way, you can conveniently view the event information related to the resource. Search for events in any of the following ways: ● Enter the name of the event to be searched for in the text box, select a namespace or event type, and click **Search**. ● Click **Advanced Search** and enter the desired workload, node, pod, event content, resource type, or resource name. ● Select a time interval in the upper left corner to view the events generated in that period, including last hour, last day, last week, and a custom interval. **Event List** You can view details about events that meet your search criteria in the list. The details include the last occurrence time, event name, resource type, resource name, event content, event type, and occurrence times. Click **Historical Events** in the **Operation** column. A dialog box is displayed to show all events of the current resource type and resource.

9.1.4 Health Diagnosis

Overview

An important function of CIA is to diagnose the health of clusters. CIA automatically checks whether clusters, nodes, workloads, core add-ons, and external dependencies are healthy based on cluster configurations and metrics reported by the kube-prometheus-stack add-on to AOM. CIA also provides diagnosis results and rectification suggestions for abnormal items based on best O&M practices of Kubernetes clusters.

Constraints

- The cluster version is later than v1.17.
- The clusters are in the **Running** state.

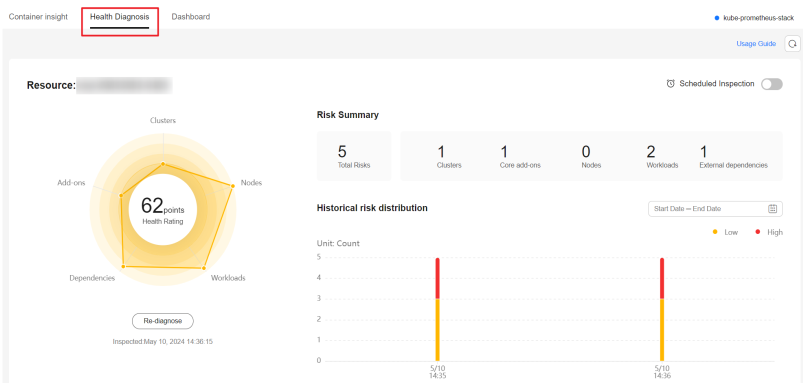
Viewing Health Diagnosis Results

Step 1 Select a fleet or a cluster that is not added to the fleet.

Figure 9-13 Selecting a fleet or a cluster not in the fleet

Step 2 Click the **Health Diagnosis** tab to view the numbers of normal clusters and risky clusters.

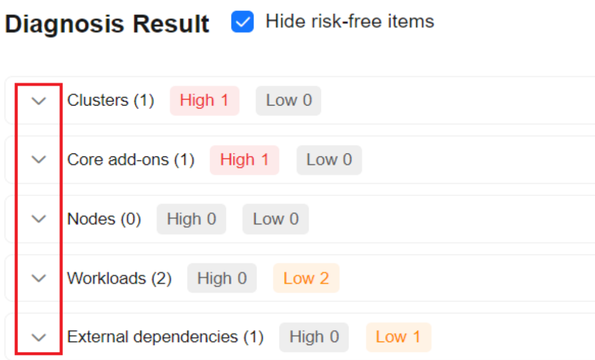
Figure 9-14 Health diagnosis



Step 3 In **Diagnosis Result**, view the diagnosis results of the current cluster.

Click and click **View Diagnosis Details** to access the diagnosis details page and view diagnosis items and results.

Figure 9-15 Diagnosis results



----End

Configuring a Scheduled Inspection

Step 1 Select a fleet or a cluster that is not added to the fleet.

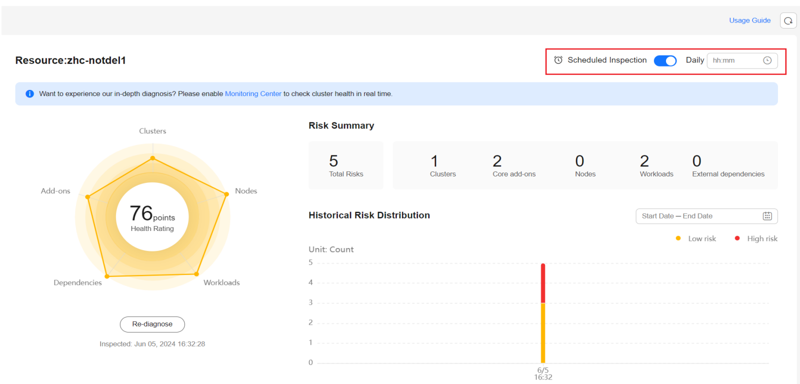
Figure 9-16 Selecting a fleet or a cluster not in the fleet

Step 2 Choose **Container insight** > **Clusters** to view the clusters for which monitoring has been enabled.

Step 3 Click **Health Diagnosis**, enable **Scheduled Inspection** in the upper right corner, and configure the start time of the inspection.

The inspection will automatically start at the specified time. A cluster can be scheduled to be inspected only once every day.

Figure 9-17 Scheduled inspection configuration



NOTE

You can also go to the inspection details page of a cluster as instructed in **Viewing Health Diagnosis Results**.

----End

Health Diagnosis

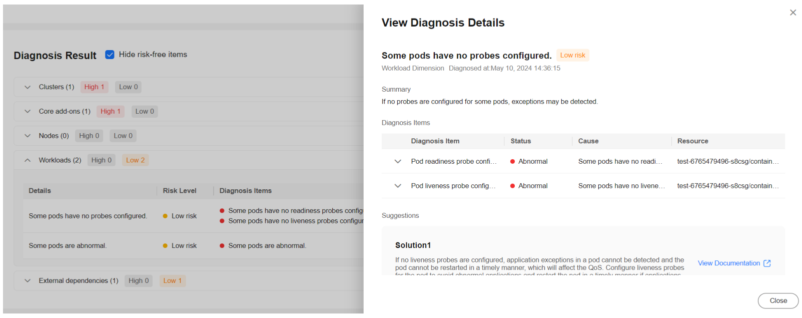
Step 1 Go to the inspection details page of a cluster as instructed in **Viewing Health Diagnosis Results**.

Step 2 In the **Cluster Inspection** area, select the cluster that is not inspected and click **Diagnose Now**.

After the diagnosis is complete, the page will be automatically refreshed to display the diagnosis results. Normal items are hidden by default.

Kubernetes problems will be summarized from the abnormal items. Troubleshooting suggestions will also be provided. You can click **View Diagnosis Details** to view the diagnosis details and rectification suggestions of a specific diagnosis item.

Figure 9-18 Diagnosis details



----End

Inspection Items

Table 9-8 Inspection items for CCE clusters

Dimension	Scenario	Inspection Item
Cluster	Cluster resource planning	Whether HA is enabled for master nodes
		Whether the CPU requests of pods in the cluster have exceeded 80% of the cluster CPU
		Whether the CPU limits of pods in the cluster have exceeded 150% of the cluster CPU

		Whether the memory requests of pods in the cluster have exceeded 80% of the cluster memory
		Whether the memory limits of pods in the cluster have exceeded 150% of the cluster memory
		Whether the cluster version has expired
	Cluster O&M	Whether kube-prometheus-stack is normal
		Whether log-agent is normal
		Whether npd is normal
	Cluster configuration	Whether security groups are correctly configured
Core add-ons	Whether coredns status is normal	Whether the CPU usage of coredns has exceeded 80% in the last 24 hours
		Whether the memory usage of coredns has exceeded 80% in the last 24 hours
		Whether coredns failed to resolve domain names for more than XX times in the last 24 hours
		Whether the P99 latency of coredns has exceeded 5s in the last 24 hours
		Whether coredns is normal
	Whether everest status is normal	Whether everest is normal
		Whether the CPU usage of everest has exceeded 80% in the last 24 hours
		Whether the memory usage of everest has exceeded 80% in the last 24 hours
	Whether kube-prometheus-stack status is normal	Whether the CPU usage of kube-prometheus-stack has exceeded 80% in the last 24 hours
		Whether the memory usage of kube-prometheus-stack has exceeded 80% in the last 24 hours
		Whether kube-prometheus-status is normal
		Whether OOM occurred on kube-prometheus-status in the last 24 hours

		Whether the PVC usage of prometheus-server has exceeded 80% when kube-prometheus-status is deployed in server mode
	Whether log-agent status is normal	Whether log-agent is normal
		Whether LTS log groups and log stream are created successfully
		Whether log structuring is enabled for LTS log groups
	autoscaler status	Whether autoscaler is available when auto scaling is enabled for node pools
Node	Node status	Whether nodes are ready
		Whether nodes can be scheduled
		Whether kubelet is normal
	Node configuration	Whether the memory requests of pods on a node have exceeded 80% of the node memory
		Whether the CPU requests of pods on a node have exceeded 80% of the node CPU
		Whether the memory limits of pods on a node have exceeded 150% of the node memory
		Whether the CPU limits of pods on a node have exceeded 150% of the node CPU
	Resource requests and limits of nodes	Whether the CPU usage of a node has exceeded 80% in the last 24 hours
		Whether the memory usage of a node has exceeded 80% in the last 24 hours
		Whether the disk usage of a node has exceeded 80%
		Whether the number of PIDs for a node exceeds the limit
		Whether OOM has occurred on a node in the last 24 hours
Workload	Pod status	Whether pods are normal
	Pod workload	Whether OOM has occurred on a pod in the last 24 hours

		Whether the CPU usage of a pod has exceeded 80% in the last 24 hours
		Whether the memory usage of a pod has exceeded 80% in the last 24 hours
	Pod configuration	Whether requests are configured for containers in a pod
		Whether limits are configured for containers in a pod
	Pod probe configuration	Whether liveness probes are configured for containers in a pod
		Whether readiness probes are configured for containers in a pod
External dependency	Resource quotas of a node	Whether 90% or more of the EVS disk quota has been used
		Whether 90% or more of the ECS quota has been used

Table 9-9 Inspection items for on-premises clusters

Dimension	Scenario	Inspection Item
Cluster	Cluster resource planning	Whether HA is enabled for master nodes
		Whether the CPU requests of pods in the cluster have exceeded 80% of the cluster CPU
		Whether the CPU limits of pods in the cluster have exceeded 150% of the cluster CPU
		Whether the memory requests of pods in the cluster have exceeded 80% of the cluster memory
		Whether the memory limits of pods in the cluster have exceeded 150% of the cluster memory
	Cluster O&M	Whether kube-prometheus-stack is normal
		Whether log-agent is normal
Core add-ons	Whether kube-prometheus-stack status is normal	Whether the CPU usage of kube-prometheus-stack has exceeded 80% in the last 24 hours

		Whether the memory usage of kube-prometheus-stack has exceeded 80% in the last 24 hours
		Whether kube-prometheus-status is normal
		Whether OOM occurred on kube-prometheus-status in the last 24 hours
	Whether log-agent status is normal	Whether log-agent is normal
		Whether LTS log groups and log stream are created successfully
		Whether log structuring is enabled for LTS log groups
Node	Node status	Whether nodes are ready
		Whether nodes can be scheduled
		Whether kubelet is normal
	Node configuration	Whether the memory requests of pods on a node have exceeded 80% of the node memory
		Whether the CPU requests of pods on a node have exceeded 80% of the node CPU
		Whether the memory limits of pods on a node have exceeded 150% of the node memory
		Whether the CPU limits of pods on a node have exceeded 150% of the node CPU
	Resource requests and limits of nodes	Whether the CPU usage of a node has exceeded 80% in the last 24 hours
		Whether the memory usage of a node has exceeded 80% in the last 24 hours
		Whether the disk usage of a node has exceeded 80%
		Whether the number of PIDs for a node exceeds the limit
		Whether OOM has occurred on a node in the last 24 hours
Workload	Pod status	Whether pods are normal

	Pod workload	Whether OOM has occurred on a pod in the last 24 hours
		Whether the CPU usage of a pod has exceeded 80% in the last 24 hours
		Whether the memory usage of a pod has exceeded 80% in the last 24 hours
	Pod configuration	Whether requests are configured for containers in a pod
		Whether limits are configured for containers in a pod
	Pod probe configuration	Whether liveness probes are configured for containers in a pod
		Whether readiness probes are configured for containers in a pod
External dependency	Resource quotas of a node	Whether 90% or more of the EVS disk quota has been used
		Whether 90% or more of the ECS quota has been used

Table 9-10 Inspection items for attached clusters, multi-cloud clusters, and partner cloud clusters

Dimension	Scenario	Inspection Item
Cluster	Cluster resource planning	Whether HA is enabled for master nodes
		Whether the CPU requests of pods in the cluster have exceeded 80% of the cluster CPU
		Whether the CPU limits of pods in the cluster have exceeded 150% of the cluster CPU
		Whether the memory requests of pods in the cluster have exceeded 80% of the cluster memory
		Whether the memory limits of pods in the cluster have exceeded 150% of the cluster memory
	Cluster O&M	Whether kube-prometheus-stack is normal

Core add-ons	Whether kube-prometheus-stack status is normal	Whether the CPU usage of kube-prometheus-stack has exceeded 80% in the last 24 hours
		Whether the memory usage of kube-prometheus-stack has exceeded 80% in the last 24 hours
		Whether kube-prometheus-status is normal
		Whether OOM occurred on kube-prometheus-status in the last 24 hours
Node	Node status	Whether nodes are ready
		Whether nodes can be scheduled
		Whether kubelet is normal
	Node configuration	Whether the memory requests of pods on a node have exceeded 80% of the node memory
		Whether the CPU requests of pods on a node have exceeded 80% of the node CPU
		Whether the memory limits of pods on a node have exceeded 150% of the node memory
		Whether the CPU limits of pods on a node have exceeded 150% of the node CPU
	Resource requests and limits of nodes	Whether the CPU usage of a node has exceeded 80% in the last 24 hours
		Whether the memory usage of a node has exceeded 80% in the last 24 hours
		Whether the disk usage of a node has exceeded 80%
		Whether the number of PIDs for a node exceeds the limit
		Whether OOM has occurred on a node in the last 24 hours
Workload	Pod status	Whether pods are normal
	Pod workload	Whether OOM has occurred on a pod in the last 24 hours
		Whether the CPU usage of a pod has exceeded 80% in the last 24 hours

		Whether the memory usage of a pod has exceeded 80% in the last 24 hours
	Pod configuration	Whether requests are configured for containers in a pod
		Whether limits are configured for containers in a pod
	Pod probe configuration	Whether liveness probes are configured for containers in a pod
		Whether readiness probes are configured for containers in a pod
External dependency	Resource quotas of a node	Whether 90% or more of the EVS disk quota has been used
		Whether 90% or more of the ECS quota has been used

9.1.5 Dashboard

With a dashboard, different graphs such as line graphs and digit graphs are displayed on the same screen, which lets you view comprehensive monitoring data.

Checking and Switching Views

Step 1 Select a fleet or a cluster that is not added to the fleet.

Figure 9-19 Selecting a fleet or a cluster not in the fleet

Step 2 The view is displayed by default after the **Dashboard** tab is selected.

Step 3 Configure related parameters for checking views. Parameters available for setting vary with views. See **Table 9-11** for details.

Step 4 Specify the view window.

Select or customize time segments in the upper right corner of the page, and click  to refresh the page.

Step 5 The CIA dashboard provides preset views. You can click the **Switch View** button next to the view name to select monitoring data to view. **Table 9-11** describes the preset views.

Table 9-11 Preset views

View Name	Parameter	Monitoring Metric Included
Cluster View (Default View)	Cluster	• Number of Nodes/Nodes with Unavailable Disks/Nodes Unavailable • CPU/Memory Usage • CPU/Memory Requests Commitment • CPU/Memory Limits Commitment • Num of Pods/Containers • CPU/Memory Usage • Network Receive/Transmit Rate • Average Network Receive/Transmit Rate • Rate of Received/Transmitted Packets • Packet Loss Rate (Receive/Transmit) • Disk IOPS (Read+Write) • Throughput (Read+Write)
APIServer View	• Cluster • Instance	• Alived • QPS • Request Success Rate (Read) • Requests Being Processed • Request Rate (Read/Write) • Request Error Rate (Read/Write) • P99 Request Latency (Read/Write) • Work Queue Growth Rate/Work Queue Depth • Work Queue Latency (P99) • Memory/CPU Usage • Goroutines

View Name	Parameter	Monitoring Metric Included
Pod View	• Cluster • Namespace • Pod	• Total Containers/Running Containers • Pod Status • Container Restarts • CPU/Memory Usage • CPU Throttling • Network Receive/Transmit Rate • Rate of Received/Transmitted Packets • Packet Loss Rate (Receive/Transmit) • Disk IOPS (Read+Write) • Throughput (Read+Write) • File System Usage/Used
Host View	• Cluster • Node	• CPU/Memory Usage • Load Average • Memory Usage • Disk Written/Read • Disk Space Usage • Disk I/O
k8s-node	• Cluster • Node	• CPU/Memory Usage • CPU/Memory Requests Commitment • CPU/Memory Limits Commitment • Memory Usage • Network Receive/Transmit Rate • Rate of Received/Transmitted Packets (Pod) • Rate of Received/Transmitted Packets • Packet Loss Rate (Receive/Transmit) • Disk IOPS (Read+Write) • Throughput (Read+Write)

View Name	Parameter	Monitoring Metric Included
CoreDNS	• Cluster • Instance	• Request Rate (by qtype/zone/DO bit) • Request Packet (UDP/TCP) • Response Rate (by rcode) • Response Rate (duration) • Response Packet (UDP/TCP) • Cache (size) • Cache (hitrate)
PVC View (CCE Clusters Only)	• Cluster • Namespace • PV • PVC	• PV/PVC Status • Used PVC/PVC Usage • Used PVC Inodes/PVC Inodes Usage • Hourly/Daily/Weekly PVC Usage • Volumes Full in Week Based on Daily Use Rate
kubelet	• Cluster • Instance	• Running Kubelets/Pods/Containers • Actual Volumes/Expected Volumes/Configuration Errors • Operation Rate/Error Rate/Latency • Pod Startup Rate/Latency (P99) • Storage Operation Rate/Error Rate/Latency (P99) • Cgroup Manager Operation Rate/Latency (P99) • PLEG Relist Rate/Interval/Latency (P99) • RPC Rate • Request Latency (P99) • Memory/CPU Usage • Goroutines
Prometheus	• Cluster • Job • Instance	• Target Sync Interval • Targets • Average Pull Interval • Pull Failures • Appended Samples • Series/Chunks in the Head • Query Rate/Query Duration

View Name	Parameter	Monitoring Metric Included
Prometheus Remote Write	• Cluster • Instance • url	• Highest Timestamp In vs. Highest Timestamp Sent • Rate5m • Rate in vs. succeeded or dropped 5m • Current/Maximum/Minimum/Expected Shards • Shard Size • Pend Samples • Current Segment of TSDB/Remote Write • Sample Discard/Failure/Retry Rate • Retry Rate of Enqueuing
Workload	• Cluster • Namespace • Type • Workload	• CPU/Memory Usage • Network Receive/Transmit Rate • Average Network Receive/Transmit Rate • Rate of Received/Transmitted Packets • Packet Loss Rate (Receive/Transmit)

View Name	Parameter	Monitoring Metric Included
xGPU View	Cluster	• Cluster - xGPU Device GPU Memory Usage • Cluster - xGPU Device GPU Compute Usage • Node - xGPU Device GPU Memory Usage • Node - xGPU Device Compute Usage • Node - Number of xGPU Devices • Node - Allocated GPU Memory of xGPU Devices • GPU - xGPU Device GPU Memory Usage • GPU - Allocated GPU Memory of xGPU Devices • GPU - GPU Memory Allocation Rate of xGPU Devices • GPU - xGPU Device Compute Usage • GPU - Number of xGPU Devices • GPU - Scheduling Policy • GPU - Number of Unhealthy xGPU Devices • Allocated Container GPU Memory • Container GPU Compute Usage • Used Container GPU Memory • Container GPU Memory Usage

----End

9.2 Logging

9.2.1 Overview

Kubernetes logs allow you to locate and rectify faults. This section describes how you can manage Kubernetes logs generated for UCS in the following ways:

- Use the cloud native logging add-on to collect application logs and report them to LTS, which provides log statistics and analysis. For details, see **Collecting Data Plane Logs**.
- Collect control plane component logs and Kubernetes audit logs from master nodes and add them to the LTS log streams in your account. For details, see **Collecting Control Plane Component Logs** and **Collecting Kubernetes Audit Logs**.

- Collect Kubernetes events and add them to the LTS log stream in your account for persistent storage and statistical analysis. For details, see **Collecting Kubernetes Events**.

Constraints

Logging is available only for **Huawei Cloud clusters** and **on-premises clusters**.

9.2.2 Enabling Logging

An add-on based on Fluent Bit and OpenTelemetry is provided for log and Kubernetes event collection. It supports CRD-based log collection policies, and collects and forwards standard output logs, container file logs, node logs, and Kubernetes events of containers in a cluster. It also reports all abnormal Kubernetes events and some normal Kubernetes events to AOM.

Constraints

- This add-on is available only for Huawei Cloud clusters or on-premises clusters v1.21 or later.
- A maximum of 50 log collection rules can be configured for each cluster.
- This add-on cannot collect .gz, .tar, and .zip logs.
- If the node storage driver is Device Mapper, the container file logs must be collected from the path where the data disk is attached to the node.
- If the container runtime is containerd, each standard output log cannot be in multiple lines. (This does not apply to log-agent v1.3.0 or later.)
- In each cluster, up to 10,000 single-line logs can be collected per second, and up to 2,000 multi-line logs can be collected per second.
- On each node, up to 4,096 logs can be collected.
- If a volume is attached to the data directory of a service container, this add-on cannot collect data from the parent directory. In this case, you need to configure a complete data directory.
- The container running time must be longer than 1 minute for log collection to prevent logs from being deleted too quickly.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota.

A network access mode is required for an on-premises cluster. If you select Direct Connect or VPN, the VPC endpoint will be billed based on how long you use it.

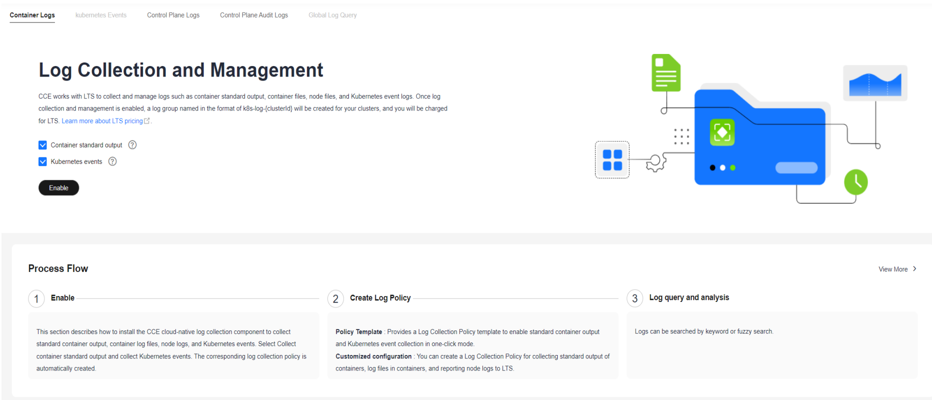
Log Collection

Step 1 Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page.

Step 2 Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.

Step 3 (Only for Huawei Cloud clusters) If you are not authorized, obtain required permissions first. In the displayed dialog box, click **Authorize**.

Step 4 (Only for Huawei Cloud clusters) Click **Enable** and wait for about 30 seconds until the log page is automatically displayed.



- Standard output logs: A log collection policy named **default-stdout** will be created, and standard output logs in all namespaces will be reported to LTS.
- Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.

Step 5 (Only for on-premises clusters) Obtain permissions required by the cloud native logging add-on. For details, see **Assigning Authorization for log-agent in Your On-Premises Cluster**.

Step 6 (Only for on-premises clusters) Click **Enable**. In the displayed dialog box, configure log collection and network parameters. Wait for about 30 seconds until the log page is automatically displayed.

Table 9-12 Log collection configuration and network configuration of on-premises clusters

Configuration	Description
Log collection	• Standard output logs: A log collection policy named **default-stdout** will be created, and standard output logs in all namespaces will be reported to LTS. • Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS and AOM. • Kubernetes audit logs: Kubernetes audit logs will be collected and reported to LTS. • kube-apiserver logs: Logs of the kube-apiserver component on the control plane will be collected and reported to LTS. • kube-controller-manager logs: Logs of the kube-controller-manager component on the control plane will be collected and reported to LTS. • kube-scheduler logs: Logs of the kube-scheduler component on the control plane will be collected and reported to LTS.

Configuration	Description
Network	• Public network: This option features flexibility, cost-effectiveness, and easy access. It is only available for clusters that can access the public network. • Direct Connect or VPN: After you connect an on-premises data center to a VPC over Direct Connect or VPN, you can use a VPC endpoint to access CIA over the private network. This option features high speed, low latency, and high security. For details, see **Using Direct Connect or VPN to Report Logs of On-Premises Clusters**.

----End

Troubleshooting

All components except log-operator are not ready, and the volume failed to be attached to the node.

Solution: Check the logs of log-operator. During add-on installation, the configuration files required by other components are generated by log-operator. If the configuration files are invalid, all components cannot be started.

9.2.3 Collecting Data Plane Logs

An add-on based on Fluent Bit and OpenTelemetry is provided for log and Kubernetes event collection. It supports CRD-based log collection policies, and collects and forwards standard output logs, container file logs, node logs, and Kubernetes events of containers in a cluster. It also reports all abnormal Kubernetes events and some normal Kubernetes events to AOM.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota.

Control Plane Components

There are two control plane log types. Each log stream corresponds to a component of the Kubernetes control plane. To learn more about these components, see **Kubernetes Components**.

Table 9-13 Control plane components

Log Type	Component	Log Stream	Description
Control plane component logs	default-stdout	stdout-{clusterID}	Standard output logs Default log group: **k8s-logs-** *{Cluster ID}*
	default-event	event-{clusterID}	Kubernetes events Default log group: **k8s-logs-** *{Cluster ID}*

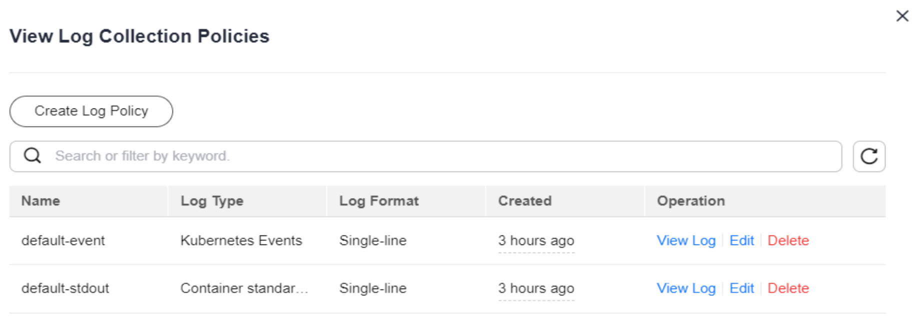
Log Collection

Step 1 View and configure log collection policies.

1. Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.

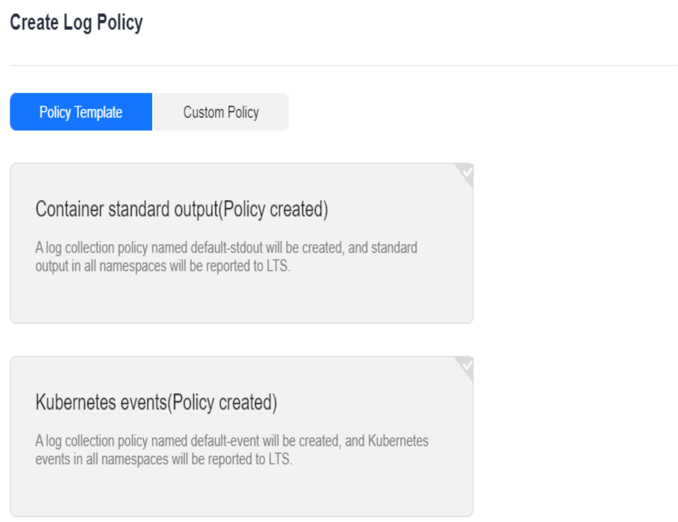
   Figure 9-20 Viewing log collection policies

   If **Container standard output** and **Kubernetes events** are selected during add-on installation, two log collection policies will be created, and the collected logs will be reported to the default log group and log streams.
3. Click **Create Log Policy** and configure parameters as required.

   Policy Template: If no log collection policy is selected during add-on installation or the log collection policy is deleted, you can use this option to create a default log collection policy.

Figure 9-21 Policy template



Custom Policy: You can use this option to create custom log collection policies.

Figure 9-22 Custom policy

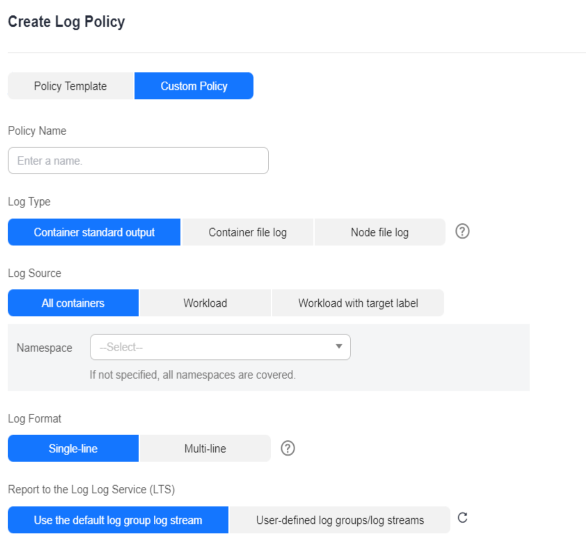


Table 9-14 Custom policy parameters

Parameter	Description
Log Type	Type of logs to be collected. – **Container standard output**: used to collect container standard output logs. You can create a log collection policy by namespace, workload name, or instance label. – **Container file log**: used to collect text logs. You can create a log collection policy by workload or instance label. – **Node file log**: used to collect logs from a node. Only one file path can be configured for a log collection policy.
Log Source	Containers whose logs are to be collected. – **All containers**: You can specify all containers in a namespace. If this parameter is not specified, logs of containers in all namespaces will be collected. – **Workload**: You can specify a workload and its containers. If this parameter is not specified, logs of all containers running the workload will be collected. – **Workload with target label**: You can specify a workload by label and its containers. If this parameter is not specified, logs of all containers running the workload will be collected.

Parameter	Description
Collection Path	Path of files where logs are to be collected. The path must start with a slash (/) and contain a maximum of 512 characters. Only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), slashes (/), asterisks (*), and question marks (?) are allowed. The file name can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), asterisks (*), question marks (?), and periods (.). Enter an absolute path for the log directory. Logs in the format of .gz, .tar, and .zip are not supported. A maximum of three levels of directories can be matched using wildcards. The level-1 directory does not support wildcards. The directory name and file name must be complete names and support asterisks (*) and question marks (?) as wildcards. An asterisk (*) can match multiple characters. A question mark (?) can match only one character. Example: – If the directory is **/var/logs/*** and the file name is ***.log**, any log files with the extension .log in all directories in the **/var/logs** directory will be reported. – If the directory is **/var/logs/app_*** and the file name is ***.log**, any log files with the extension .log in all directories that match app_* in the **/var/logs** directory will be reported. If a volume is attached to the data directory of a service container, this add-on cannot collect data from the parent directory. In this case, you need to configure a complete data directory. For example, if the data volume is attached to the **/var/log/service** directory, logs cannot be collected from the **/var/log** or **/var/log/*** directory. In this case, you need to set the collection directory to **/var/log/service**.

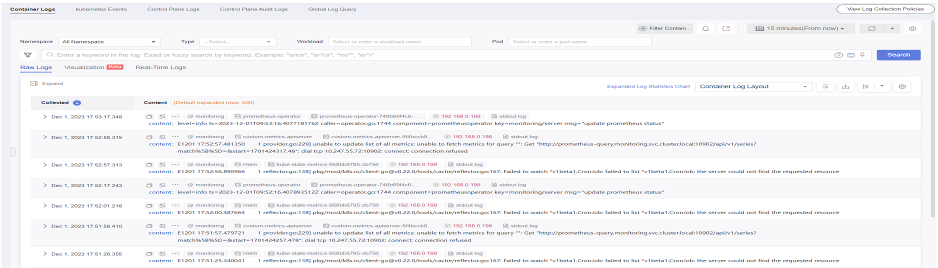
Parameter	Description
Log Format	– Single-line Each log contains only one line of text. The newline character \n denotes the start of a new log. – Multi-line Some programs (for example, Java program) print a log that occupies multiple lines. By default, logs are collected by line. If you want to display logs as a single message, you can enable multi-line logging and use the regular pattern. If you select the multi-line text, you need to enter the log matching format. Example: If logs need to be collected by line, enter **\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.***. The following three lines starting with the date are regarded as a log. 2022-01-01 00:00:00 Exception in thread "main" java.lang.RuntimeException: Something has gone wrong, aborting! at com.myproject.module.MyProject.badMethod(MyProject.java:22) at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
Report to LTS	This parameter is used to configure the log group and log stream for log reporting. – Default log groups/log streams: The default log group (**k8s-log-***{Cluster ID}*) and default log stream (**stdout-***{Cluster ID}*) are automatically selected. – Custom log groups/log streams: You can select any log group and log stream.
Log Group	A log group is the basic unit for LTS to manage logs. If you do not have a log group, CCE prompts you to create one. The default name is **k8s-log-***{Cluster ID}*, for example, **k8s-log-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**.
Log Stream	A log stream is the basic unit for log read and write. You can create log streams in a log group to store different types of logs for finer log management. When you install the add-on or create a log policy based on a template, the following log streams are automatically created: – **stdout-***{Cluster ID}* for container logs, for example, **stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3** – **event-***{Cluster ID}* for Kubernetes events, for example, **event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3**

4. Click **Edit** to modify an existing log collection policy.
5. Click **Delete** to delete an existing log collection policy.

Step 2 View the logs.

1. On the UCS console, choose **Fleets** and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. View different types of logs:
   – **Container Logs**: displays all logs in the default log stream **stdout-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*. You can search for logs by workload for a Huawei Cloud cluster.

   Figure 9-23 Querying container logs

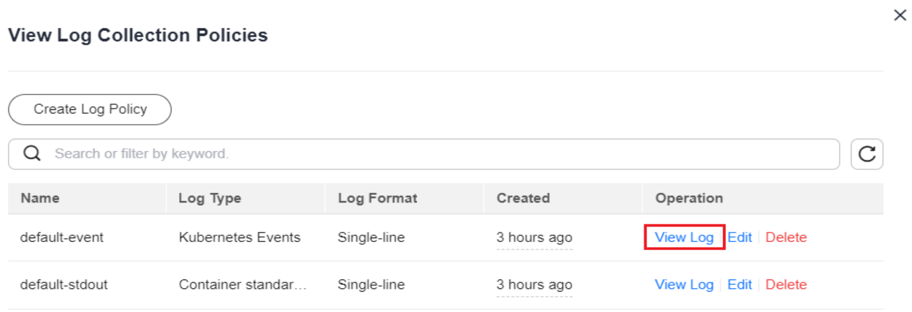
   – **Kubernetes Events**: displays all Kubernetes events in the default log stream **event-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   – **Control Plane Logs**: displays all logs of components on the control plane in the default log stream *{Component name}*-*{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   – **Control Plane Audit Logs**: displays all control plane audit logs in the default log stream **audit-***{Cluster ID}* of the default log group **k8s-log-***{Cluster ID}*.
   – **Global Log Query**: You can view logs in the log streams of all log groups. You can specify a log stream to view the logs. By default, the default log group **k8s-log-***{Cluster ID}* is selected. You can click the edit icon on the right of **Switching Log Groups** to switch to another log group.

   Figure 9-24 Global log query

3. Click **View Log Collection Policies** in the upper right corner. Locate the log collection policy and click **View Log** to go to the log list.

Figure 9-25 Viewing logs



----End

Troubleshooting

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

   Example:

   ```
   2023/05/05 12:17:20.799 [E] call 3 times failed, resion: create group failed, projectID: xxx, groupName: k8s-log-xxx, err: create groups status code: 400, response: {"error_code":"LTS.0104","error_msg":"Failed to create log group, the number of log groups exceeds the quota"}, url: https://lts.cn-north-4.myhuaweicloud.com/v2/xxx/groups, process will retry after 45s
   ```

   Solution: On the LTS console, delete unnecessary log groups. For details about the quota limit of log groups, see **Log Groups**.

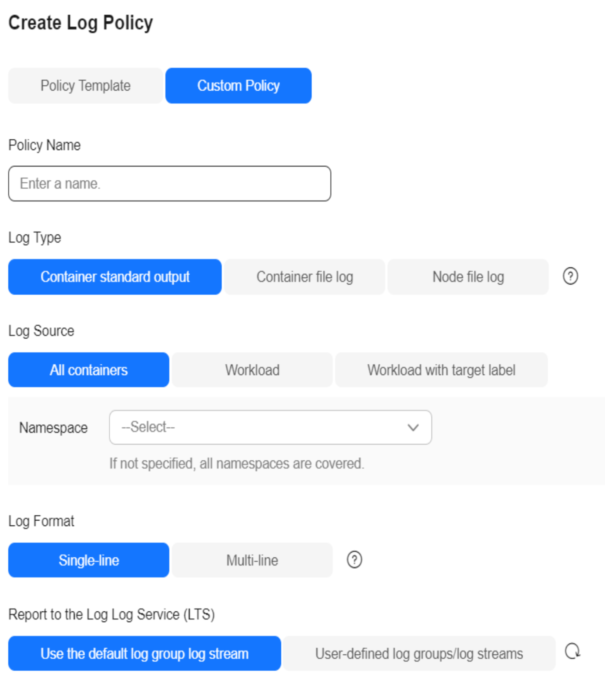
2. **A container file path is configured but is not mounted to the container, and Docker is used as the container engine. As a result, logs cannot be collected.**

   Solution:

   Check whether Device Mapper is used for the node where the workload resides. Device Mapper does not support text log collection. (This restriction has been displayed when you create a log collection policy, as shown in **Figure 9-26**.) To check this, perform the following operations:

   a. Go to the node where the workload resides.
   b. Run the **docker info | grep "Storage Driver"** command.
   c. If the value of **Storage Driver** is **devicemapper**, text logs cannot be collected.

Figure 9-26 Creating a log policy



3. **Logs cannot be reported, and "log's quota has full" is reported in the standard output log of the OTel component.**

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

Solution:

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

4. **Text logs cannot be collected because wildcards are configured for the collection directory.**

   Troubleshooting: Check the volume mounting status in the workload configuration. If a volume is attached to the data directory of a service container, this add-on cannot collect data from the parent directory. In this case, you need to set the collection directory to a complete data directory. For example, if the data volume is attached to the **/var/log/service** directory, logs cannot be collected from the **/var/log** or **/var/log/*** directory. In this case, you need to set the collection directory to **/var/log/service**.

   Solution: If the log generation directory is **/application/logs/***{Application name}***/*.log**, attach the data volume to the **/application/logs** directory and set the collection directory in the log collection policy to **/application/logs/*/*.log**.

9.2.4 Collecting Control Plane Component Logs

CCE supports logging for master nodes. On the **Control Plane Logs** tab, you can select one or more components (kube-controller-manager, kube-apiserver, and kube-scheduler) whose logs need to be reported to TLS.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota.

Constraints

- Huawei Cloud clusters must be of v1.21.7-r0 or later, v1.23.5-r0 or later, or v1.25.
- There is required LTS resource quota. For details about the default LTS quota, see **Basic Resources**.

Control Plane Components

There are three control plane log types. Each log stream corresponds to a component of the Kubernetes control plane. To learn more about these components, see **Kubernetes Components**.

Table 9-15 Control plane components

Log Type	Component	Log Stream	Description
Control plane component logs	kube-apiserver	kube-apiserver-{{clusterID}}	It exposes Kubernetes APIs. For more information, see **kube-apiserver**.
	kube-controller-manager	kube-controller-manager-{{clusterID}}	It manages controllers and embeds the core control loops shipped with Kubernetes. For more information, see **kube-controller-manager**.

Log Type	Component	Log Stream	Description
	kube-scheduler	kube-scheduler-{{clusterID}}	It manages when and where to run Pods in your cluster. For more information, see **kube-scheduler**.

Enabling Log Collection for an On-Premises Cluster

The cloud native logging add-on is not installed in a cluster.

When installing the cloud native logging add-on, you can select control plane component logs to create a default log collection policy, so that this add-on collects all component logs and reports them to LTS. For details about the add-on installation, see **Log Collection**.

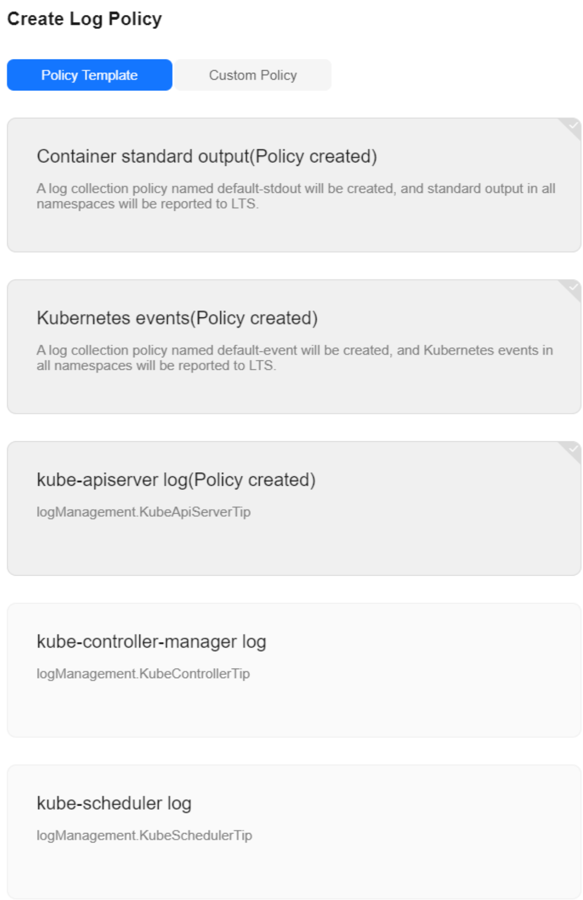
The cloud native logging add-on has been installed in a cluster.

1. Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. Click **View Log Collection Policies** in the upper right corner.

   All log collection policies reported to LTS are displayed.
3. Click **Create Log Policy** and configure parameters as required.

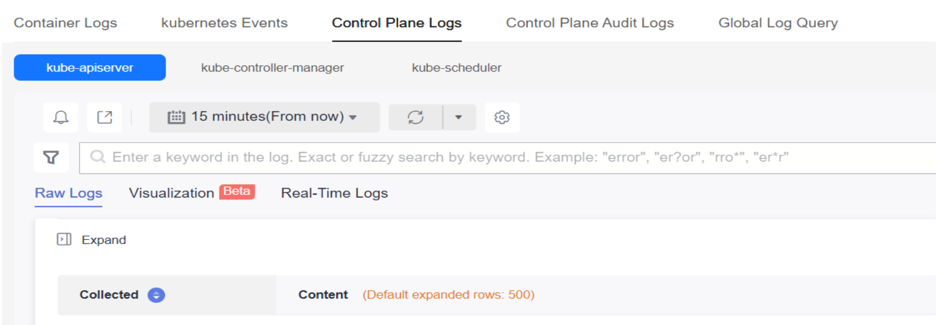
   Policy Template: If no log collection policy is selected during add-on installation or the log collection policy is deleted, you can use this option to create a default log collection policy.

Figure 9-27 Creating a log collection policy



4. On the **Logging** page, click the **Control Plane Logs** tab. Select the log stream configured in the log policy to view the logs reported to LTS.

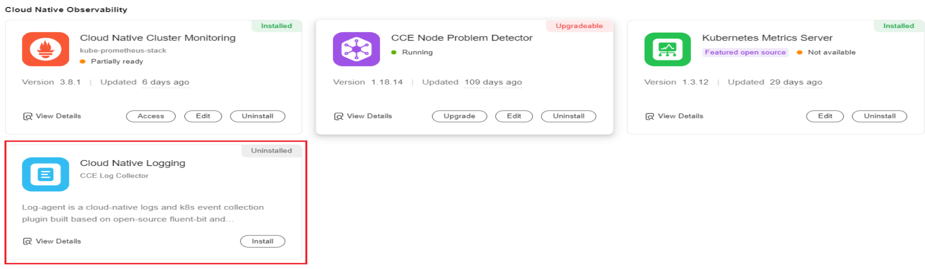
Figure 9-28 Viewing logs



Enabling Log Collection for a Huawei Cloud Cluster

Enabling log collection during cluster creation

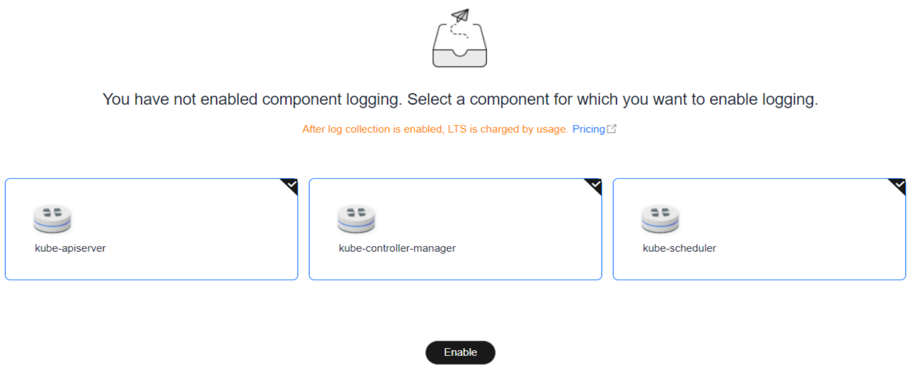
1. Log in to the CCE console.
2. Click **Buy Cluster** from the top menu.
3. On the **Select Add-on** page, select **Cloud Native Logging**.



4. On the **Add-on Configuration** page, select **Custom Installation** for **Cloud Native Logging** and then select control plane logs.
   - Standard output logs: A log collection policy named **default-stdout** will be created, and standard output logs in all namespaces will be reported to LTS.
   - Kubernetes events: A log collection policy named **default-event** will be created, and Kubernetes events in all namespaces will be reported to LTS.
5. Click **Next: Confirm** in the lower right corner. In the displayed dialog box, click **Submit**.

Enabling log collection for an existing cluster

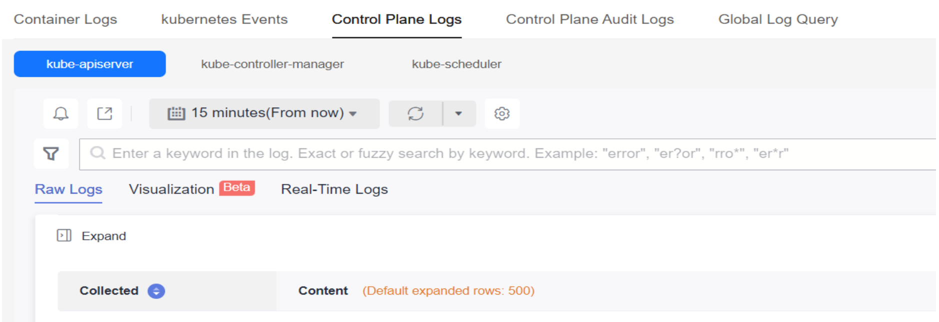
1. Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. Click the **Control Plane Logs** tab, select the control plane components whose logs need to be collected, and click **Enable**.



Viewing Control Plane Component Logs

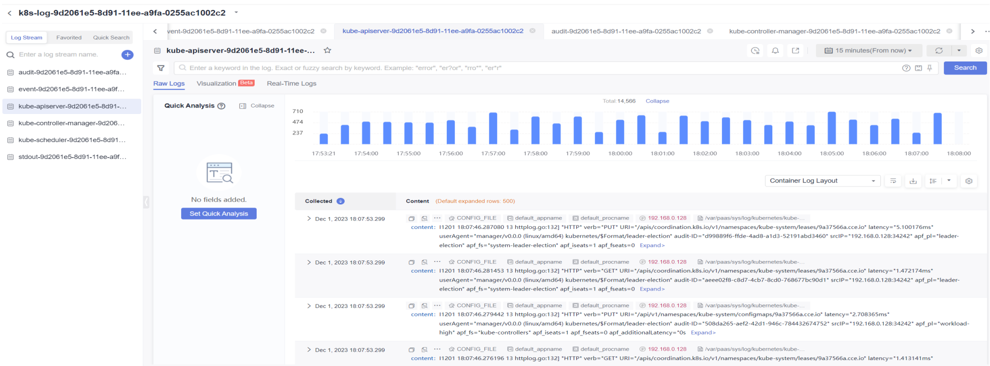
Viewing control plane component logs on the UCS console

1. Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. Click the **Control Plane Logs** tab and select the component whose logs to be viewed. For details about available control plane log types, see **Control Plane Components**. For details about operations, see **LTS User Guide**.



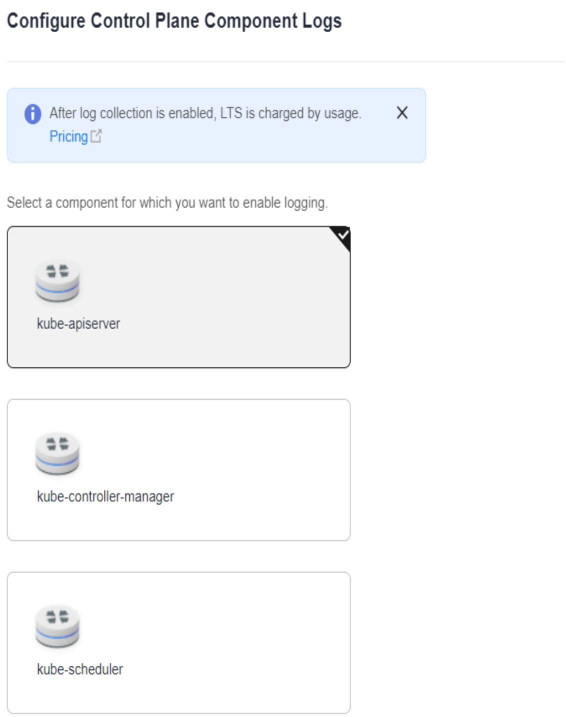
Viewing control plane component logs on the LTS console

1. Log in to the LTS console and choose **Log Management**.
2. Query the log group based on the cluster ID and click the log group name to view the log stream. For details, see **LTS User Guide**.



Disabling Log Collection of a Huawei Cloud Cluster

1. Log in to the UCS console, choose **Fleets**, and click the fleet name to access the fleet details page. Choose **Container Clusters**, click the cluster name to access the cluster details page, and choose **Logging**.
2. Click the **Control Plane Logs** tab, click **Configure Control Plane Component Logs** in the upper right corner, and modify the log settings.



3. Determine whether to enable logging for each component and click **OK**.

NOTE

After you disable control plane logging, logs are no longer written to the original log stream, but the existing logs will not be deleted and expenses may be incurred for this.

9.2.5 Collecting Kubernetes Audit Logs

CCE supports logging for master nodes. On the **Kubernetes Events** tab, you can select the audit component whose logs to be reported to LTS.

Constraints

- Huawei Cloud clusters must be of v1.21.7-r0 or later, v1.23.5-r0 or later, or v1.25.
- There is required LTS resource quota. For details about the default LTS quota, see **Basic Resources**.

Kubernetes Audit Logs

Table 9-16 Kubernetes audit logs

Log Type	Component	Log Stream	Description
Control plane audit logs	audit	audit-{{clusterID}}	An audit log is a chronological record of user operations on Kubernetes APIs and control plane activities for security.

Enabling Log Collection for an On-Premises Cluster

The cloud native logging add-on is not installed in a cluster.

When installing the cloud native logging add-on, you can select control plane audit logs to create a default log collection policy, so that this add-on collects logs and reports them to LTS. For details about the add-on installation, see **Log Collection**.

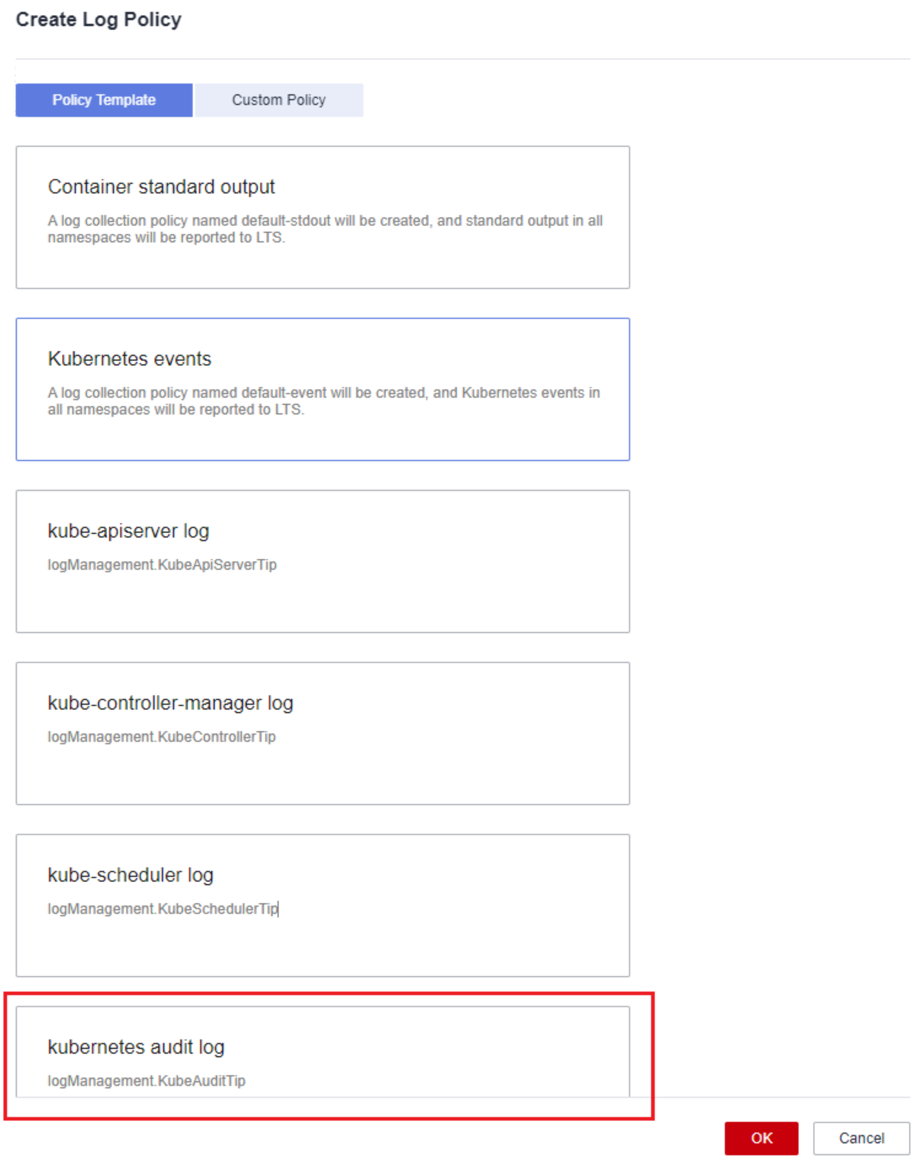
The cloud native logging add-on has been installed in a cluster.

Step 1 Access the fleet console. In the navigation pane on the left, choose **Container Clusters**. Then, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.

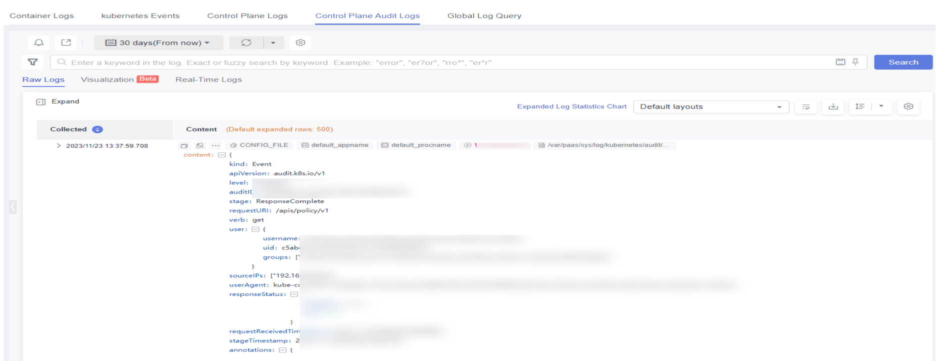
Step 2 Click **View Log Collection Policies** in the upper right corner. All log policies reported to LTS in the current cluster are displayed.

Step 3 Click **Create Log Policy** and configure parameters as required.

Policy Template: If no log collection policy is selected during add-on installation or the log collection policy is deleted, you can use this option to create a default log collection policy.



Step 4 On the **Logging** page, click the **Control Plane Audit Logs** tab. Select the log stream configured in the log policy to view the logs reported to LTS.



----End

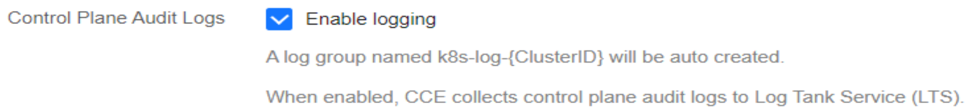
Enabling Log Collection for a Huawei Cloud Cluster

Enabling log collection during cluster creation

Step 1 Log in to the CCE console.

Step 2 Click **Buy Cluster** from the top menu.

Step 3 On the **Add-on Configuration** page, check the box of **Enable logging** for **Control Plane Audit Logs**.



----End

Enabling log collection for an existing cluster

Step 1 Access the fleet console. In the navigation pane on the left, choose **Container Clusters**. Then, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.

Step 2 Click the **Control Plane Audit Logs** tab, select the audit component, and click **Enable Logging**.

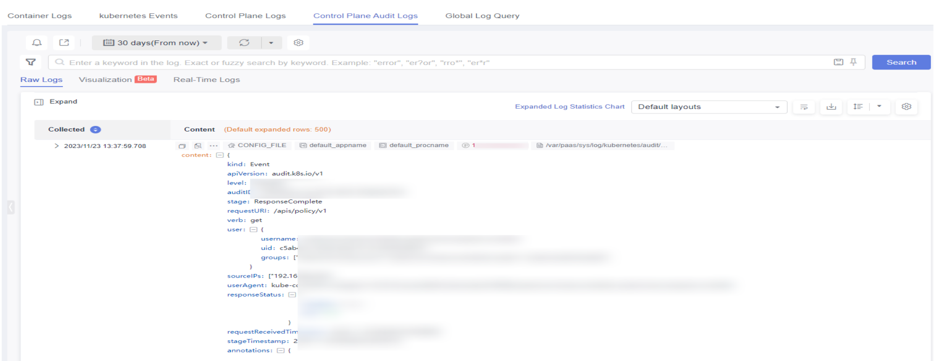
----End

Viewing Control Plane Audit Logs

Viewing control plane audit logs on the UCS console

Step 1 Access the fleet console. In the navigation pane on the left, choose **Container Clusters**. Then, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.

Step 2 Click the **Control Plane Audit Logs** tab and select a component for which you want to enable audit logs. For details about operations, see **LTS User Guide**.

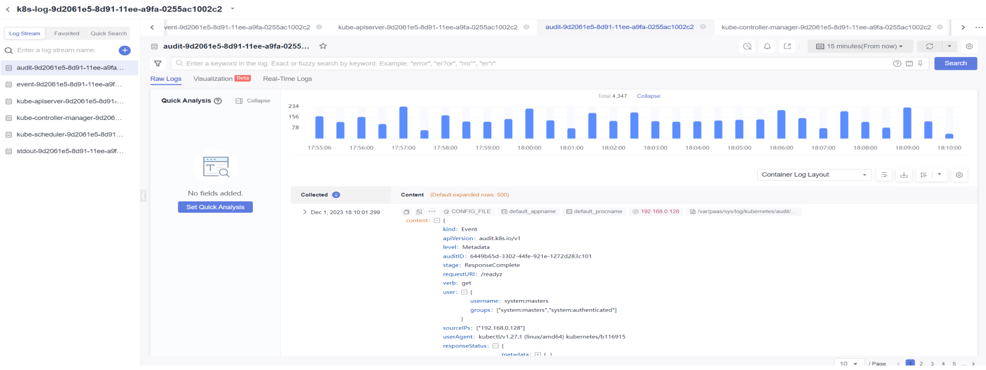


----End

Viewing control plane audit logs on the TLS console

Step 1 Log in to the LTS console and choose **Log Management**.

Step 2 Query the log group based on the cluster ID and click the log group name to view the log stream. For details, see **LTS User Guide**.

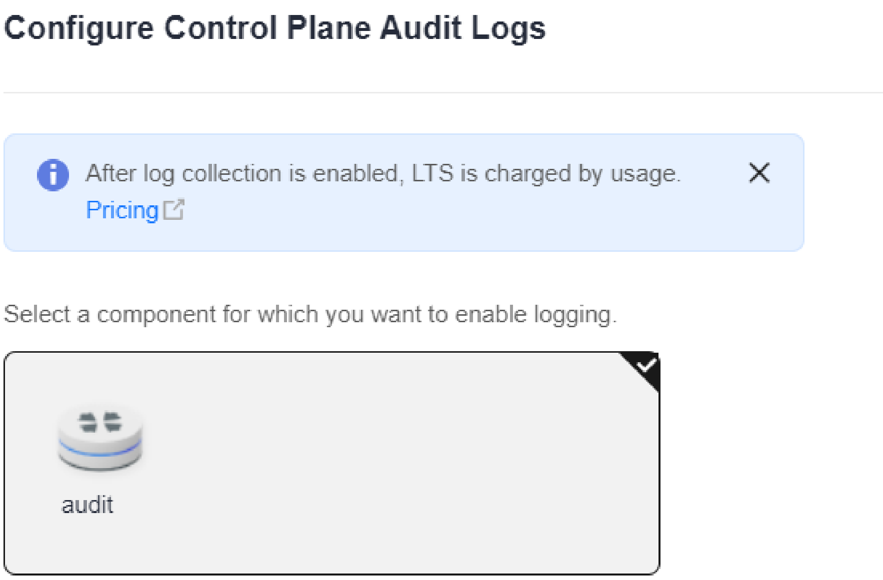


----End

Disabling Log Collection of a Huawei Cloud Cluster

Step 1 Access the fleet console. In the navigation pane on the left, choose **Container Clusters**. Then, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Logging**.

Step 2 Click the **Control Plane Audit Logs** tab and click **Configure Control Plane Audit Logs** to modify the log settings.



Step 3 Deselect **audit** and click **OK**.

NOTE

After you disable control plane audit logging, logs are no longer written to the original log stream, but the existing logs will not be deleted and expenses may be incurred for this.

----End

9.2.6 Collecting Kubernetes Events

The cloud native logging add-on works with LTS to collect and store Kubernetes events and works with AOM to generate alarms.

Billing

LTS does not charge you for creating log groups and offers a free quota for log collection every month. You pay only for log volume that exceeds the quota.

Reporting Kubernetes Events to LTS

The cloud native logging add-on is not installed in a cluster.

During add-on installation, you can select Kubernetes events to create a default log collection policy, so that this add-on collects all events and reports them to LTS. For details about the add-on installation, see **Collecting Data Plane Logs**.

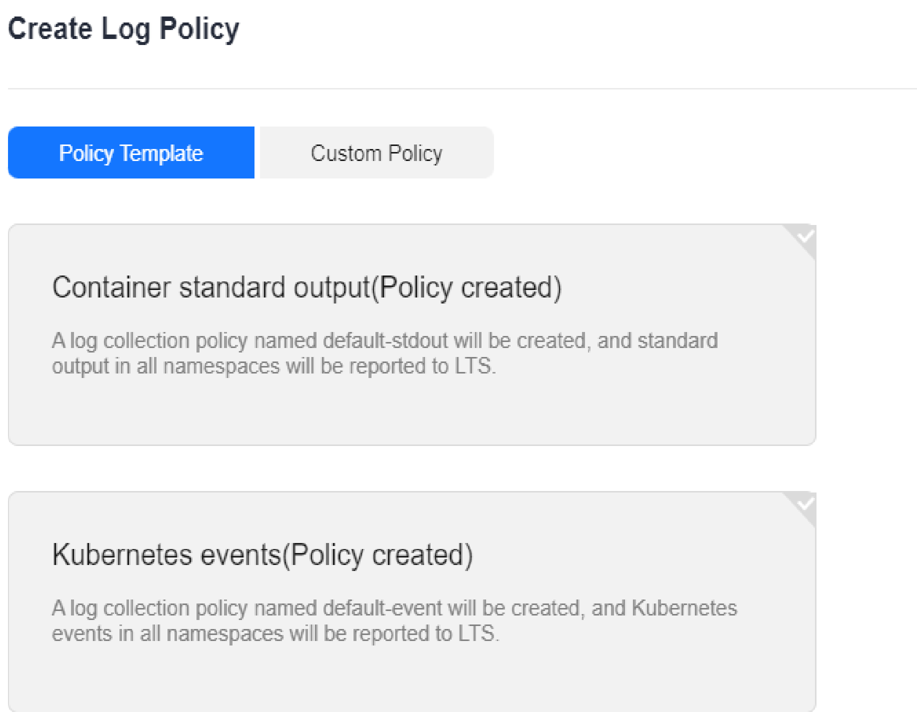
The cloud native logging add-on has been installed in a cluster.

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

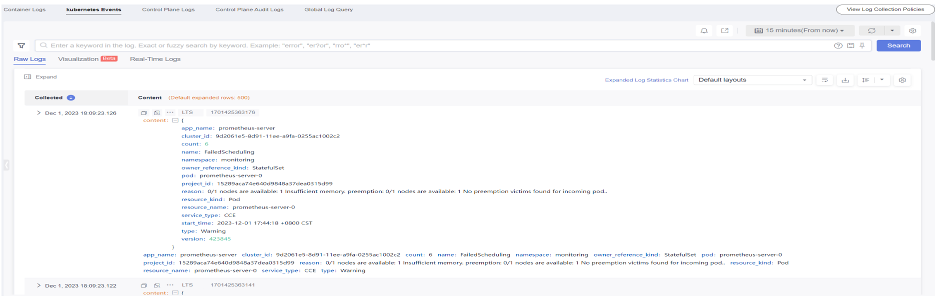
Step 2 Click **View Log Collection Policies** in the upper right corner. All log policies reported to LTS in the current cluster are displayed.

Step 3 Click **Create Log Policy** and configure parameters as required.

Policy Template: If **Kubernetes events** is not selected during add-on installation or the log collection policy is deleted, you can use this option to create a default log collection policy.



Step 4 On the logging management page, select the log stream configured in the log collection policy to view the events reported to LTS.



----End

Reporting Kubernetes Events to AOM

For a Huawei Cloud cluster of version 1.19.16, 1.21.11, 1.23.9, or 1.25.4, after the cloud native logging add-on is installed, all Warning events and some Normal events are reported to AOM by default. The reported events can be used to configure alarms. For details about the add-on installation, see **Collecting Data Plane Logs**.

You can enable or disable this function when installing the add-on for an on-premises cluster.

Custom Event Reporting

If the reported events cannot meet requirements, you can modify the settings for the events.

Step 1 Run the following command on the cluster to modify the event collection settings:

kubectl edit logconfig -n kube-system default-event-aom

Step 2 Modify the event collection settings as required.

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    # Settings on UCS from which events are collected
    type: event    # Type of logs to be collected. Do not change the value.
    event:
      normalEvents:    # Used to configure normal events
        enable: true    # Whether to enable normal event collection
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events will
be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all
events will be collected.
        - NotTriggerScaleUp
      warningEvents:    # Used to configure warning events
        enable: true    # Whether to enable warning event collection
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events will
be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all
events will be collected.
```

```
      - NotTriggerScaleUp
  outputDetail:
    type: AOM    # Type of the system that receives the events. Do not change the value.
    AOM:
      events:
      - name: DeleteNodeWithNoServer    # Event name. This parameter is mandatory.
        resourceType: Namespace    # Type of the resource that operations are performed on.
        severity: Major    # Event severity after an event is reported to AOM, which can be Critical, Major,
Minor, or Info. The default value is Major.
```

----End

9.2.7 Cloud Native Logging Add-on

When logging is enabled (**Enabling Logging**), log-agent is automatically installed for an on-premises cluster. You can also manually install this add-on by referring to this section. For details about this add-on, see **Cloud Native Logging**.

Overview

log-agent is an add-on based on Fluent Bit and OpenTelemetry for cloud native logging. It supports CRD-based log collection policies, collects and forwards standard output logs, container file logs, node logs, and Kubernetes events of containers in a cluster. After the add-on is installed, standard output logs and Kubernetes events are collected by default. For details about how to use log-agent to collect logs, see **Collecting Data Plane Logs**.

Constraints

The following are constraints on using log-agent:

- log-agent is available only in clusters v1.21 or later.
- A maximum of 50 log collection rules can be configured for each cluster.
- log-agent cannot collect .gz, .tar, and .zip logs.
- If the node storage driver is Device Mapper, the container file logs must be collected from the path where the data disk is attached to the node.
- If the container runtime is containerd, each standard output log cannot be in multiple lines.
- In each cluster, up to 10,000 single-line logs can be collected per second, and up to 2,000 multi-line logs can be collected per second.
- The container running time must be longer than 1 minute for log collection to prevent logs from being deleted too quickly.

Permissions

The fluent-bit component of the log-agent add-on reads and collects the standard output logs on each node, file logs in pods, and node logs based on the collection configuration.

The following permissions are required for running the fluent-bit component:

- CAP_DAC_OVERRIDE: ignores the discretionary access control (DAC) restrictions on files.
- CAP_FOWNER: ignores the restrictions that the file owner ID must match the process user ID.

- DAC_READ_SEARCH: ignores the DAC restrictions on file reading and catalog research.
- SYS_PTRACE: allows all processes to be traced.

Assigning Authorization for log-agent in Your On-Premises Cluster

The log-agent add-on needs to be authenticated before accessing LTS and AOM. This add-on uses Workload Identity to allow workloads in your on-premises cluster to impersonate IAM service accounts to access cloud services.

Workload Identity allows you to configure the public key of your cluster for the IAM IdP and add a mapping rule to map a ServiceAccount to an IAM service account. During workload deployment, the token of the ServiceAccount is mounted to the workload. This token is used to access cloud services. This way, the AK/SK of the IAM service account is not required, reducing security risks.

Step 1 Obtain the JSON Web Key Set (JWKS) of the on-premises cluster, which is used to verify the ServiceAccount token issued by ClusterIssuer.

1. Use kubectl to access the on-premises cluster.
2. Run the following command to obtain the public key:

   kubectl get --raw /openid/v1/jwks

   A json string is returned, containing the signature public key of the cluster for accessing the IdP.

```
{
    "keys": [
        {
            "kty": "RSA",
            "e": "AQAB",
            "use": "sig",
            "kid": "Ew29q....",
            "alg": "RS256",
            "n": "peJdm...."
        }
    ]
}
```

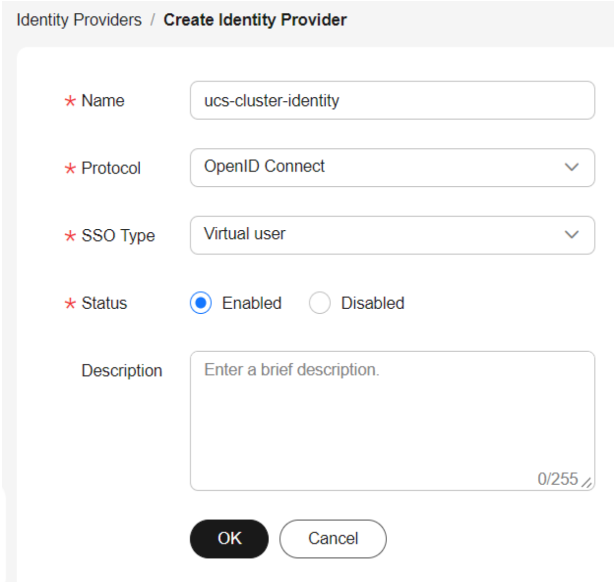
Step 2 Configure an IdP entity for your on-premises cluster on IAM.

1. Log in to the IAM console, query the ID of the project that the on-premises cluster belongs to, create an identity provider, and select **OpenID Connect** for **Protocol**. Enter the IdP name for log-agent. For details, see **Table 9-17**.

Table 9-17 log-agent IdP settings

Add-on Name	IdP Name	Client ID	Namespace	ServiceAccount Name	Minimum Permissions on User Groups
log-agent	ucs-cluster-identity-{Project ID}	ucs-cluster-identity	monitoring	log-agent-serviceaccount	aom:alarm:* lts:*:*

Figure 9-29 Modifying identity provider information



2. Click **OK** and modify the IdP information as described in **Table 9-18**. Click **Create Rule** to create an identity conversion rule.

Figure 9-30 Modifying identity provider information

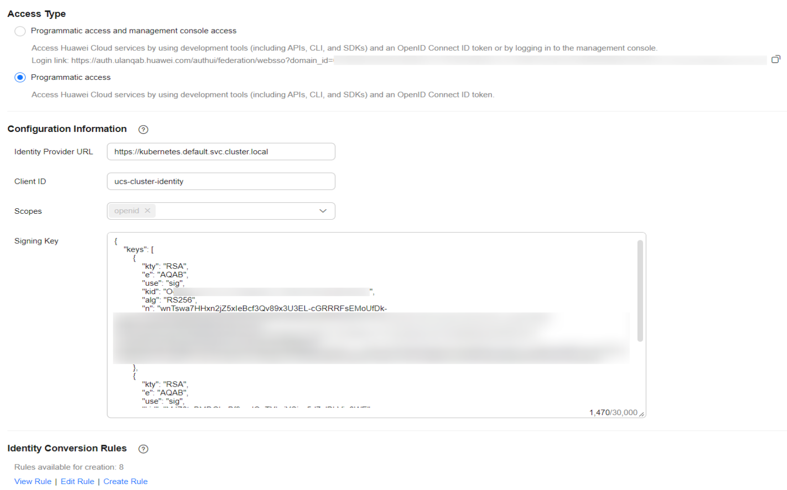
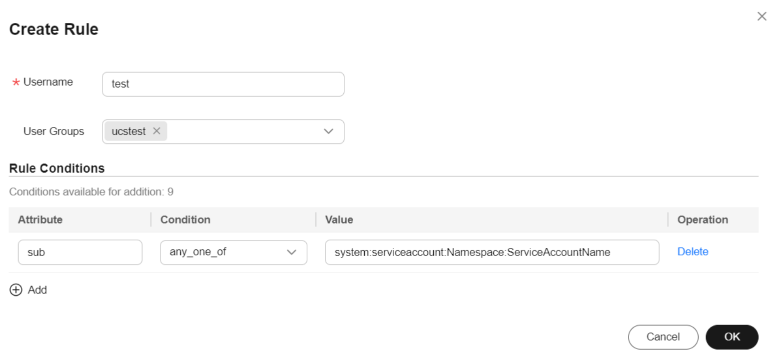


Table 9-18 IdP parameters

Parameter	Description
Access Type	Select **Programmatic access**.
Configuration Information	– **Identity Provider URL**: Enter **https://kubernetes.default.svc.cluster.local**. – **Client ID**: Enter the client ID of log-agent. For details, see **Table 9-17**. – **Signing Key**: Enter the JWKS of the on-premises cluster obtained in **Step 1**.

Parameter	Description
Identity Conversion Rules	– An identity conversion rule maps a ServiceAccount in an on-premises to an IAM user group. – For example, create a ServiceAccount in namespace **default** of the cluster and map it to user group **demo**. If you use the IAM token obtained by the ServiceAccount to access cloud services, you have the permissions of the **demo** user group. – In a mapping rule, the attribute must be **sub**. The value format is **system:serviceaccount:***Namespace*:*ServiceAccountName*. – *ServiceAccountName* and user group permissions are required for the running of the log-agent in an on-premises cluster. For details, see **Table 9-17**.

Figure 9-31 Creating an identity conversion rule



3. Click **OK**.

----End

Installing log-agent in an On-Premises Cluster

Step 1 Log in to the UCS console and choose **Fleets**. Then, click the cluster name to access the cluster console. In the navigation pane on the left, choose **Add-ons**. Locate **Cloud Native Logging** on the right and click **Install**.

Step 2 On the **Install Add-on** page, configure the specifications.

Table 9-19 Add-on specifications

Parameter	Description
Add-on Specifications	The add-on specifications can be of the **Low**, **High**, or **custom-resources** type.

Parameter	Description
Pods	Number of pods that will be created to match the selected add-on specifications. If you select **custom-resources**, you can adjust the number of pods as required.
Containers	The log-agent add-on contains the following containers, whose specifications can be adjusted as required: ● fluent-bit: indicates the log collector, which is installed on each node as a DaemonSet. ● cop-logs: generates and updates configuration files on the collection side. ● log-operator: parses and updates log collection rules. ● otel-collector: forwards logs collected by fluent-bit to LTS in a centralized manner.

Step 3 Configure the parameters in **Parameters**.

Interconnection with AOM: If this option is enabled, Kubernetes events will be collected and reported to AOM. You can configure alarm rules on AOM.

Step 4 Configure the network for reporting add-on instance logs.

- Public network: This option features flexibility, cost-effectiveness, and easy access. It is only available for clusters that can access the public network.
- Direct Connect or VPN: After you connect the on-premises network to the cloud network over Direct Connect or VPN, you can use a VPC endpoint to access CIA over the private network. This option features high speed, low latency, and security. For details, see **Using Direct Connect or VPN to Report Logs of On-Premises Clusters**.

Step 5 Click **Install**.

----End

log-agent Components

Table 9-20 log-agent components

Component	Description	Resource Type
fluent-bit	Lightweight log collector and forwarder deployed on each node to collect logs	DaemonSet
cop-logs	Used to generate soft links for collected files and run in the same pod as fluent-bit	DaemonSet
log-operator	Used to generate internal configuration files	Deployment

Component	Description	Resource Type
otel-collector	Used to collect logs from applications and services and report the logs to LTS	Deployment

Change History

Table 9-21 Release history

Add-on Version	Supported Cluster Version	New Feature
1.4.1	v1.21 v1.22 v1.23 v1.24 v1.25 v1.26 v1.27 v1.28	This is the first official release. It can be installed in the on-premises clusters.

Reporting Custom Events to AOM

The log-agent add-on reports all warning events and some normal events to AOM. You can also set the events to be reported as required.

1. Run the following command on the cluster to modify the event collection settings:

   kubectl edit logconfig -n kube-system default-event-aom

2. Modify the event collection settings as required.

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    # Settings on UCS from which events are collected
    type: event    # Type of logs to be collected. Do not change the value.
    event:
      normalEvents:    # Used to configure normal events
        enable: true    # Whether to enable normal event collection
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events
will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all
events will be collected.
        - NotTriggerScaleUp
      warningEvents:    # Used to configure warning events
        enable: true    # Whether to enable warning event collection
```

```
        includeNames:    # Names of events to be collected. If this parameter is not specified, all events will be collected.
        - NotTriggerScaleUp
        excludeNames:    # Names of events that are not collected. If this parameter is not specified, all events will be collected.
        - NotTriggerScaleUp
  outputDetail:
    type: AOM    # Type of the system that receives the events. Do not change the value.
    AOM:
      events:
      - name: DeleteNodeWithNoServer    # Event name. This parameter is mandatory.
        resourceType: Namespace    # Type of the resource that operations are performed on.
        severity: Major    # Event severity after an event is reported to AOM, which can be Critical, Major, Minor, or Info. The default value is Major.
```

log-agent Events

During log-agent installation and running, the log-operator component reports events. You can determine whether log-agent is installed and determine fault causes based on these events. For details, see **Table 9-22**.

Table 9-22 log-agent events

Event Name	Description
InitLTSFailed	Failed to initialize the log streams in the LTS log group.
WatchAKSKFailed	Failed to listen to the AK/SK.
WatchAKSKSuccessful	AK/SK listened.
RequestLTSFailed	Failed to request the LTS interface.
InitLTSSuccessful	Log streams in the LTS log group initialized.
CreateWebhookConfig-Failed	Failed to create MutatingWebhookConfiguration.
CreateWebhookConfig-Successful	MutatingWebhookConfiguration created.
StartServerSuccessful	Listening enabled.
StartServerFailed	Failed to enable listening.
StartManagerFailed	Failed to enable CRD listening.
InjectAnnotationFailed	Failed to inject annotations.
InjectAnnotationSuc-cessful	Annotations injected.
UpdateLogConfigFailed	Failed to update the logconfig information.
GetConfigListFailed	Failed to obtain the CR list.
GenerateConfigFailed	Failed to generate the fluent-bit and otel settings.

log-agent Metrics

The log-operator, fluent-bit, and otel-collector components of the log-agent add-on have a series of metrics. You can use AOM or Prometheus to monitor these metrics to check the running of the log-agent add-on in a timely manner. For details, see **Monitoring Custom Metrics Using AOM** or **Monitoring Custom Metrics Using Prometheus**. The following lists the metrics:

- log-operator (only for Huawei Cloud clusters)

  Port: 8443

  Address: /metrics

  Protocol: HTTPS

  Table 9-23 Metrics

Metric	Description	Type
log_operator_aksk_latest_update_times	Last update time of the AK/SK	Gauge
log_operator_aksk_update_total	Cumulative count of AK/SK update times	Counter
log_operator_send_request_total	Cumulative count of requests that have been sent	Counter
log_operator_webhook_listen_status	Webhook listening status	Gauge
log_operator_http_request_duration_seconds	HTTP request latency	Histogram
log_operator_http_request_total	Cumulative count of HTTP requests	Counter
log_operator_webhook_request_total	Cumulative count of webhook requests	Counter

- fluent-bit

  Port: 2020

  Address: /api/v1/metrics/prometheus

  Protocol: HTTP

  Table 9-24 Metrics

Metric	Description	Type
fluentbit_filter_add_records_total	Cumulative count of records added by the Fluent Bit filter add-on	Counter

Metric	Description	Type
fluentbit_filter_drop_records_total	Cumulative count of records dropped by the Fluent Bit filter add-on	Counter
fluentbit_input_bytes_total	Number of input bytes	Counter
fluentbit_input_files_closed_total	Cumulative count of files closed by the Fluent Bit input add-on	Counter
fluentbit_input_files_opened_total	Cumulative count of files opened by the Fluent Bit input add-on	Counter
fluentbit_input_files_rotated_total	Cumulative count of files rotated by the Fluent Bit input add-on	Counter
fluentbit_input_records_total	Number of input records	Counter
fluentbit_output_dropped_records_total	Number of dropped records	Counter
fluentbit_output_errors_total	Number of output errors	Counter
fluentbit_output_proc_bytes_total	Number of processed output bytes	Counter
fluentbit_output_proc_records_total	Number of processed output records	Counter
fluentbit_output_retried_records_total	Number of retried records	Counter
fluentbit_output_retries_total	Number of output retries	Counter
fluentbit_uptime	Number of seconds that Fluent Bit has been running	Counter
fluentbit_build_info	Build version information	Gauge

- otel-collector
  Port: 8888
  Address: /metrics
  Protocol: HTTP

Table 9-25 Metrics

Metric	Description	Type
otelcol_exporter_enqueue_failed_log_records	Number of log records failed to be added to the sending queue	Counter
otelcol_exporter_enqueue_failed_metric_points	Number of metric points failed to be added to the sending queue	Counter
otelcol_exporter_enqueue_failed_spans	Number of spans failed to be added to the sending queue	Counter
otelcol_exporter_send_failed_log_records	Number of log records failed to be sent	Counter
otelcol_exporter_sent_log_records	Number of log records that have been sent	Counter
otelcol_process_cpu_seconds	Total CPU user and system time in seconds	Counter
otelcol_process_memory_rss	Total physical memory (resident set size)	Gauge
otelcol_process_runtime_heap_alloc_bytes	Bytes of allocated heap objects	Gauge
otelcol_process_runtime_total_alloc_bytes	Cumulative bytes allocated for heap objects	Counter
otelcol_process_runtime_total_sys_memory_bytes	Total bytes of memory obtained from the OS	Gauge
otelcol_process_uptime	Uptime of the process in seconds	Counter
otelcol_receiver_accepted_log_records	Number of log records received and processed by the OpenTelemetry receiver	Counter
otelcol_receiver_refused_log_records	Number of log records rejected by the OpenTelemetry receiver	Counter

9.2.8 Using Direct Connect or VPN to Report Logs of On-Premises Clusters

After you connect the on-premises network to the cloud network over Direct Connect or VPN, you can use a VPC endpoint to access CIA over the private network. This approach features high speed, low latency, and security.

LTS VPC Endpoint Authorization

Step 1 On the top menu bar, choose **Service Tickets** > **Create Service Ticket**.

Step 2 Enter LTS in the **My Service/Product** text box and click **Search**.

Step 3 Select **Other** and click **Create Now**.

Step 4 Describe the problem, set **Contact Information**, and click **Submit**.

NOTE

Problem description example: LTS VPC endpoint authorization, *{Account ID}*. For example, enable the LTS VPC endpoint permission.

----End

Direct Connect/VPN Access

Step 1 Submit a service ticket to enable the VPC endpoint of LTS. For details, see **LTS VPC Endpoint Authorization**.

Step 2 On the on-premises cluster details page, edit the settings of log-agent.

- If log-agent is not installed, you can click **Enable** on the **Logging** page.
- If log-agent has been installed, you can edit its settings on the add-on page.

Step 3 Select the VPC endpoint. If no VPC endpoint is available, create one. If you submit a service ticket again, the VPC endpoint of LTS needs to be approved by LTS personnel. For details, see **LTS VPC Endpoint Authorization**.

NOTICE

The VPC endpoint of LTS and the node where an on-premises cluster resides must be in the same VPC. If they are in different VPCs, you need to create a VPC peering connection so that the VPCs communicate with each other.

Step 4 Click **OK**.

----End

10 Container Migration

10.1 Overview

The container migration service of Huawei Cloud UCS provides you with a reliable, secure, flexible, and efficient migration solution. UCS allows you to migrate cloud native applications from Kubernetes clusters in your on-premises data center or of another cloud provider to the Kubernetes clusters managed by Huawei Cloud UCS. In this way, you can implement unified O&M for less expensive and more efficient management.

Migrating applications from one environment to another is a challenging task, so you need to plan and prepare carefully. The container migration service of UCS guides you throughout the following four phases of migration:

1. Cluster evaluation: Evaluate the status of the source cluster to determine the type of the destination cluster.
2. Data migration: Migrate images and data related to dependent services to the cloud.
3. Application backup: Back up applications in the source cluster.
4. Application migration: Migrate applications from the source cluster to the destination cluster by restoring backup data.

The guide provided by UCS throughout the entire migration process will help you smoothly migrate applications from one environment to another.

Advantages

The container migration service of UCS has the following advantages:

- **Ease of use**

  Tool-based migration has been implemented throughout the cluster evaluation, image migration, application backup, and application migration phases. These tools are installation-free, easy to use, lightweight, and flexible.

- **Versioning**

  Resources can be migrated from clusters of Kubernetes 1.19 or later to UCS.

- **No dependency**
  The migration tools do not require any external dependency and can run independently.
- **Multi-architecture**
  The migration tools can run on Linux (x86 and Arm) and Windows.
- **Multi-scenario**
  Cluster migration in multiple scenarios is supported to meet different migration requirements. For details, see **Table 10-1**.

Table 10-1 Migration scenarios

Scenario	Description
Migration from clusters in an on-premises IDC to the cloud	Applications can be migrated from Kubernetes clusters in your on-premises data center to the Kubernetes clusters managed by Huawei Cloud UCS to implement cloud deployment and O&M of applications.
Migration from clusters on a third-party cloud	Applications can be migrated from Kubernetes clusters of another cloud provider to the Kubernetes clusters managed by Huawei Cloud UCS to implement cross-cloud migration and unified management.
Migration across Huawei Cloud UCS clusters in different regions	Applications can be migrated between Kubernetes clusters managed by Huawei Cloud UCS from one geographic region to another to meet data compliance, latency, and availability requirements.
Migration across Huawei Cloud UCS clusters in the same region	Applications can be migrated between Kubernetes clusters managed by Huawei Cloud UCS in the same geographical region to meet management requirements such as better resource utilization and application upgrade.

- **No downtime**
  No downtime occurs during the migration so there is zero impact on the source cluster.

10.2 Preparations

Hardware Resources

Before the migration, ensure that you have prepared a server with kubectl installed to enable networking between the source cluster and the destination cluster. The server must have at least 5 GB local disk space and at least 8 GB memory for the migration tool to work properly and store related data, such as collected data of the source cluster and recommendation data of the destination cluster.

The migration tool can run on Linux (x86 and Arm) and Windows, meaning that the server can run on these OS types.

Tool Package

Tool-based migration has been implemented throughout the cluster evaluation, image migration, application backup, and application migration phases. You need to download these tools in advance and upload them to the preceding server.

NOTE

Before using the following tools in Linux OSs, run the **chmod u+x *tool name*** command (for example, **chmod u+x kspider-linux-amd64**) to grant the execute permission.

Table 10-2 Preparations

<table>
<tr><th>Tool</th><th>Description</th><th>Download Link</th><th>Remarks</th></tr>
<tr><td>kspider</td><td>kspider is a tool used to collect information about the source cluster. It provides cluster-related data such as the Kubernetes version, scale, workload quantity, storage, and in-use images. The data helps you understand the current status of the cluster and evaluate migration risks, and select a proper destination cluster version and scale.</td><td>https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/kspider-23.3.0.0317182614.tar.gz</td><td rowspan="2">These tools can run on Linux (x86 and Arm) and Windows. After the tool package is decompressed, two binary files and one application are obtained, which are applicable to Linux and Windows, respectively.
kspider includes:
• kspider-linux-amd64
• kspider-linux-arm64
• kspider-windows-amd64.exe
image-migrator includes:
• image-migrator-linux-amd64
• image-migrator-linux-arm64
• image-migrator-windows-amd64.exe
k8clone includes:
• k8clone-linux-amd64
• k8clone-linux-arm64
• k8clone-windows-amd64.exe</td></tr>
<tr><td>image-migrator</td><td>image-migrator is an image migration tool that can automatically migrate images from the Docker image repository built on Docker Registry v2 to SWR, or from the image repository on a third-party cloud to SWR.</td><td>https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/image-migrator-23.3.0.0323215042.tar.gz</td></tr>
</table>

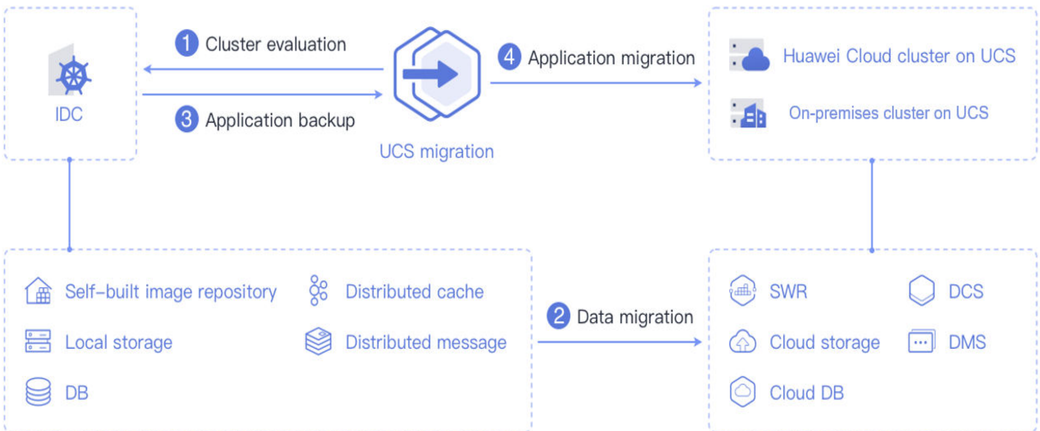
Tool	Description	Download Link	Remarks
k8clone	k8clone is an easy-to-use Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster.	**https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/k8clone-22.3.0.0329155427.tar.gz**	

10.3 Migration from Clusters in an On-premises Data Center to the Cloud

10.3.1 Migration Process

The container migration service of UCS allows you to migrate applications from Kubernetes clusters in an on-premises data center to the Huawei Cloud clusters or on-premises clusters of UCS for cloud deployment and O&M of applications. **Figure 10-1** shows the migration process.

Figure 10-1 Migration process



The process is as follows:

Step 1 **Cluster evaluation**

In this phase, you will evaluate the status of the source cluster to determine the type of the destination cluster. UCS kspider can automatically collect information about the source cluster, including the Kubernetes version, cluster scale, workload, and storage, and provide you with information about the recommended destination cluster. For details, see **Cluster Evaluation**.

Step 2 **Data migration**

In this phase, you will migrate images and data related to dependent services to the cloud. UCS image-migrator is an automatic image migration tool. It can migrate images from the Docker image repository built on Docker Registry v2 to SWR. To migrate data of dependent services, you can use other Huawei Cloud products together with image-migrator. For details, see **Image Migration** and **Dependent Service Migration**.

Step 3 **Application backup**

In this phase, you will back up applications in the on-premises IDC cluster. UCS k8clone can automatically collect Kubernetes metadata and save it as a compressed package to the local host to back up applications in the cluster. For details, see **Application Backup**.

Step 4 **Application migration**

In this phase, you will restore backup data to migrate applications from the cluster in an on-premises data center to a Huawei Cloud clusteror on-premises cluster of UCS. For details, see **Application Migration**.

----End

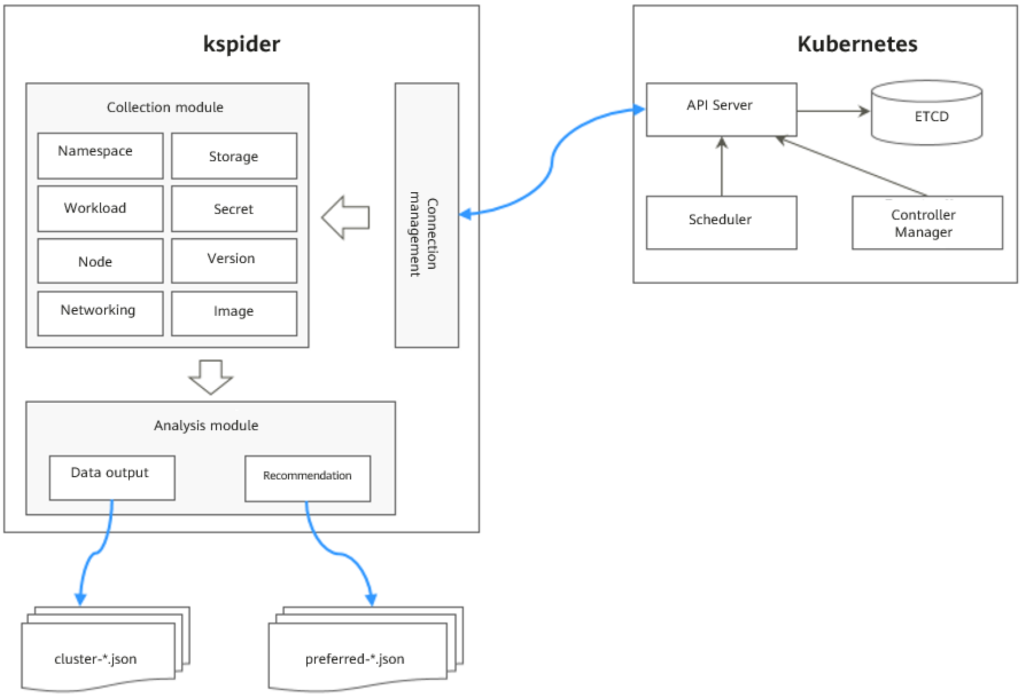
10.3.2 Cluster Evaluation

Migrating applications from one environment to another is a challenging task, so you need to plan and prepare carefully. kspider is a tool used to collect information about the source cluster. It provides cluster-related data such as the Kubernetes version, scale, workload quantity, storage, and in-use images. The data helps you understand the current status of the cluster and evaluate migration risks, and select a proper destination cluster version and scale.

How kspider Works

Figure 10-2 shows the architecture of kspider, which consists of three modules: collection, connection management, and analysis. The collection module can collect data of the source cluster, including namespaces, workloads, nodes, and networks. The connection management module establishes connections with the API Server of the source cluster. The analysis module aims to output the collected data of the source cluster (generating the **cluster-*.json** file) and provide the recommendation information of the destination cluster (generating the **preferred-*.json** file) after evaluation.

Figure 10-2 kspider architecture



Usage of kspider

NOTE

kspider can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **kspider-linux-amd64** in the following command with **kspider-linux-arm64** or **kspider-windows-amd64.exe**.

Prepare a server, upload kspider to the server, and decompress the tool package. For details, see **Preparations**. Run **./kspider-linux-amd64 -h** in the directory where kspider is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the collected namespace. By default, system namespaces such as **kube-system**, **kube-public**, and **kube-node-lease** are excluded.
- **-q, --quiet**: indicates static exit.
- **-s, --serial**: specifies the unique sequence number of the output aggregation file (**cluster-{serial}.json**) and recommendation file (**preferred-{serial}.json**).

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]
```

```
Aliases:
  kspider, kspider

Flags:
  -h, --help                help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet               command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

Step 1: Collect Data from the Source Cluster

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

Step 2 Use the default parameter settings to collect data of all namespaces in the cluster. Run the **./kspider-linux-amd64** command.

Command output:

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
    Name      CPU     Memory      IP      Arch      OS      Kernel      MachineID
    10.1.18.64      4      8008284Ki      [10.1.18.64 10.1.18.64]      amd64      linux
3.10.0-1127.19.1.el7.x86_64      ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
    10.1.19.13      4      8008284Ki      [10.1.19.13 10.1.19.13]      amd64      linux
3.10.0-1127.19.1.el7.x86_64      2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
    Name      Namespace      NodeAffinity
    minio      default      false
    minio      minio      false
There are 3 Deployments:
    Name      Namespace      NodeAffinity
    rctest      default      true
    flink-operator-controller-manager      flink-operator-system      false
    rctest      minio      false
There are 1 DaemonSets:
    Name      Namespace      NodeAffinity
    ds-nginx      minio      false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
    Namespace/Name      Pods
    default/pvc-data-minio-0      default/minio-0
    minio/obs-testing      minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
    minio/pvc-data-minio-0      minio/minio-0
There are 5 PersistentVolumes:
    Name      Namespace      pvcName      scName      size      key
    pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8      minio-pv      obs-testing      csi-obs      1Gi      pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
    pvc-c7c768aa-373a-4c52-abea-e8b486d23b47      minio-pv      pvc-data-minio-0      csi-disk-sata      10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
    pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05      minio      obs-testing      csi-obs      1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
    pvc-9fd92c99-805a-4e65-9f22-e238130983c8      default      pvc-data-minio-0      csi-disk      10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
    pvc-a22fd877-f98d-4c3d-a04e-191d79883f97      minio      pvc-data-minio-0      csi-disk-sata      10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
    Name      Namespace      ServiceType
    headless-lxprus      default      ClusterIP
    kubernetes      default      ClusterIP
```

```
    minio      default      NodePort
    flink-operator-controller-manager-metrics-service      flink-operator-system      ClusterIP
    flink-operator-webhook-service      flink-operator-system      ClusterIP
    headless-lxprus      minio      ClusterIP
    minio      minio      NodePort
There are 0 Ingresses:
There are 6 Images:
    Name
    gcr.io/flink-operator/flink-operator:v1beta1-6
    flink:1.8.2
    swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
    nginx:stable-alpine-perl
    swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
    gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
    SecretType
    cfe/secure-opaque
    helm.sh/release.v1
```

After the kspider command is executed, the following files are generated in the current directory:

- **cluster-*.json**: This file contains data collected from the source cluster and applications. The data can be used to analyze and plan the migration.
- **preferred-*.json**: This file contains information about the recommended destination cluster. A preliminary evaluation is performed for the source cluster according to its scale and node specifications. The file provides suggestions on the version and scale of the destination cluster.

Step 3 View the data collected from the source cluster and applications.

You can use a text editor or JSON viewer to open the **cluster-*.json** file to view the data. Replace the asterisk (*) in the file name with the actual timestamp or serial number to find and open the correct file.

Description of the **cluster-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Namespaces: number of namespaces. The value is a string.
  Pods: total number of pods. The value is an integer.
  Nodes: node information. The IP address is used as the key to display node information.
    IP addresses
      CPU: CPU. The value is a string.
      Arch: CPU architecture. The value is a string.
      Memory: memory. The value is a string.
      HugePages1Gi: 1 GB hugepage memory. The value is a string.
      HugePages2Mi: 2 MB hugepage memory. The value is a string.
      OS: node OS. The value is a string.
      KernelVersion: OS kernel version. The value is a string.
      RuntimeVersion: running status and version of the node container. The value is a string.
      InternalIP: internal IP address. The value is a string.
      ExternalIP: external IP address. The value is a string.
      MachineID: node ID. The value is a string. Ensure that the CCE ID is the same as the ECS ID.
  Workloads: workload
    Deployment: workload type. The value can be Deployment, StatefulSet, DaemonSet, CronJob, Job, or
LonePod.
      default: namespace name
        Count: quantity. The value is an integer.
        Items: details. The value is an array.
          Name: workload name. The value is a string.
          Namespace: namespace name. The value is a string.
          NodeAffinity: node affinity. The value is of the Boolean type.
          Replicas: number of replicas. The value is an integer.
  Storage: storage
    PersistentVolumes: persistent volume
      pv-name: The PV name is used as the key.
```

```
        VolumeID: volume ID. The value is a string.
        Namespace: namespace. The value is a string.
        PvcName: name of the bound PVC. The value is a string.
        ScName: storage class name. The value is a string.
        Size: size of the space to request. The value is a string.
        Pods: name of the pod that uses the PV. The value is a string.
        NodeIP: IP address of the node where the pod is located. The value is a string.
        VolumePath: path of the node to which the pod is mounted. The value is a string.
    OtherVolumes: volumes of other types
      Type: AzureFile, AzureDisk, GCEPersistentDisk, AWSElasticBlockStore, Cinder, Glusterfs, NFS, CephFS,
FlexVolume, FlexVolume, DownwardAPI
        The volume ID, volume name, and volume shared path are keys.
        Pods: name of the pod. The value is a string.
          NodeIP: IP address of the node where the pod is located. The value is a string.
          Information that uniquely identifies a volume, such as the volume ID, volume name, and volume
shared path. The value is a string.
  Networks: network
    LoadBalancer: load balancing type
      service: network type, which can be Service or Ingress.
        Name: name. The value is a string.
        Namespace: namespace name. The value is a string.
        Type: type. The value is a string.
  ExtraSecrets: extended secret type
    Secret type. The value is a string.
  Images: image
    Image repo. The value is a string.
}
```

Example:

```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
      "ExternalIP": "100.85.xxx.xxx",
      "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
    }
  },
  "Workloads": {
    "CronJob": {},
    "DaemonSet": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-prometheus-node-exporter",
```

```
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 3
        }
      ]
    }
  },
  "Deployment": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-cost-analyzer",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    },
    "kubecost": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-kube-state-metrics",
          "Namespace": "kubecost",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  },
  "Job": {},
  "LonePod": {},
  "StatefulSet": {
    "minio-all": {
      "Count": 1,
      "Items": [
        {
          "Name": "minio",
          "Namespace": "minio-all",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  }
},
"Storage": {
  "PersistentVolumes": {
    "demo": {
      "VolumeID": "demo",
      "Namespace": "fluid-demo-test",
      "PvcName": "demo",
      "ScName": "fluid",
      "Size": "100Gi",
      "Pods": "",
      "NodeIP": "",
      "VolumePath": ""
    },
    "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
      "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
      "Namespace": "minio-all",
      "PvcName": "pvc-data-minio-0",
      "ScName": "csi-disk",
      "Size": "10Gi",
      "Pods": "minio-all/minio-0",
      "NodeIP": "10.1.23.159",
      "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
```

```
      }
    },
    "OtherVolumes": {}
  },
  "Networks": {
    "LoadBalancer": {}
  },
  "ExtraSecrets": [
    "cfe/secure-opaque",
    "helm.sh/release.v1"
  ],
  "Images": [
    "nginx:stable-alpine-perl",
    "ghcr.io/koordinator-sh/koord-manager:0.6.2",
    "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
    "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
    "gcr.io/kubecost1/cost-model:prod-1.91.0",
    "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```

----End

Step 2: Evaluate the Destination Cluster

After the kspider command is executed, in addition to the **cluster-*.json** file, the **preferred-*.json** file is also generated in the current directory. After performing preliminary evaluation for the source cluster according to its scale and node specifications, the file provides the recommended version and scale of the destination cluster. This helps you better plan and prepare for the migration.

Description of the **preferred-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Scale: cluster scale. The value is a string.
  Nodes: node information
    CPU: CPU. The value is a string.
    Memory: memory. The value is a string.
    Arch: CPU architecture. The value is a string.
    KernelVersion: OS kernel version. The value is a string.
    ProxyMode: cluster proxy mode. The value is a string.
  ELB: whether the ELB service is a dependent service. The value is of the Boolean type.
}
```

Evaluation rules for each field in the preceding file:

Table 10-3 Evaluation rules

Field	Evaluation Rule
K8sVersion	If the version is earlier than 1.21, the main release version of the UCS cluster (for example, 1.21, which changes over time) is recommended. If the version is later than the main release version, the latest version of the UCS cluster is recommended.

Field	Evaluation Rule
Scale	< 25 nodes in the source cluster: Destination cluster of 50 nodes is recommended. 25 ≤ Nodes in the source cluster < 100: Destination cluster of 200 nodes is recommended. 100 ≤ Nodes in the source cluster < 500: Destination cluster of 1000 nodes is recommended. Nodes in the source cluster ≥ 500: Destination cluster of 2000 nodes is recommended.
CPU/Memory	Statistics about the specification of the largest quantity are collected.
Arch	Statistics about the specification of the largest quantity are collected.
KernelVersion	Statistics about the specification of the largest quantity are collected.
ProxyMode	Configure this parameter according to the cluster scale. For a cluster with more than 1000 nodes, **ipvs** is recommended. For a cluster with fewer than 1000 nodes, **iptables** is recommended.
ELB	Check whether the source cluster has a load balancing Service.

Example:

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

CAUTION

The evaluation result is for reference only. You need to determine the version and scale of the destination cluster.

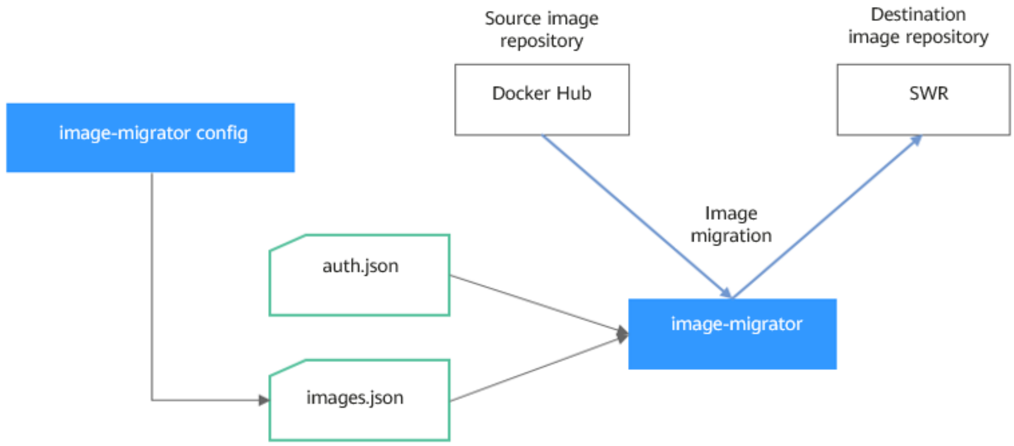
10.3.3 Image Migration

To ensure that container images can be properly pulled after cluster migration and improve container deployment efficiency, you are advised to migrate self-built image repositories to Huawei Cloud SWR. The Huawei Cloud clusters and on-premises clusters of UCS work with SWR to provide a pipeline for automated container delivery. Images are pulled in parallel, which greatly improves container delivery efficiency.

image-migrator is an image migration tool that can automatically migrate images from the Docker image repository built on Docker Registry v2 to SWR.

How image-migrator Works

Figure 10-3 How image-migrator works



When using image-migrator to migrate images to SWR, you need to prepare two files. One is the image repository access permission file **auth.json**. The two objects in the file are the accounts and passwords of the source and destination image repositories (registries). The other is the image list file **images.json**, which consists of multiple image synchronization rules. Each rule contains a source image repository (key) and a destination image repository (value). Place these two files in the directory where image-migrator is located and run a simple command to migrate the image. The two files are described as follows:

- **auth.json**

  auth.json is the image repository access permission file. Each object is the username and password of a registry. Generally, the source image repository must have the permissions for pulling images and accessing tags, and the destination image repository must have the permissions for pushing images and creating repositories. If you access the image repository anonymously, you do not need to enter the username and password. Structure of the **auth.json** file:

```
{
    "Source image repository address": { },
    "Destination image repository address": {
        "username": "xxxxxx",
        "password": "***************",
        "insecure": true
    }
}
```

  To be more specific:

  – The values of **Source image repository address** and **Destination image repository address** can be in the *registry* or *registry/namespace* format, which must correspond to the *registry* or *registry/namespace* format in

images.json. The matched URL in images uses the corresponding username and password for image synchronization. The *registry/namespace* format is preferred.

If the destination image repository address is in the *registry* format, you can obtain it from the SWR console. On the **Dashboard** page, click **Generate Login Command** in the upper right corner. The domain name at the end of the login command is the SWR image repository address, for example, swr.cn-north-4.myhuaweicloud.com. Note that the address varies depending on the region. Switch to the corresponding region to obtain the address. If the value is in the *registry/namespace* format, replace *namespace* with the organization name of SWR.

- **username**: (Optional) username. You can set it to a specific value or use a string of the ${env} or $env type to reference an environment variable.
- **password**: (Optional) password. You can set it to a specific value or use a string of the ${env} or $env type to reference an environment variable.
- **insecure**: (Optional) whether *registry* is an HTTP service. If yes, the value of **insecure** is **true**. The default value is **false**.

NOTE

The username of the destination SWR image repository is in the following format: *Regional project name@AK*. The password is the encrypted login key of the AK and SK. For details, see **Obtaining a Long-Term Valid Login Command**.

Example:

```
{
   "quay.io/coreos": { },
   "swr.cn-north-4.myhuaweicloud.com": {
      "username": "cn-north-4@RVHVMX******",
      "password": "***************",
      "insecure": true
   }
}
```

- **images.json**

  This file is essentially a list of images to migrate and consists of multiple image synchronization rules. Each rule contains a source image repository (key) and a destination image repository (value). The specific requirements are as follows:

  a. The largest unit that can be synchronized using one rule is repository. The entire namespace or registry cannot be synchronized using one rule.

  b. The formats of the source and destination repositories are similar to those of the image URL used by the **docker pull/push** command (**registry/namespace/repository:tag**).

  c. Both the source and destination repositories (if the destination repository is not an empty string) contain at least *registry/namespace/repository*.

  d. The source repository field cannot be empty. To synchronize data from a source repository to multiple destination repositories, you need to configure multiple rules.

  e. The destination repository name can be different from the source repository name. In this case, the synchronization function is similar to docker pull + docker tag + docker push.

f. If the source repository field does not contain tags, all tags of the repository have been synchronized to the destination repository. In this case, the destination repository cannot contain tags.

g. If the source repository field contains tags, only one tag in the source repository has been synchronized to the destination repository. If the destination repository does not contain tags, the source tag is used by default.

h. If the destination repository is an empty string, the source image will be synchronized to the default namespace of the default registry. The repository and tag are the same as those of the source repository. The default registry and namespace can be configured using command line parameters and environment variables.

Example:

```
{
    "quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
    "quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
    "quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
}
```

You can use a config subcommand of the image-migrator tool to automatically obtain the image that is being used by the workload in the cluster. For details, see **Usage of image-migrator config**. After obtaining the **images.json** file, you can modify, add, or delete its content as required.

Usage of image-migrator

NOTE

image-migrator can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **image-migrator-linux-amd64** in the following command with **image-migrator-linux-arm64** or **image-migrator-windows-amd64.exe**.

Run **./image-migrator-linux-amd64 -h** in the directory where image-migrator is located to learn about its usage.

- **--auth**: specifies the path of **auth.json**. By default, **auth.json** is stored in the directory where image-migrator is located.
- **--images**: specifies the path of **images.json**. By default, **images.json** is stored in the directory where image-migrator is located.
- **--log**: specifies the path for storing logs generated by image-migrator. The default value is **image-migrator.log** in the current directory of image-migrator.
- **--namespace**: specifies the default namespace of the destination repository. That is, if the namespace of the destination repository is not specified in **images.json**, you can specify it when running the migration command.
- **--registry**: specifies the default registry of the destination repository. That is, if the registry of the destination repository is not specified in **images.json**, you can specify it when running the migration command.
- **--retries**: specifies the number of retry times when the migration fails. The default value is **3**.
- **--workers**: specifies the number of concurrent workers for image migration. The default value is **7**.

```
$ ./image-migrator-linux-amd64 -h
A Fast and Flexible docker registry image images tool implement by Go.

Usage:
  image-migrator [flags]

Aliases:
  image-migrator, image-migrator

Flags:
      --auth string       auth file path. This flag need to be pair used with --images. (default "./auth.json")
  -h, --help              help for image-migrator
      --images string     images file path. This flag need to be pair used with --auth (default "./images.json")
      --log string        log file path (default "./image-migrator.log")
      --namespace string   default target namespace when target namespace is not given in the images 
config file, can also be set with DEFAULT_NAMESPACE environment value
      --registry string    default target registry url when target registry is not given in the images config file, 
can also be set with DEFAULT_REGISTRY environment value
  -r, --retries int       times to retry failed tasks (default 3)
  -w, --workers int       numbers of working goroutines (default 7)

$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \
--registry=swr.cn-north-4.myhuaweicloud.com --retries=2
$ ./image-migrator
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

Example:

./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

The preceding command is used to migrate the images in the **images.json** file to the image repository **swr.cn-north-4.myhuaweicloud.com/test**. If the migration fails, you can retry twice. A maximum of five images can be migrated at a time.

Usage of image-migrator config

The config subcommand of image-migrator can be used to obtain images used in cluster applications and generate the **images.json** file in the directory where the tool is located. You can run **./image-migrator-linux-amd64 config -h** to learn how to use the config subcommand.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the namespace of the image to be obtained. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that images of all namespaces are obtained.
- **-t, --repo**: specifies the destination repository address (*registry*/*namespace*).

```
$ ./image-migrator-linux-amd64 config -h
generate images.json

Usage:
    image-migrator config [flags]

Flags:
    -h, --help              help for config
```

```
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "/
root/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -t, --repo string         target repo,such as swr.cn-north-4.myhuaweicloud.com/test
```

Examples:

- Specify a namespace:
  ./image-migrator-linux-amd64 config -n default -t swr.cn-north-4.myhuaweicloud.com/test
- Specify multiple namespaces:
  ./image-migrator-linux-amd64 config -n default,kube-system -t swr.cn-north-4.myhuaweicloud.com/test
- If no namespace is specified, images of all namespaces are obtained:
  ./image-migrator-linux-amd64 config -t swr.cn-north-4.myhuaweicloud.com/test

Procedure

Step 1 Prepare the image repository access permission file **auth.json**.

Create an **auth.json** file and modify it based on the format. If the repository is accessed anonymously, you do not need to enter information such as the username and password. Place the file in the directory where image-migrator is located.

Example:

```
{
   "quay.io/coreos": { },
   "swr.cn-north-4.myhuaweicloud.com": {
      "username": "cn-north-4@RVHVMX******",
      "password": "***************",
      "insecure": true
   }
}
```

For details about the parameters, see the **auth.json file**.

Step 2 Prepare the image list file **images.json**.

1. Connect to the source cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.
2. Run the config subcommand for image migration to generate the **images.json** file.

   You can refer to the methods and examples in **Usage of image-migrator config** to obtain the image used in the source cluster application without specifying the namespace, or by specifying one or more namespaces.
3. Modify the **images.json** file as required. Ensure that the file meets the eight requirements described in **images.json file**.

Step 3 Migrate images.

You can run the default **./image-migrator-linux-amd64** command to migrate images or configure image-migrator parameters as required.

For example, run the following command:

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

Example:

```
$ ./image-migrator-linux-amd64
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

Step 4 View the result.

After the preceding command is executed, information similar to the following is displayed:

```
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

The preceding information indicates that 38 images have been migrated to the SWR repository.

----End

10.3.4 Dependent Service Migration

Migrate data of services on which the cluster depends, such as local storage, database, distributed cache, and distributed message. If your cluster does not involve the data of these services or the data does not need to be migrated to the cloud, skip this section.

Storage Migration

- If your cluster uses local storage, you can use Huawei Cloud **Data Express Service (DES)** to migrate data to the cloud. DES provides you with physical devices to migrate hundreds of terabytes of data to Huawei Cloud inexpensively and much faster than would be possible over a network connection.
- If your cluster has connected to an object storage service and needs to be migrated to the cloud, Huawei Cloud **Object Storage Migration Service (OMS)** can help you migrate data to Huawei Cloud OBS.
- If your cluster uses SFS for storage, you can use Huawei Cloud **Scalable File Service (SFS)** to migrate data to the cloud.

Database Migration

If your database is not containerized and needs to be migrated to the cloud, Huawei Cloud **Data Replication Service (DRS)** is an ideal option. DRS provides multiple functions, including real-time migration, backup migration, real-time synchronization, data subscription, and real-time DR.

Migrating Other Data

- Big data migration: **Cloud Data Migration (CDM)** on Huawei Cloud
- Kafka service migration: **Migrating Kafka Services** using Distributed Message Service (DMS) on Huawei Cloud for Kafka
- Redis service migration: **Data Migration Guide** of Distributed Cache Service (DCS) on Huawei Cloud

10.3.5 Application Backup

Application migration from clusters in an on-premises IDC consists of two steps: application backup and application migration. That is, applications in the clusters in an on-premises IDC are backed up and then migrated to the destination cluster through data restoration.

k8clone is a simple Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster (Huawei Cloud cluster or on-premises cluster of UCS). In this way, applications can be migrated from clusters in an on-premises data center to the cloud.

NOTICE

Back up data during off-peak hours.

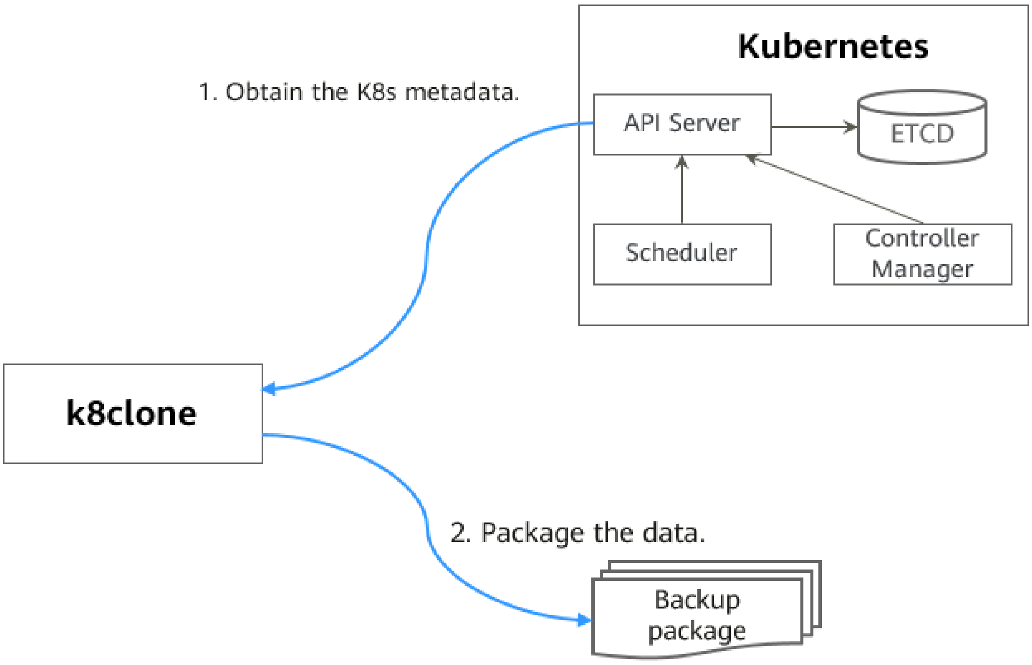
Prerequisites

Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.

How k8clone Backs Up Data

Data backup process:

Figure 10-4 Data backup process



k8clone Usage for Backup

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 backup -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-n, --namespace**: backs up cloud native applications of a specified namespace. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that the entire cluster is backed up.
- **-e, --exclude-namespaces**: excludes the backup of objects of a specified namespace. This parameter cannot be used together with **--namespace**.
- **-x, --exclude-kind**: excludes the backup of a specified resource type.
- **-i, --include-kind**: specifies the backup of a resource type.
- **-y, --exclude-object**: excludes the backup of a specified resource object.
- **-z, --include-object**: specifies the backup of a resource object.
- **-w, --exclude-having-owner-ref**: excludes the backup of resource objects with ownerReferences. The default value is **false**. The equal sign (=) must be contained in the parameter transfer process, for example, -w=true.
- **-d, --local-dir**: path for storing backup data. The default value is the **k8clone-dump** folder in the current directory.

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
  k8clone backup [flags]

Flags:
  -s, --api-server string           Kubernetes api-server url
  -q, --context string              Kubernetes configuration context
-w, --exclude-having-owner-ref      Exclude all objects having an Owner Reference. The following form is
not permitted for boolean flags such as '-w false', please use '-w=false'
  -x, --exclude-kind strings        Resource kind to exclude. Eg. 'deployment'
  -i, --include-kind strings        Ressource kind to include. Eg. 'deployment'
  -e, --exclude-namespaces strings  Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all
namespaces and then filters them. Don't use it with the namespace flag.
  -y, --exclude-object strings      Object to exclude. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -z, --include-object strings      Object to include. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -h, --help                        help for backup
  -k, --kubeconfig string           The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
```

```
-d, --local-dir string          Where to dump yaml files (default "./k8clone-dump")
-n, --namespace string          Only dump objects from this namespace
```

Examples:

- Backs up objects of the entire cluster. The default path is the **k8clone-dump** folder in the current directory.

  ./k8clone-linux-amd64 backup

- Backs up objects of the entire cluster and specifies the path for storing backup data.

  ./k8clone-linux-amd64 backup -d ./xxxx

- Backs up objects of a specified namespace.

  ./k8clone-linux-amd64 backup -n default

- Excludes the backup of objects of a specified namespace.

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease

- Excludes the backup of specified resource types.

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice

- Specifies the backup of resource types.

  ./k8clone-linux-amd64 backup -x rolebinding

- Excludes the backup of specified resource objects.

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns

- Specifies the backup of resource objects.

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns

- Excludes the backup of resource objects with ownerReferences.

  ./k8clone-linux-amd64 backup -w=true

Procedure

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

Step 2 Go to the directory where k8clone is located and run the backup command to back up data to a local directory and compress the data into a package.

The examples in **k8clone Usage for Backup** provide several common backup methods. You can select a method as required or customize one.

----End

10.3.6 Application Migration

Application migration from clusters in an on-premises IDC consists of two steps: application backup and application migration. That is, applications in the clusters in an on-premises IDC are backed up and then migrated to the destination cluster through data restoration.

k8clone is a simple Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster (Huawei Cloud cluster or on-premises cluster of UCS). In this way, applications can be migrated from clusters in an on-premises data center to the cloud.

Constraints

Currently, applications in a cluster of a later version cannot be migrated to a cluster of an earlier version.

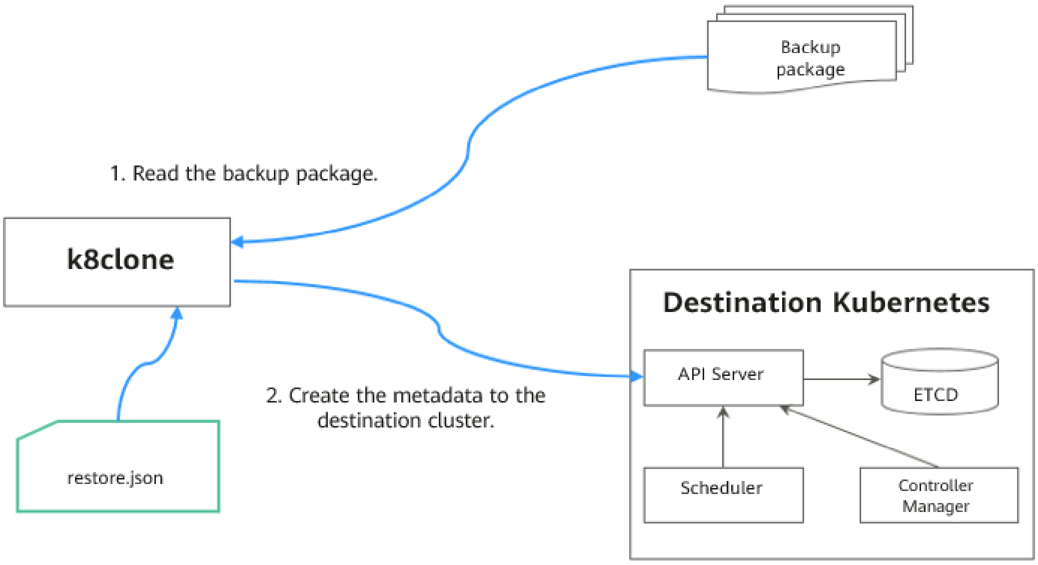
Prerequisites

- Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.
- Ensure that the metadata backup in the source cluster has been downloaded to the server where k8clone is executed.

How k8clone Restores Data

Data restoration process:

Figure 10-5 Data restoration process



Before the restoration, prepare a data restoration configuration file **restore.json** to automatically change the storage class names of PVC and StatefulSet and the repository address of the image used by the workload during application restoration.

The file content is as follows:

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  // The StorageClassName field of PVC and StatefulSet can be changed.
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```


- **StorageClass**: The storage class names of PVC and VolumeClaimTemplates can be automatically changed based on settings.
- **ImageRepo**: The repository address of the image used by the workload can be changed. The workload can be Deployment (including initContainer), StatefulSet, Orphaned Pod, Job, CronJob, Replica Set, Replication Controller, and DaemonSet.

k8clone Usage for Restoration

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 restore -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-f, --restore-conf**: path of **restore.json**. The default value is the directory where k8clone is located.
- **-d, --local-dir**: path for storing backup data. The default value is the directory where k8clone is located.

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
  k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

Procedure

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

Step 2 Prepare the data restoration configuration file **restore.json**.

Create a **restore.json** file, modify it based on the format, and place it in the directory where k8clone is located.

Example:

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

Step 3 Go to the directory where k8clone is located and run the restoration command to restore the backup data to the destination cluster.

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

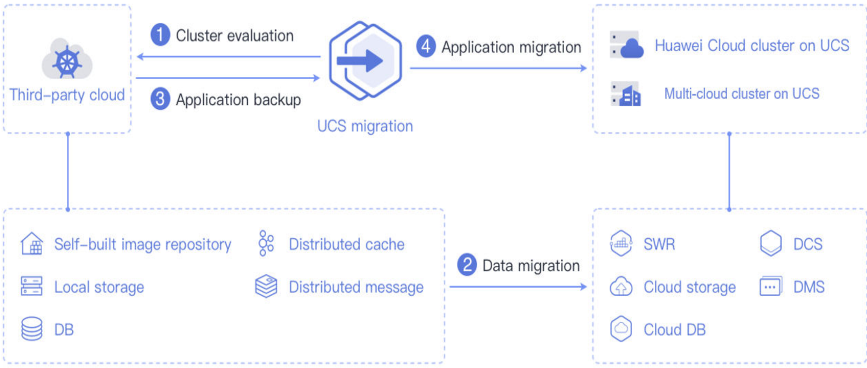
----End

10.4 Migration from Clusters on a Third-party Cloud

10.4.1 Migration Process

The container migration service of UCS allows you to migrate applications from the Kubernetes cluster on a third-party cloud to a Huawei Cloud cluster or multi-cloud cluster of UCS for cross-cloud migration and unified management.

Figure 10-6 Migration process



The process is as follows:

Step 1 **Cluster evaluation**

In this phase, you will evaluate the status of the source cluster to determine the type of the destination cluster. UCS kspider can automatically collect information about the source cluster, including the Kubernetes version, cluster scale, workload, and storage, and provide you with information about the recommended destination cluster. For details, see **Cluster Evaluation**.

Step 2 **Data migration**

In this phase, you will migrate images and data related to dependent services to the cloud. UCS image-migrator is an automatic image migration tool. It can migrate images from the image repository on a third-party cloud to SWR. To migrate data of dependent services, you can use other Huawei Cloud products together with image-migrator.

Step 3 **Application backup**

In this phase, you will back up applications in the cluster on a third-party cloud. UCS k8clone can automatically collect Kubernetes metadata and save it as a compressed package to the local host to back up applications in the cluster. For details, see **Application Backup**.

Step 4 **Application migration**

In this phase, you will restore backup data to migrate applications from the cluster on a third-party cloud to a Huawei Cloud cluster or multi-cloud cluster of UCS. For details, see **Application Migration**.

----End

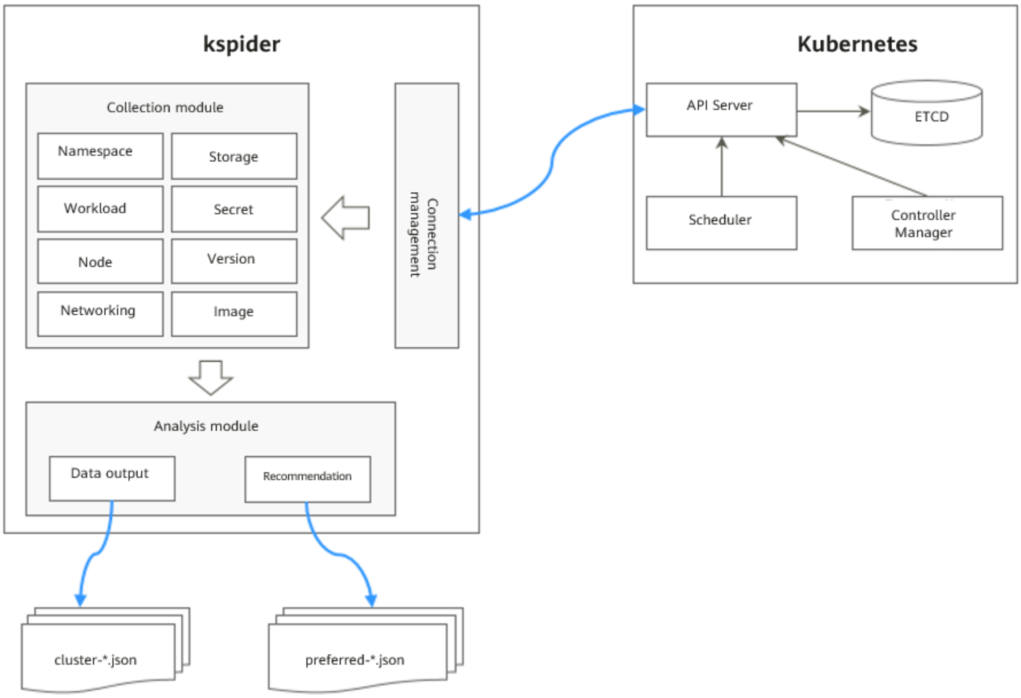
10.4.2 Cluster Evaluation

Migrating applications from one environment to another is a challenging task, so you need to plan and prepare carefully. kspider is a tool used to collect information about the source cluster. It provides cluster-related data such as the Kubernetes version, scale, workload quantity, storage, and in-use images. The data helps you understand the current status of the cluster and evaluate migration risks, and select a proper destination cluster version and scale.

How kspider Works

Figure 10-7 shows the architecture of kspider, which consists of three modules: collection, connection management, and analysis. The collection module can collect data of the source cluster, including namespaces, workloads, nodes, and networks. The connection management module establishes connections with the API Server of the source cluster. The analysis module aims to output the collected data of the source cluster (generating the **cluster-*.json** file) and provide the recommendation information of the destination cluster (generating the **preferred-*.json** file) after evaluation.

Figure 10-7 kspider architecture



Usage of kspider

NOTE

kspider can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **kspider-linux-amd64** in the following command with **kspider-linux-arm64** or **kspider-windows-amd64.exe**.

Prepare a server, upload kspider to the server, and decompress the tool package. For details, see **Preparations**. Run **./kspider-linux-amd64 -h** in the directory where kspider is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the collected namespace. By default, system namespaces such as **kube-system**, **kube-public**, and **kube-node-lease** are excluded.
- **-q, --quiet**: indicates static exit.
- **-s, --serial**: specifies the unique sequence number of the output aggregation file (**cluster-{serial}.json**) and recommendation file (**preferred-{serial}.json**).

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]
```

```
Aliases:
  kspider, kspider

Flags:
  -h, --help                help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet               command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

Step 1: Collect Data from the Source Cluster

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

Step 2 Use the default parameter settings to collect data of all namespaces in the cluster. Run the **./kspider-linux-amd64** command.

Command output:

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
    Name    CPU    Memory    IP    Arch    OS    Kernel    MachineID
    10.1.18.64    4    8008284Ki    [10.1.18.64 10.1.18.64]    amd64    linux
3.10.0-1127.19.1.el7.x86_64    ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
    10.1.19.13    4    8008284Ki    [10.1.19.13 10.1.19.13]    amd64    linux
3.10.0-1127.19.1.el7.x86_64    2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
    Name    Namespace    NodeAffinity
    minio    default    false
    minio    minio    false
There are 3 Deployments:
    Name    Namespace    NodeAffinity
    rctest    default    true
    flink-operator-controller-manager    flink-operator-system    false
    rctest    minio    false
There are 1 DaemonSets:
    Name    Namespace    NodeAffinity
    ds-nginx    minio    false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
    Namespace/Name    Pods
    default/pvc-data-minio-0    default/minio-0
    minio/obs-testing    minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
    minio/pvc-data-minio-0    minio/minio-0
There are 5 PersistentVolumes:
    Name    Namespace    pvcName    scName    size    key
    pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8    minio-pv    obs-testing    csi-obs    1Gi    pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
    pvc-c7c768aa-373a-4c52-abea-e8b486d23b47    minio-pv    pvc-data-minio-0    csi-disk-sata    10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
    pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05    minio    obs-testing    csi-obs    1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
    pvc-9fd92c99-805a-4e65-9f22-e238130983c8    default    pvc-data-minio-0    csi-disk    10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
    pvc-a22fd877-f98d-4c3d-a04e-191d79883f97    minio    pvc-data-minio-0    csi-disk-sata    10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
    Name    Namespace    ServiceType
    headless-lxprus    default    ClusterIP
    kubernetes    default    ClusterIP
```

```
    minio     default     NodePort
    flink-operator-controller-manager-metrics-service     flink-operator-system     ClusterIP
    flink-operator-webhook-service     flink-operator-system     ClusterIP
    headless-lxprus     minio     ClusterIP
    minio     minio     NodePort
There are 0 Ingresses:
There are 6 Images:
    Name
    gcr.io/flink-operator/flink-operator:v1beta1-6
    flink:1.8.2
    swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
    nginx:stable-alpine-perl
    swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
    gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
    SecretType
    cfe/secure-opaque
    helm.sh/release.v1
```

After the kspider command is executed, the following files are generated in the current directory:

- **cluster-*.json**: This file contains data collected from the source cluster and applications. The data can be used to analyze and plan the migration.
- **preferred-*.json**: This file contains information about the recommended destination cluster. A preliminary evaluation is performed for the source cluster according to its scale and node specifications. The file provides suggestions on the version and scale of the destination cluster.

Step 3 View the data collected from the source cluster and applications.

You can use a text editor or JSON viewer to open the **cluster-*.json** file to view the data. Replace the asterisk (*) in the file name with the actual timestamp or serial number to find and open the correct file.

Description of the **cluster-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Namespaces: number of namespaces. The value is a string.
  Pods: total number of pods. The value is an integer.
  Nodes: node information. The IP address is used as the key to display node information.
    IP addresses
      CPU: CPU. The value is a string.
      Arch: CPU architecture. The value is a string.
      Memory: memory. The value is a string.
      HugePages1Gi: 1 GB hugepage memory. The value is a string.
      HugePages2Mi: 2 MB hugepage memory. The value is a string.
      OS: node OS. The value is a string.
      KernelVersion: OS kernel version. The value is a string.
      RuntimeVersion: running status and version of the node container. The value is a string.
      InternalIP: internal IP address. The value is a string.
      ExternalIP: external IP address. The value is a string.
      MachineID: node ID. The value is a string. Ensure that the CCE ID is the same as the ECS ID.
  Workloads: workload
    Deployment: workload type. The value can be Deployment, StatefulSet, DaemonSet, CronJob, Job, or
LonePod.
      default: namespace name
        Count: quantity. The value is an integer.
        Items: details. The value is an array.
          Name: workload name. The value is a string.
          Namespace: namespace name. The value is a string.
          NodeAffinity: node affinity. The value is of the Boolean type.
          Replicas: number of replicas. The value is an integer.
  Storage: storage
    PersistentVolumes: persistent volume
      pv-name: The PV name is used as the key.
```

```
        VolumeID: volume ID. The value is a string.
        Namespace: namespace. The value is a string.
        PvcName: name of the bound PVC. The value is a string.
        ScName: storage class name. The value is a string.
        Size: size of the space to request. The value is a string.
        Pods: name of the pod that uses the PV. The value is a string.
        NodeIP: IP address of the node where the pod is located. The value is a string.
        VolumePath: path of the node to which the pod is mounted. The value is a string.
    OtherVolumes: volumes of other types
      Type: AzureFile, AzureDisk, GCEPersistentDisk, AWSElasticBlockStore, Cinder, Glusterfs, NFS, CephFS, 
FlexVolume, FlexVolume, DownwardAPI
        The volume ID, volume name, and volume shared path are keys.
        Pods: name of the pod. The value is a string.
          NodeIP: IP address of the node where the pod is located. The value is a string.
          Information that uniquely identifies a volume, such as the volume ID, volume name, and volume 
shared path. The value is a string.
  Networks: network
    LoadBalancer: load balancing type
      service: network type, which can be Service or Ingress.
        Name: name. The value is a string.
        Namespace: namespace name. The value is a string.
        Type: type. The value is a string.
  ExtraSecrets: extended secret type
    Secret type. The value is a string.
  Images: image
    Image repo. The value is a string.
}
```

Example:


```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
      "ExternalIP": "100.85.xxx.xxx",
      "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
    }
  },
  "Workloads": {
    "CronJob": {},
    "DaemonSet": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-prometheus-node-exporter",
```

```
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 3
        }
      ]
    }
  },
  "Deployment": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-cost-analyzer",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    },
    "kubecost": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-kube-state-metrics",
          "Namespace": "kubecost",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  },
  "Job": {},
  "LonePod": {},
  "StatefulSet": {
    "minio-all": {
      "Count": 1,
      "Items": [
        {
          "Name": "minio",
          "Namespace": "minio-all",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  }
},
"Storage": {
  "PersistentVolumes": {
    "demo": {
      "VolumeID": "demo",
      "Namespace": "fluid-demo-test",
      "PvcName": "demo",
      "ScName": "fluid",
      "Size": "100Gi",
      "Pods": "",
      "NodeIP": "",
      "VolumePath": ""
    },
    "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
      "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
      "Namespace": "minio-all",
      "PvcName": "pvc-data-minio-0",
      "ScName": "csi-disk",
      "Size": "10Gi",
      "Pods": "minio-all/minio-0",
      "NodeIP": "10.1.23.159",
      "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
```

```
      }
    },
    "OtherVolumes": {}
  },
  "Networks": {
    "LoadBalancer": {}
  },
  "ExtraSecrets": [
    "cfe/secure-opaque",
    "helm.sh/release.v1"
  ],
  "Images": [
    "nginx:stable-alpine-perl",
    "ghcr.io/koordinator-sh/koord-manager:0.6.2",
    "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
    "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
    "gcr.io/kubecost1/cost-model:prod-1.91.0",
    "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```

----End

Step 2: Evaluate the Destination Cluster

After the kspider command is executed, in addition to the **cluster-*.json** file, the **preferred-*.json** file is also generated in the current directory. After performing preliminary evaluation for the source cluster according to its scale and node specifications, the file provides the recommended version and scale of the destination cluster. This helps you better plan and prepare for the migration.

Description of the **preferred-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Scale: cluster scale. The value is a string.
  Nodes: node information
    CPU: CPU. The value is a string.
    Memory: memory. The value is a string.
    Arch: CPU architecture. The value is a string.
    KernelVersion: OS kernel version. The value is a string.
    ProxyMode: cluster proxy mode. The value is a string.
  ELB: whether the ELB service is a dependent service. The value is of the Boolean type.
}
```

Evaluation rules for each field in the preceding file:

Table 10-4 Evaluation rules

Field	Evaluation Rule
K8sVersion	If the version is earlier than 1.21, the main release version of the UCS cluster (for example, 1.21, which changes over time) is recommended. If the version is later than the main release version, the latest version of the UCS cluster is recommended.

Field	Evaluation Rule
Scale	< 25 nodes in the source cluster: Destination cluster of 50 nodes is recommended. 25 ≤ Nodes in the source cluster < 100: Destination cluster of 200 nodes is recommended. 100 ≤ Nodes in the source cluster < 500: Destination cluster of 1000 nodes is recommended. Nodes in the source cluster ≥ 500: Destination cluster of 2000 nodes is recommended.
CPU/Memory	Statistics about the specification of the largest quantity are collected.
Arch	Statistics about the specification of the largest quantity are collected.
KernelVersion	Statistics about the specification of the largest quantity are collected.
ProxyMode	Configure this parameter according to the cluster scale. For a cluster with more than 1000 nodes, **ipvs** is recommended. For a cluster with fewer than 1000 nodes, **iptables** is recommended.
ELB	Check whether the source cluster has a load balancing Service.

Example:

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

CAUTION

The evaluation result is for reference only. You need to determine the version and scale of the destination cluster.

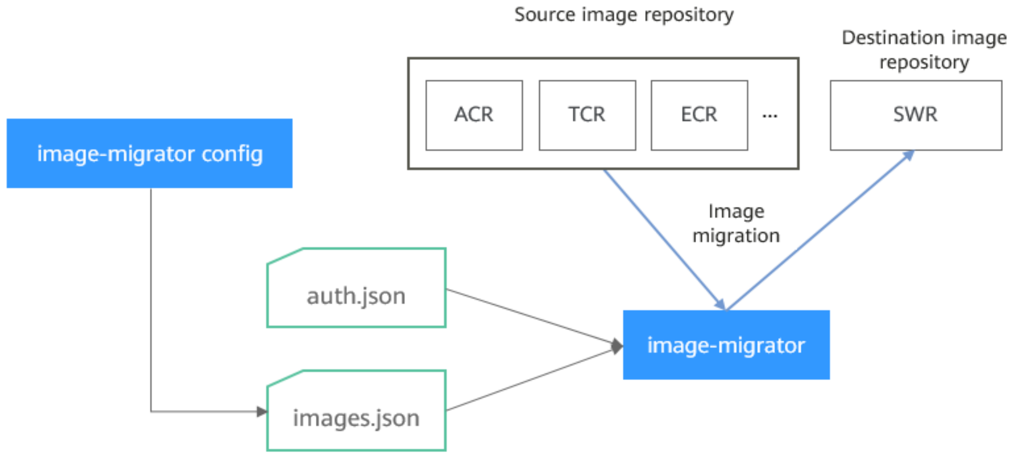
10.4.3 Image Migration

To ensure that container images can be properly pulled after cluster migration and improve container deployment efficiency, you are advised to migrate image repositories on a third-party cloud to Huawei Cloud SWR. The Huawei Cloud clusters and multi-cloud clusters of UCS work with SWR to provide a pipeline for automated container delivery. Images are pulled in parallel, which greatly improves container delivery efficiency.

image-migrator is an image migration tool that can automatically migrate images from image repositories on a third-party cloud to SWR.

How image-migrator Works

Figure 10-8 How image-migrator works



When using image-migrator to migrate images to SWR, you need to prepare two files. One is the image repository access permission file **auth.json**. The two objects in the file are the accounts and passwords of the source and destination image repositories (registries). The other is the image list file **images.json**, which consists of multiple image synchronization rules. Each rule contains a source image repository (key) and a destination image repository (value). Place these two files in the directory where image-migrator is located and run a simple command to migrate the image. The two files are described as follows:

- **auth.json**

  auth.json is the image repository access permission file. Each object is the username and password of a registry. Generally, the source image repository must have the permissions for pulling images and accessing tags, and the destination image repository must have the permissions for pushing images and creating repositories. If you access the image repository anonymously, you do not need to enter the username and password. Structure of the **auth.json** file:

```
{
    "Source image repository address": { },
    "Destination image repository address": {
        "username": "xxxxxx",
        "password": "***************",
        "insecure": true
    }
}
```

  For details about the parameters in this file, see **Table 10-5**.

Table 10-5 Parameters in the **auth.json** file

Parameter	Description
Source image repository address	The value can be in *registry* or *registry/namespace* format, which must correspond to the *registry* or *registry/namespace* format in **images.json**. **NOTE** The matched URL in images uses the corresponding username and password for image synchronization. The *registry/namespace* format is preferred.
Destination image repository address	The value can be in *registry* or *registry/namespace* format, which must correspond to the *registry* or *registry/namespace* format in **images.json**. ● If your image repository is Huawei Cloud SWR and the destination image repository address is in the *registry* format, you can obtain it from the SWR console as follows: On the **Dashboard** page, click **Generate Login Command** in the upper right corner. The domain name at the end of the login command is the SWR image repository address, for example, **swr.cn-north-4.myhuaweicloud.com**. Note that the address varies with the region. Switch to the corresponding region to obtain the address. If the value is in the *registry/namespace* format, replace *namespace* with the organization name of SWR. ● If your image repository is Amazon ECR or ACR, log in to the image repository console of the corresponding vendor and view the push command of the image repository to obtain the image repository address.
username	Username. You can set it to a specific value or use a string of the ${env} or $env type to reference an environment variable. ● If your image repository is Huawei Cloud SWR, the username of the destination SWR image repository is in the following format: *Regional project name***@AK**. ● If your image repository is Amazon ECR or ACR, log in to the image repository console of the corresponding vendor and view the push command of the image repository to obtain the corresponding username.
password	Password. You can set it to a specific value or use a string of the ${env} or $env type to reference an environment variable. ● If your image repository is Huawei Cloud SWR, the password of the destination SWR image repository is the encrypted login key of the AK and SK. For details, see **Obtaining a Long-Term Valid Login Command**. ● If your image repository is Amazon ECR or ACR, log in to the image repository console of the corresponding vendor and view the push command of the image repository to obtain the corresponding password.

Parameter	Description
insecure	Whether *registry* is an HTTP service. If yes, the value of **insecure** is **true**. The default value is **false**.

Example:

```
{
    "quay.io/coreos": { },
    "swr.cn-north-4.myhuaweicloud.com": {
        "username": "cn-north-4@RVHVMX******",
        "password": "***************",
        "insecure": true
    }
}
```

- **images.json**

  This file is essentially a list of images to migrate and consists of multiple image synchronization rules. Each rule contains a source image repository (key) and a destination image repository (value). The specific requirements are as follows:

  a. The largest unit that can be synchronized using one rule is repository. The entire namespace or registry cannot be synchronized using one rule.
  b. The formats of the source and destination repositories are similar to those of the image URL used by the **docker pull/push** command (**registry/namespace/repository:tag**).
  c. Both the source and destination repositories (if the destination repository is not an empty string) contain at least *registry*/*namespace*/*repository*.
  d. The source repository field cannot be empty. To synchronize data from a source repository to multiple destination repositories, you need to configure multiple rules.
  e. The destination repository name can be different from the source repository name. In this case, the synchronization function is similar to docker pull + docker tag + docker push.
  f. If the source repository field does not contain tags, all tags of the repository have been synchronized to the destination repository. In this case, the destination repository cannot contain tags.
  g. If the source repository field contains tags, only one tag in the source repository has been synchronized to the destination repository. If the destination repository does not contain tags, the source tag is used by default.
  h. If the destination repository is an empty string, the source image will be synchronized to the default namespace of the default registry. The repository and tag are the same as those of the source repository. The default registry and namespace can be configured using command line parameters and environment variables.

  Example:

```
{
    "quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
    "quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
    "quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
}
```

You can use a config subcommand of the image-migrator tool to automatically obtain the image that is being used by the workload in the cluster. For details, see **Usage of image-migrator config**. After obtaining the **images.json** file, you can modify, add, or delete its content as required.

Usage of image-migrator

NOTE

image-migrator can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **image-migrator-linux-amd64** in the following command with **image-migrator-linux-arm64** or **image-migrator-windows-amd64.exe**.

Run **./image-migrator-linux-amd64 -h** in the directory where image-migrator is located to learn about its usage.

- **--auth**: specifies the path of **auth.json**. By default, **auth.json** is stored in the directory where image-migrator is located.
- **--images**: specifies the path of **images.json**. By default, **images.json** is stored in the directory where image-migrator is located.
- **--log**: specifies the path for storing logs generated by image-migrator. The default value is **image-migrator.log** in the current directory of image-migrator.
- **--namespace**: specifies the default namespace of the destination repository. That is, if the namespace of the destination repository is not specified in **images.json**, you can specify it when running the migration command.
- **--registry**: specifies the default registry of the destination repository. That is, if the registry of the destination repository is not specified in **images.json**, you can specify it when running the migration command.
- **--retries**: specifies the number of retry times when the migration fails. The default value is **3**.
- **--workers**: specifies the number of concurrent workers for image migration. The default value is **7**.

```
$ ./image-migrator-linux-amd64 -h
A Fast and Flexible docker registry image images tool implement by Go.

Usage:
  image-migrator [flags]

Aliases:
  image-migrator, image-migrator

Flags:
      --auth string        auth file path. This flag need to be pair used with --images. (default "./auth.json")
  -h, --help              help for image-migrator
      --images string      images file path. This flag need to be pair used with --auth (default "./images.json")
      --log string         log file path (default "./image-migrator.log")
      --namespace string   default target namespace when target namespace is not given in the images
config file, can also be set with DEFAULT_NAMESPACE environment value
      --registry string    default target registry url when target registry is not given in the images config file,
can also be set with DEFAULT_REGISTRY environment value
  -r, --retries int        times to retry failed tasks (default 3)
  -w, --workers int        numbers of working goroutines (default 7)

$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \
--registry=swr.cn-north-4.myhuaweicloud.com --retries=2
$ ./image-migrator
```

```
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

Example:

./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

The preceding command is used to migrate the images in the **images.json** file to the image repository **swr.cn-north-4.myhuaweicloud.com/test**. If the migration fails, you can retry twice. A maximum of five images can be migrated at a time.

Usage of image-migrator config

The config subcommand of image-migrator can be used to obtain images used in cluster applications and generate the **images.json** file in the directory where the tool is located. You can run **./image-migrator-linux-amd64 config -h** to learn how to use the config subcommand.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the namespace of the image to be obtained. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that images of all namespaces are obtained.
- **-t, --repo**: specifies the destination repository address (*registry*/*namespace*).

```
$ ./image-migrator-linux-amd64 config -h
generate images.json

Usage:
  image-migrator config [flags]

Flags:
  -h, --help                help for config
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "/
root/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -t, --repo string         target repo,such as swr.cn-north-4.myhuaweicloud.com/test
```

Examples:

- Specify a namespace:

  ./image-migrator-linux-amd64 config -n default -t swr.cn-north-4.myhuaweicloud.com/test
- Specify multiple namespaces:

  ./image-migrator-linux-amd64 config -n default,kube-system -t swr.cn-north-4.myhuaweicloud.com/test
- If no namespace is specified, images of all namespaces are obtained:

  ./image-migrator-linux-amd64 config -t swr.cn-north-4.myhuaweicloud.com/test

Procedure

Step 1 Prepare the image repository access permission file **auth.json**.

Create an **auth.json** file and modify it based on the format. If the repository is accessed anonymously, you do not need to enter information such as the username and password. Place the file in the directory where image-migrator is located.

Example:

```
{
    "quay.io/coreos": { },
    "swr.cn-north-4.myhuaweicloud.com": {
        "username": "cn-north-4@RVHVMX******",
        "password": "***************",
        "insecure": true
    }
}
```

For details about the parameters, see the **auth.json file**.

Step 2 Prepare the image list file **images.json**.

1. Connect to the source cluster using kubectl. For details, see **Connecting to a Cluster Using kubectl**.
2. Run the config subcommand for image migration to generate the **images.json** file.

   You can refer to the methods and examples in **Usage of image-migrator config** to obtain the image used in the source cluster application without specifying the namespace, or by specifying one or more namespaces.
3. Modify the **images.json** file as required. Ensure that the file meets the eight requirements described in **images.json file**.

Step 3 Migrate images.

You can run the default **./image-migrator-linux-amd64** command to migrate images or configure image-migrator parameters as required.

For example, run the following command:

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

Example:

```
$ ./image-migrator-linux-amd64
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

Step 4 View the result.

After the preceding command is executed, information similar to the following is displayed:

```
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

The preceding information indicates that 38 images have been migrated to the SWR repository.

----End

10.4.4 Dependent Service Migration

Migrate data of services on which the cluster depends, such as storage, database, distributed cache, and distributed message. If your cluster does not involve the data of these services or the data does not need to be migrated to Huawei Cloud, skip this section.

Storage Migration

- If your cluster uses EVS disks, you can use Huawei Cloud **Data Express Service (DES)** for cross-cloud migration. DES provides you with physical devices to migrate hundreds of terabytes of data to Huawei Cloud inexpensively and much faster than would be possible over a network connection.
- If your cluster uses object storage, you can use Huawei Cloud **Object Storage Migration Service (OMS)** for cross-cloud migration. OMS is an online data migration service. It can migrate data from object storage services of other cloud service providers to Huawei Cloud Object Storage Service (OBS).
- If your cluster uses SFS for storage, you can use Huawei Cloud **Scalable File Service (SFS)** for cross-cloud migration.

Database Migration

If you need to migrate databases to Huawei Cloud, you can use **Data Replication Service (DRS)**. DRS provides multiple functions, including real-time migration, backup migration, real-time synchronization, data subscription, and real-time DR.

Migrating Other Data

- Big data migration: **Cloud Data Migration (CDM)** on Huawei Cloud
- Kafka service migration: **Migrating Kafka Services** using Distributed Message Service (DMS) on Huawei Cloud for Kafka
- Redis service migration: **Data Migration Guide** of Distributed Cache Service (DCS) on Huawei Cloud

10.4.5 Application Backup

Application migration from the cluster on a third-party cloud consists of two steps: application backup and application migration. That is, applications in the cluster on a third-party cloud are backed up and then migrated to the destination cluster through data restoration.

k8clone is a simple Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster (Huawei Cloud cluster or multi-cloud cluster of UCS). In this way, applications can be migrated from clusters on a third-party cloud to the cloud.

NOTICE

Back up data during off-peak hours.

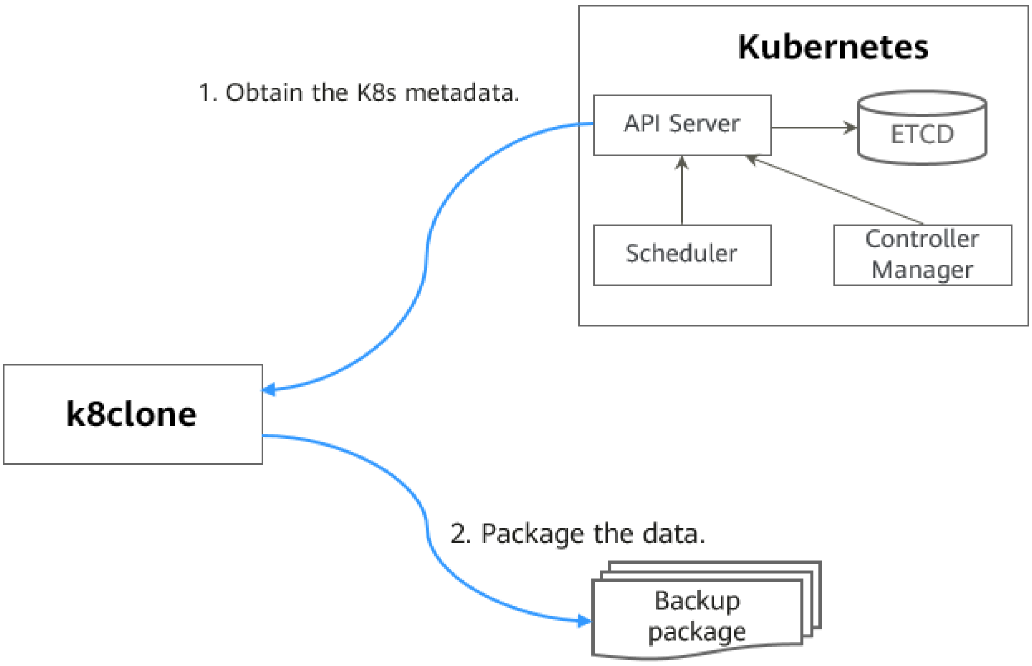
Prerequisites

Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.

How k8clone Backs Up Data

Data backup process:

Figure 10-9 Data backup process



k8clone Usage for Backup

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 backup -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.

- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-n, --namespace**: backs up cloud native applications of a specified namespace. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that the entire cluster is backed up.
- **-e, --exclude-namespaces**: excludes the backup of objects of a specified namespace. This parameter cannot be used together with **--namespace**.
- **-x, --exclude-kind**: excludes the backup of a specified resource type.
- **-i, --include-kind**: specifies the backup of a resource type.
- **-y, --exclude-object**: excludes the backup of a specified resource object.
- **-z, --include-object**: specifies the backup of a resource object.
- **-w, --exclude-having-owner-ref**: excludes the backup of resource objects with ownerReferences. The default value is **false**. The equal sign (=) must be contained in the parameter transfer process, for example, -w=true.
- **-d, --local-dir**: path for storing backup data. The default value is the **k8clone-dump** folder in the current directory.

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
  k8clone backup [flags]

Flags:
  -s, --api-server string            Kubernetes api-server url
  -q, --context string               Kubernetes configuration context
-w, --exclude-having-owner-ref      Exclude all objects having an Owner Reference. The following form is
not permitted for boolean flags such as '-w false', please use '-w=false'
  -x, --exclude-kind strings         Resource kind to exclude. Eg. 'deployment'
  -i, --include-kind strings         Ressource kind to include. Eg. 'deployment'
  -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all
namespaces and then filters them. Don't use it with the namespace flag.
  -y, --exclude-object strings       Object to exclude. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -z, --include-object strings       Object to include. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -h, --help                         help for backup
  -k, --kubeconfig string            The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string             Where to dump yaml files (default "./k8clone-dump")
  -n, --namespace string             Only dump objects from this namespace
```

Examples:

- Backs up objects of the entire cluster. The default path is the **k8clone-dump** folder in the current directory.

  ./k8clone-linux-amd64 backup
- Backs up objects of the entire cluster and specifies the path for storing backup data.

  ./k8clone-linux-amd64 backup -d ./xxxx
- Backs up objects of a specified namespace.

  ./k8clone-linux-amd64 backup -n default
- Excludes the backup of objects of a specified namespace.

./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease

- Excludes the backup of specified resource types.

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice

- Specifies the backup of resource types.

  ./k8clone-linux-amd64 backup -x rolebinding

- Excludes the backup of specified resource objects.

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns

- Specifies the backup of resource objects.

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns

- Excludes the backup of resource objects with ownerReferences.

  ./k8clone-linux-amd64 backup -w=true

Procedure

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

Step 2 Go to the directory where k8clone is located and run the backup command to back up data to a local directory and compress the data into a package.

The examples in **k8clone Usage for Backup** provide several common backup methods. You can select a method as required or customize one.

----End

10.4.6 Application Migration

Application migration from the cluster on a third-party cloud consists of two steps: application backup and application migration. That is, applications in the cluster on a third-party cloud are backed up and then migrated to the destination cluster through data restoration.

k8clone is a simple Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster (Huawei Cloud cluster or on-premises cluster of UCS). In this way, applications can be migrated from clusters in an on-premises data center to the cloud.

Constraints

Currently, applications in a cluster of a later version cannot be migrated to a cluster of an earlier version.

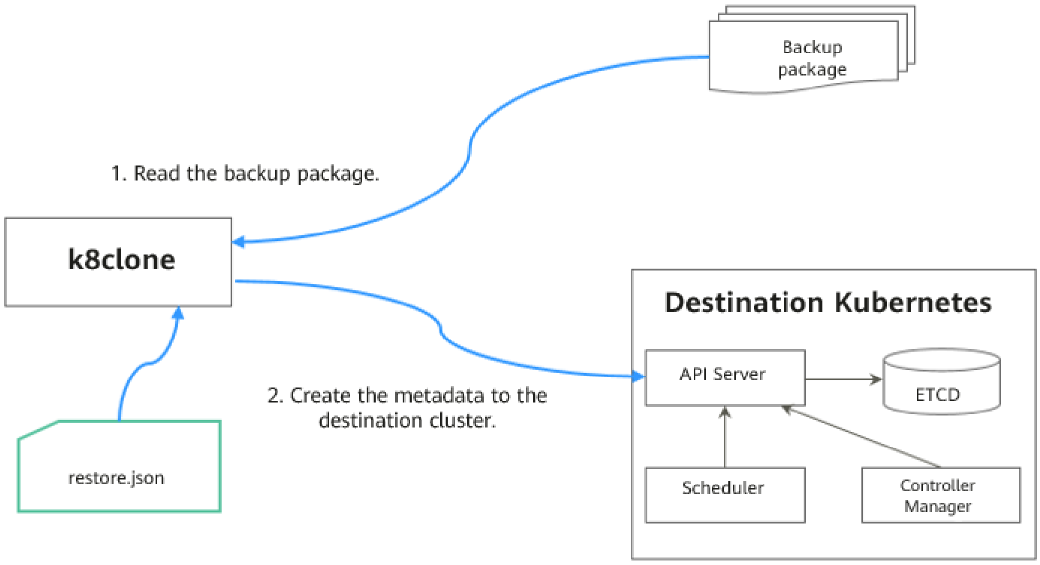
Prerequisites

- Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.
- Ensure that the metadata backup in the source cluster has been downloaded to the server where k8clone is executed.

How k8clone Restores Data

Data restoration process:

Figure 10-10 Data restoration process



Before the restoration, prepare a data restoration configuration file **restore.json** to automatically change the storage class names of PVC and StatefulSet and the repository address of the image used by the workload during application restoration.

The file content is as follows:

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  // The StorageClassName field of PVC and
StatefulSet can be changed.
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-
north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- **StorageClass**: The storage class names of PVC and VolumeClaimTemplates can be automatically changed based on settings.
- **ImageRepo**: The repository address of the image used by the workload can be changed. The workload can be Deployment (including initContainer), StatefulSet, Orphaned Pod, Job, CronJob, Replica Set, Replication Controller, and DaemonSet.

k8clone Usage for Restoration

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 restore -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-f, --restore-conf**: path of **restore.json**. The default value is the directory where k8clone is located.
- **-d, --local-dir**: path for storing backup data. The default value is the directory where k8clone is located.

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
    k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

Procedure

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

Step 2 Prepare the data restoration configuration file **restore.json**.

Create a **restore.json** file, modify it based on the format, and place it in the directory where k8clone is located.

Example:

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

Step 3 Go to the directory where k8clone is located and run the restoration command to restore the backup data to the destination cluster.

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

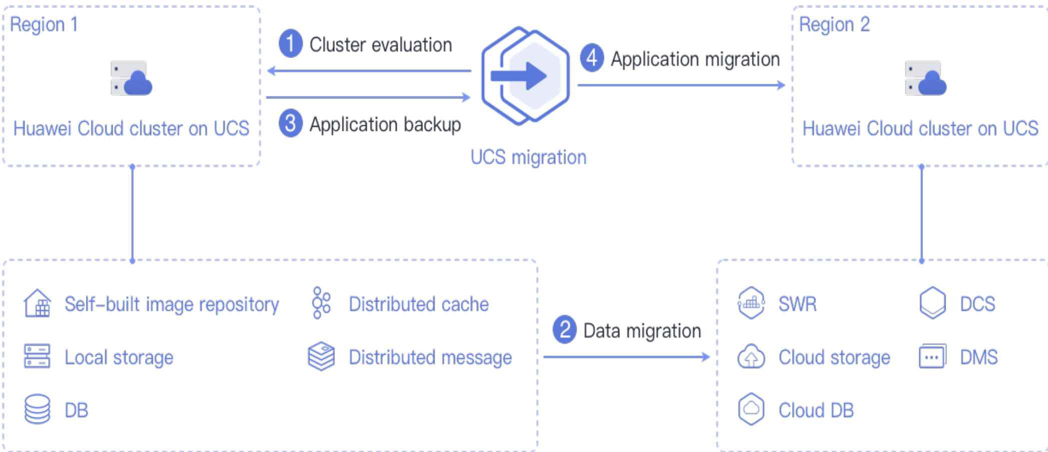
----End

10.5 Migration Across Huawei Cloud Clusters of UCS in Different Regions

10.5.1 Migration Process

Applications can be migrated between Kubernetes clusters managed by Huawei Cloud UCS from one geographic region to another to meet data compliance, latency, and availability requirements.

Figure 10-11 Migration process



The process is as follows:

Step 1 **Cluster evaluation**

In this phase, you will evaluate the status of the source cluster to determine the type of the destination cluster. UCS kspider can automatically collect information about the source cluster, including the Kubernetes version, cluster scale, workload, and storage, and provide you with information about the recommended destination cluster. For details, see **Cluster Evaluation**.

Step 2 **Data migration**

In this phase, you will migrate images and data related to dependent services to the destination region. You can use the image synchronization function of SWR to migrate images across regions.

For details about how to migrate data of dependent services, see the cross-region migration guides of Huawei Cloud products. For details, see **Data Migration**.

Step 3 **Application backup**

In this phase, you will back up applications in the source region cluster. UCS k8clone can automatically collect Kubernetes metadata and save it as a compressed package to the local host to back up applications in the cluster. For details, see **Application Backup**.

Step 4 **Application migration**

In this phase, you will migrate applications from the source region cluster to the destination region cluster by restoring backup data. For details, see **Application Migration**.

----End

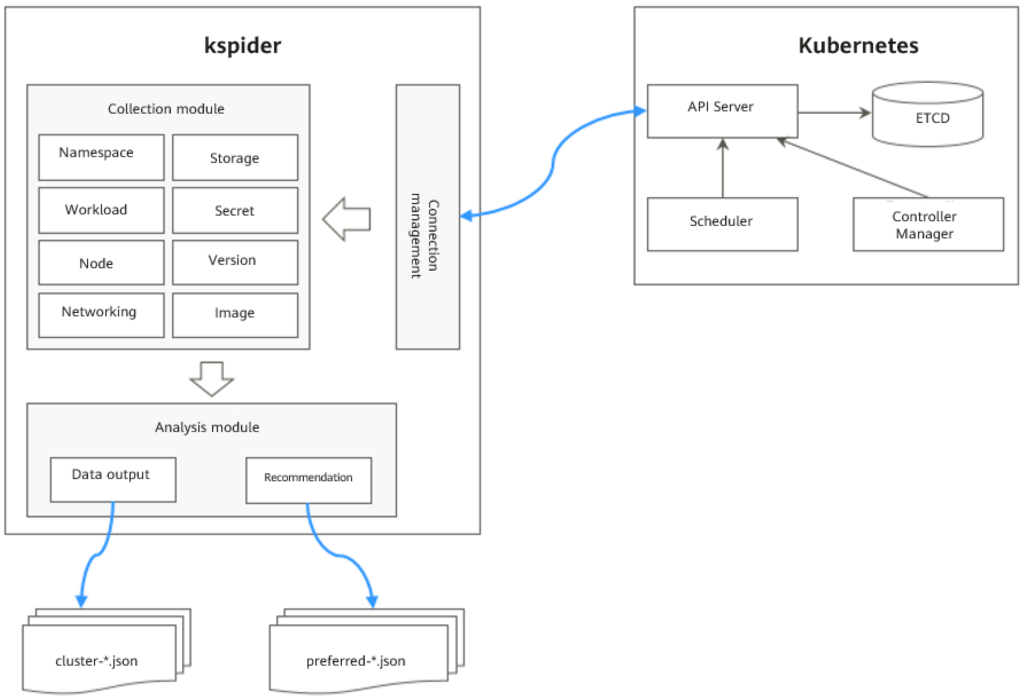
10.5.2 Cluster Evaluation

Migrating applications from one environment to another is a challenging task, so you need to plan and prepare carefully. kspider is a tool used to collect information about the source cluster. It provides cluster-related data such as the Kubernetes version, scale, workload quantity, storage, and in-use images. The data helps you understand the current status of the cluster and evaluate migration risks, and select a proper destination cluster version and scale.

How kspider Works

Figure 10-12 shows the architecture of kspider, which consists of three modules: collection, connection management, and analysis. The collection module can collect data of the source cluster, including namespaces, workloads, nodes, and networks. The connection management module establishes connections with the API Server of the source cluster. The analysis module aims to output the collected data of the source cluster (generating the **cluster-*.json** file) and provide the recommendation information of the destination cluster (generating the **preferred-*.json** file) after evaluation.

Figure 10-12 kspider architecture



Usage of kspider

NOTE

kspider can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **kspider-linux-amd64** in the following command with **kspider-linux-arm64** or **kspider-windows-amd64.exe**.

Prepare a server, upload kspider to the server, and decompress the tool package. For details, see **Preparations**. Run **./kspider-linux-amd64 -h** in the directory where kspider is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the collected namespace. By default, system namespaces such as **kube-system**, **kube-public**, and **kube-node-lease** are excluded.
- **-q, --quiet**: indicates static exit.
- **-s, --serial**: specifies the unique sequence number of the output aggregation file (**cluster-{serial}.json**) and recommendation file (**preferred-{serial}.json**).

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]

Aliases:
  kspider, kspider

Flags:
  -h, --help                help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet               command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

Step 1: Collect Data from the Source Cluster

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

Step 2 Use the default parameter settings to collect data of all namespaces in the cluster. Run the **./kspider-linux-amd64** command.

Command output:

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
    Name      CPU    Memory     IP     Arch    OS    Kernel     MachineID
    10.1.18.64    4    8008284Ki    [10.1.18.64 10.1.18.64]    amd64    linux
3.10.0-1127.19.1.el7.x86_64     ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
```

```
   10.1.19.13    4    8008284Ki    [10.1.19.13 10.1.19.13]    amd64    linux
3.10.0-1127.19.1.el7.x86_64    2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
   Name    Namespace    NodeAffinity
   minio    default    false
   minio    minio    false
There are 3 Deployments:
   Name    Namespace    NodeAffinity
   rctest    default    true
   flink-operator-controller-manager    flink-operator-system    false
   rctest    minio    false
There are 1 DaemonSets:
   Name    Namespace    NodeAffinity
   ds-nginx    minio    false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
   Namespace/Name    Pods
   default/pvc-data-minio-0    default/minio-0
   minio/obs-testing    minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
   minio/pvc-data-minio-0    minio/minio-0
There are 5 PersistentVolumes:
   Name    Namespace    pvcName    scName    size    key
   pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8    minio-pv    obs-testing    csi-obs    1Gi    pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
   pvc-c7c768aa-373a-4c52-abea-e8b486d23b47    minio-pv    pvc-data-minio-0    csi-disk-sata    10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
   pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05    minio    obs-testing    csi-obs    1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
   pvc-9fd92c99-805a-4e65-9f22-e238130983c8    default    pvc-data-minio-0    csi-disk    10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
   pvc-a22fd877-f98d-4c3d-a04e-191d79883f97    minio    pvc-data-minio-0    csi-disk-sata    10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
   Name    Namespace    ServiceType
   headless-lxprus    default    ClusterIP
   kubernetes    default    ClusterIP
   minio    default    NodePort
   flink-operator-controller-manager-metrics-service    flink-operator-system    ClusterIP
   flink-operator-webhook-service    flink-operator-system    ClusterIP
   headless-lxprus    minio    ClusterIP
   minio    minio    NodePort
There are 0 Ingresses:
There are 6 Images:
   Name
   gcr.io/flink-operator/flink-operator:v1beta1-6
   flink:1.8.2
   swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
   nginx:stable-alpine-perl
   swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
   gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
   SecretType
   cfe/secure-opaque
   helm.sh/release.v1
```

After the kspider command is executed, the following files are generated in the current directory:

- **cluster-*.json**: This file contains data collected from the source cluster and applications. The data can be used to analyze and plan the migration.
- **preferred-*.json**: This file contains information about the recommended destination cluster. A preliminary evaluation is performed for the source cluster according to its scale and node specifications. The file provides suggestions on the version and scale of the destination cluster.

Step 3 View the data collected from the source cluster and applications.

You can use a text editor or JSON viewer to open the **cluster-*.json** file to view the data. Replace the asterisk (*) in the file name with the actual timestamp or serial number to find and open the correct file.

Description of the **cluster-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Namespaces: number of namespaces. The value is a string.
  Pods: total number of pods. The value is an integer.
  Nodes: node information. The IP address is used as the key to display node information.
    IP addresses
      CPU: CPU. The value is a string.
      Arch: CPU architecture. The value is a string.
      Memory: memory. The value is a string.
      HugePages1Gi: 1 GB hugepage memory. The value is a string.
      HugePages2Mi: 2 MB hugepage memory. The value is a string.
      OS: node OS. The value is a string.
      KernelVersion: OS kernel version. The value is a string.
      RuntimeVersion: running status and version of the node container. The value is a string.
      InternalIP: internal IP address. The value is a string.
      ExternalIP: external IP address. The value is a string.
      MachineID: node ID. The value is a string. Ensure that the CCE ID is the same as the ECS ID.
  Workloads: workload
    Deployment: workload type. The value can be Deployment, StatefulSet, DaemonSet, CronJob, Job, or
LonePod.
      default: namespace name
        Count: quantity. The value is an integer.
        Items: details. The value is an array.
          Name: workload name. The value is a string.
          Namespace: namespace name. The value is a string.
          NodeAffinity: node affinity. The value is of the Boolean type.
          Replicas: number of replicas. The value is an integer.
  Storage: storage
    PersistentVolumes: persistent volume
      pv-name: The PV name is used as the key.
        VolumeID: volume ID. The value is a string.
        Namespace: namespace. The value is a string.
        PvcName: name of the bound PVC. The value is a string.
        ScName: storage class name. The value is a string.
        Size: size of the space to request. The value is a string.
        Pods: name of the pod that uses the PV. The value is a string.
        NodeIP: IP address of the node where the pod is located. The value is a string.
        VolumePath: path of the node to which the pod is mounted. The value is a string.
    OtherVolumes: volumes of other types
      Type: AzureFile, AzureDisk, GCEPersistentDisk, AWSElasticBlockStore, Cinder, Glusterfs, NFS, CephFS,
FlexVolume, FlexVolume, DownwardAPI
        The volume ID, volume name, and volume shared path are keys.
        Pods: name of the pod. The value is a string.
          NodeIP: IP address of the node where the pod is located. The value is a string.
          Information that uniquely identifies a volume, such as the volume ID, volume name, and volume
shared path. The value is a string.
  Networks: network
    LoadBalancer: load balancing type
      service: network type, which can be Service or Ingress.
        Name: name. The value is a string.
        Namespace: namespace name. The value is a string.
        Type: type. The value is a string.
  ExtraSecrets: extended secret type
    Secret type. The value is a string.
  Images: image
    Image repo. The value is a string.
}
```

Example:

```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
      "ExternalIP": "100.85.xxx.xxx",
      "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
    }
  },
  "Workloads": {
    "CronJob": {},
    "DaemonSet": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-prometheus-node-exporter",
            "Namespace": "default",
            "NodeAffinity": false,
            "Replicas": 3
          }
        ]
      }
    },
    "Deployment": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-cost-analyzer",
            "Namespace": "default",
            "NodeAffinity": false,
            "Replicas": 1
          }
        ]
      },
      "kubecost": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-kube-state-metrics",
            "Namespace": "kubecost",
            "NodeAffinity": false,
            "Replicas": 1
          }
        ]
```

```
      }
    },
    "Job": {},
    "LonePod": {},
    "StatefulSet": {
      "minio-all": {
        "Count": 1,
        "Items": [
          {
            "Name": "minio",
            "Namespace": "minio-all",
            "NodeAffinity": false,
            "Replicas": 1
          }
        ]
      }
    }
  },
  "Storage": {
    "PersistentVolumes": {
      "demo": {
        "VolumeID": "demo",
        "Namespace": "fluid-demo-test",
        "PvcName": "demo",
        "ScName": "fluid",
        "Size": "100Gi",
        "Pods": "",
        "NodeIP": "",
        "VolumePath": ""
      },
      "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
        "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
        "Namespace": "minio-all",
        "PvcName": "pvc-data-minio-0",
        "ScName": "csi-disk",
        "Size": "10Gi",
        "Pods": "minio-all/minio-0",
        "NodeIP": "10.1.23.159",
        "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
      }
    },
    "OtherVolumes": {}
  },
  "Networks": {
    "LoadBalancer": {}
  },
  "ExtraSecrets": [
    "cfe/secure-opaque",
    "helm.sh/release.v1"
  ],
  "Images": [
    "nginx:stable-alpine-perl",
    "ghcr.io/koordinator-sh/koord-manager:0.6.2",
    "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
    "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
    "gcr.io/kubecost1/cost-model:prod-1.91.0",
    "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```


----End

Step 2: Evaluate the Destination Cluster

After the kspider command is executed, in addition to the **cluster-*.json** file, the **preferred-*.json** file is also generated in the current directory. After performing preliminary evaluation for the source cluster according to its scale and node specifications, the file provides the recommended version and scale of the destination cluster. This helps you better plan and prepare for the migration.

Description of the **preferred-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Scale: cluster scale. The value is a string.
  Nodes: node information
    CPU: CPU. The value is a string.
    Memory: memory. The value is a string.
    Arch: CPU architecture. The value is a string.
    KernelVersion: OS kernel version. The value is a string.
    ProxyMode: cluster proxy mode. The value is a string.
  ELB: whether the ELB service is a dependent service. The value is of the Boolean type.
}
```

Evaluation rules for each field in the preceding file:

Table 10-6 Evaluation rules

Field	Evaluation Rule
K8sVersion	If the version is earlier than 1.21, the main release version of the UCS cluster (for example, 1.21, which changes over time) is recommended. If the version is later than the main release version, the latest version of the UCS cluster is recommended.
Scale	< 25 nodes in the source cluster: Destination cluster of 50 nodes is recommended. 25 ≤ Nodes in the source cluster < 100: Destination cluster of 200 nodes is recommended. 100 ≤ Nodes in the source cluster < 500: Destination cluster of 1000 nodes is recommended. Nodes in the source cluster ≥ 500: Destination cluster of 2000 nodes is recommended.
CPU/Memory	Statistics about the specification of the largest quantity are collected.
Arch	Statistics about the specification of the largest quantity are collected.
KernelVersion	Statistics about the specification of the largest quantity are collected.
ProxyMode	Configure this parameter according to the cluster scale. For a cluster with more than 1000 nodes, **ipvs** is recommended. For a cluster with fewer than 1000 nodes, **iptables** is recommended.
ELB	Check whether the source cluster has a load balancing Service.

Example:

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

CAUTION

The evaluation result is for reference only. You need to determine the version and scale of the destination cluster.

10.5.3 Data Migration

Migrate data of services on which the cluster and image depends, such as cloud storage, cloud database, distributed cache, and distributed message.

Image Migration

You can use the image synchronization function of SWR to migrate images across regions.

For existing images in the image repository, you need to manually synchronize the images to the destination region. Images and image updates can be automatically synchronized between regions.

For details, see **Configuring Automatic Image Synchronization Between Regions**.

Cloud Storage Migration

If your cluster uses EVS disks or SFS file systems, you can use **Cloud Backup and Recovery (CBR)** for cross-region migration. CBR lets you back up ECSs, BMSs, disks, and on-premises VMware VMs. In case of a virus intrusion, accidental deletion, or software/hardware fault, data can be restored to any backup point.

For details, see **Creating a Cloud Disk Backup** or **Creating an SFS Turbo Backup**.

Cloud Database Migration

Data Replication Service (DRS) can be used to migrate cloud databases across regions. DRS provides multiple functions, including real-time migration, backup migration, real-time synchronization, data subscription, and real-time DR.

Migrating Other Data

- Big data migration: **Cloud Data Migration (CDM)**
- Kafka service migration: **Migrating Kafka Services** using Distributed Message Service (DMS) for Kafka

- Redis service migration: **Data Migration Guide** of Distributed Cache Service (DCS)

10.5.4 Application Backup

Application migration across Huawei Cloud clusters of UCS in different regions consists of application backup and application migration. This means that applications in the source cluster are backed up and then migrated to the destination cluster through data restoration.

k8clone is an easy-to-use Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster.

NOTICE

Back up data during off-peak hours.

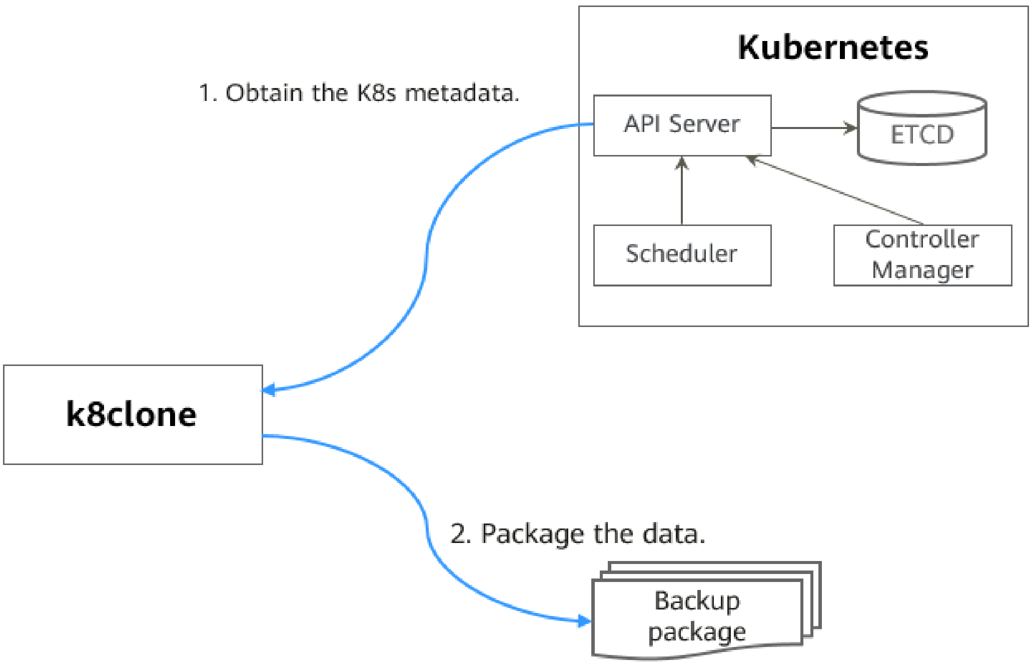
Prerequisites

Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.

How k8clone Backs Up Data

Data backup process:

Figure 10-13 Data backup process



k8clone Usage for Backup

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 backup -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-n, --namespace**: backs up cloud native applications of a specified namespace. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that the entire cluster is backed up.
- **-e, --exclude-namespaces**: excludes the backup of objects of a specified namespace. This parameter cannot be used together with **--namespace**.
- **-x, --exclude-kind**: excludes the backup of a specified resource type.
- **-i, --include-kind**: specifies the backup of a resource type.
- **-y, --exclude-object**: excludes the backup of a specified resource object.
- **-z, --include-object**: specifies the backup of a resource object.
- **-w, --exclude-having-owner-ref**: excludes the backup of resource objects with ownerReferences. The default value is **false**. The equal sign (=) must be contained in the parameter transfer process, for example, -w=true.
- **-d, --local-dir**: path for storing backup data. The default value is the **k8clone-dump** folder in the current directory.

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
   k8clone backup [flags]

Flags:
   -s, --api-server string            Kubernetes api-server url
   -q, --context string               Kubernetes configuration context
-w, --exclude-having-owner-ref      Exclude all objects having an Owner Reference. The following form is
not permitted for boolean flags such as '-w false', please use '-w=false'
   -x, --exclude-kind strings         Resource kind to exclude. Eg. 'deployment'
   -i, --include-kind strings         Ressource kind to include. Eg. 'deployment'
   -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all
namespaces and then filters them. Don't use it with the namespace flag.
   -y, --exclude-object strings       Object to exclude. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -z, --include-object strings       Object to include. The form is '<kind>:<namespace>/<name>',namespace
can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -h, --help                         help for backup
   -k, --kubeconfig string            The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
```



```
-d, --local-dir string          Where to dump yaml files (default "./k8clone-dump")
-n, --namespace string            Only dump objects from this namespace
```

Examples:

- Backs up objects of the entire cluster. The default path is the **k8clone-dump** folder in the current directory.

  ./k8clone-linux-amd64 backup
- Backs up objects of the entire cluster and specifies the path for storing backup data.

  ./k8clone-linux-amd64 backup -d ./xxxx
- Backs up objects of a specified namespace.

  ./k8clone-linux-amd64 backup -n default
- Excludes the backup of objects of a specified namespace.

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease
- Excludes the backup of specified resource types.

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice
- Specifies the backup of resource types.

  ./k8clone-linux-amd64 backup -x rolebinding
- Excludes the backup of specified resource objects.

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns
- Specifies the backup of resource objects.

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns
- Excludes the backup of resource objects with ownerReferences.

  ./k8clone-linux-amd64 backup -w=true

Procedure

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

Step 2 Go to the directory where k8clone is located and run the backup command to back up data to a local directory and compress the data into a package.

The examples in **k8clone Usage for Backup** provide several common backup methods. You can select a method as required or customize one.

----End

10.5.5 Application Migration

Application migration across Huawei Cloud clusters of UCS in different regions consists of application backup and application migration. This means that applications in the source cluster are backed up and then migrated to the destination cluster through data restoration.

k8clone is an easy-to-use Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster.

Constraints

Currently, applications in a cluster of a later version cannot be migrated to a cluster of an earlier version.

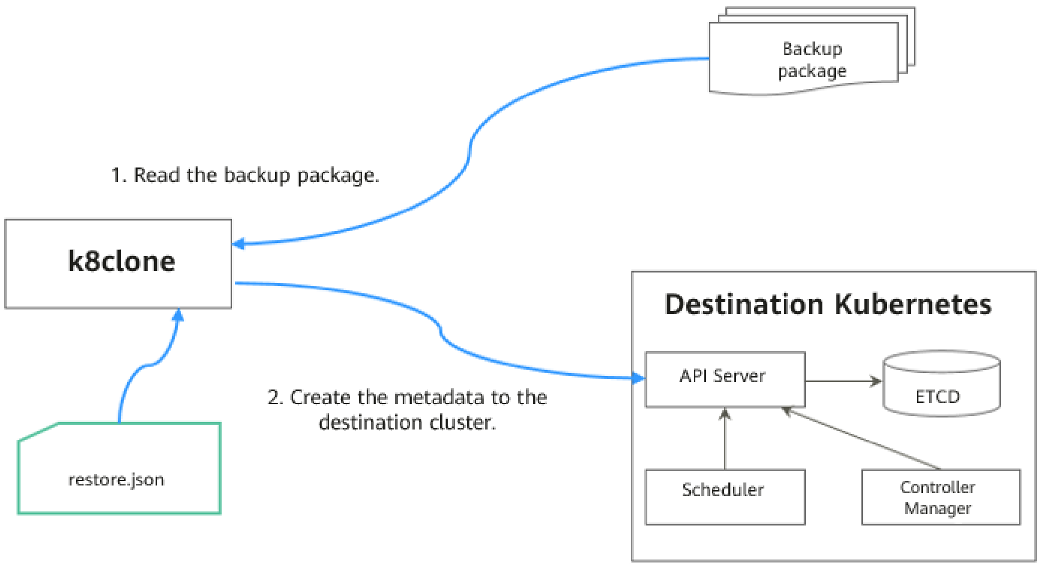
Prerequisites

- Ensure that services (data not in the cluster, such as images, storage, and databases) on which cloud native applications depend have been migrated.
- Ensure that the metadata backup in the source cluster has been downloaded to the server where k8clone is executed.

How k8clone Restores Data

Data restoration process:

Figure 10-14 Data restoration process



Before the restoration, prepare a data restoration configuration file **restore.json** to automatically change the storage class names of PVC and StatefulSet and the repository address of the image used by the workload during application restoration.

The file content is as follows:

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  // The StorageClassName field of PVC and StatefulSet can be changed.
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- **StorageClass**: The storage class names of PVC and VolumeClaimTemplates can be automatically changed based on settings.

- **ImageRepo**: The repository address of the image used by the workload can be changed. The workload can be Deployment (including initContainer), StatefulSet, Orphaned Pod, Job, CronJob, Replica Set, Replication Controller, and DaemonSet.

k8clone Usage for Restoration

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 restore -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-f, --restore-conf**: path of **restore.json**. The default value is the directory where k8clone is located.
- **-d, --local-dir**: path for storing backup data. The default value is the directory where k8clone is located.

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
  k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

Procedure

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

Step 2 Prepare the data restoration configuration file **restore.json**.

Create a **restore.json** file, modify it based on the format, and place it in the directory where k8clone is located.

Example:

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

Step 3 Go to the directory where k8clone is located and run the restoration command to restore the backup data to the destination cluster.

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

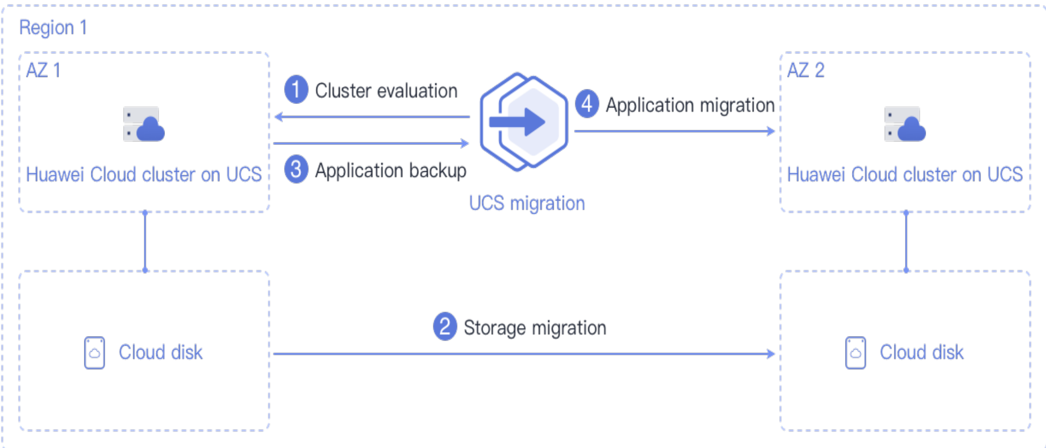
----End

10.6 Migration Across Huawei Cloud Clusters of UCS in the Same Region

10.6.1 Migration Process

Applications can be migrated between Kubernetes clusters managed by Huawei Cloud UCS in the same geographical region to meet management requirements such as better resource utilization and application upgrade. **Figure 10-15** shows the migration process.

Figure 10-15 Migration process



The process is as follows:

Step 1 **Cluster evaluation**

In this phase, you will evaluate the status of the source cluster to determine the type of the destination cluster. UCS kspider can automatically collect information about the source cluster, including the Kubernetes version, cluster scale, workload, and storage, and provide you with information about the recommended destination cluster. For details, see **Cluster Evaluation**.

Step 2 **Storage migration**

In this phase, you will migrate the EVS disk data to the destination AZ. For details, see **Storage Migration**.

Step 3 **Application backup**

In this phase, you will back up applications in the source AZ cluster. UCS k8clone can automatically collect Kubernetes metadata and save it as a compressed package to the local host to back up applications in the cluster. For details, see **Application Backup**.

Step 4 **Application migration**

In this phase, you will migrate applications from the source AZ cluster to the destination AZ cluster by restoring backup data. For details, see **Application Migration**.

----End

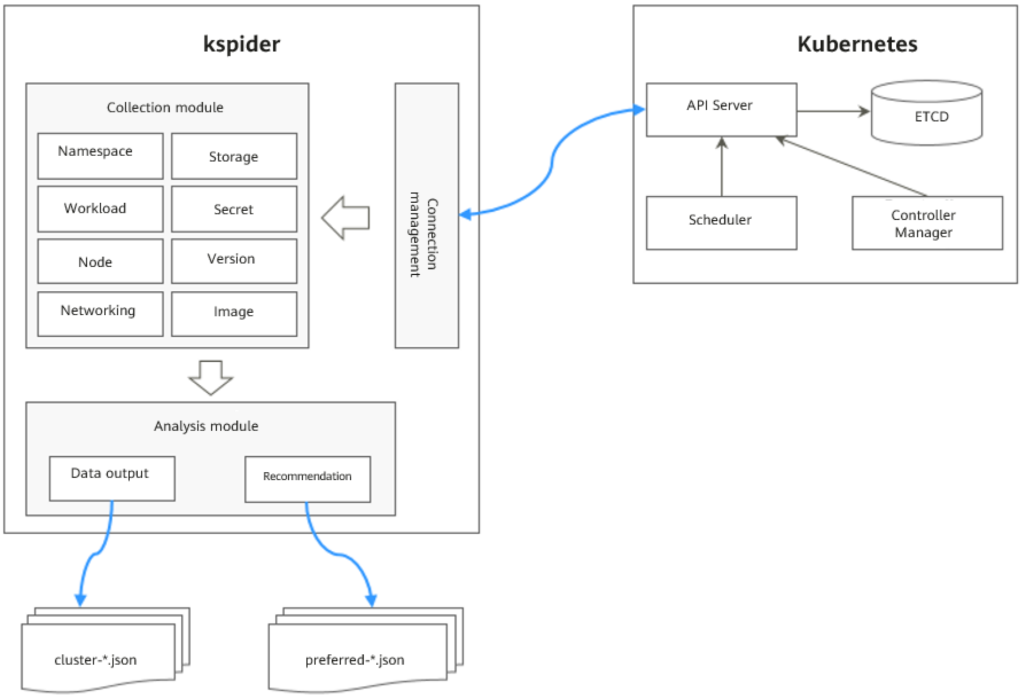
10.6.2 Cluster Evaluation

Migrating applications from one environment to another is a challenging task, so you need to plan and prepare carefully. kspider is a tool used to collect information about the source cluster. It provides cluster-related data such as the Kubernetes version, scale, workload quantity, storage, and in-use images. The data helps you understand the current status of the cluster and evaluate migration risks, and select a proper destination cluster version and scale.

How kspider Works

Figure 10-16 shows the architecture of kspider, which consists of three modules: collection, connection management, and analysis. The collection module can collect data of the source cluster, including namespaces, workloads, nodes, and networks. The connection management module establishes connections with the API Server of the source cluster. The analysis module aims to output the collected data of the source cluster (generating the **cluster-*.json** file) and provide the recommendation information of the destination cluster (generating the **preferred-*.json** file) after evaluation.

Figure 10-16 kspider architecture



Usage of kspider

NOTE

kspider can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **kspider-linux-amd64** in the following command with **kspider-linux-arm64** or **kspider-windows-amd64.exe**.

Prepare a server, upload kspider to the server, and decompress the tool package. For details, see **Preparations**. Run **./kspider-linux-amd64 -h** in the directory where kspider is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-n, --namespaces**: specifies the collected namespace. By default, system namespaces such as **kube-system**, **kube-public**, and **kube-node-lease** are excluded.
- **-q, --quiet**: indicates static exit.
- **-s, --serial**: specifies the unique sequence number of the output aggregation file (**cluster-{serial}.json**) and recommendation file (**preferred-{serial}.json**).

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]
```

```
Aliases:
  kspider, kspider

Flags:
  -h, --help                help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet               command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

Step 1: Collect Data from the Source Cluster

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

Step 2 Use the default parameter settings to collect data of all namespaces in the cluster. Run the **./kspider-linux-amd64** command.

Command output:

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
    Name    CPU    Memory    IP    Arch    OS    Kernel    MachineID
    10.1.18.64    4    8008284Ki    [10.1.18.64 10.1.18.64]    amd64    linux
3.10.0-1127.19.1.el7.x86_64    ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
    10.1.19.13    4    8008284Ki    [10.1.19.13 10.1.19.13]    amd64    linux
3.10.0-1127.19.1.el7.x86_64    2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
    Name    Namespace    NodeAffinity
    minio    default    false
    minio    minio    false
There are 3 Deployments:
    Name    Namespace    NodeAffinity
    rctest    default    true
    flink-operator-controller-manager    flink-operator-system    false
    rctest    minio    false
There are 1 DaemonSets:
    Name    Namespace    NodeAffinity
    ds-nginx    minio    false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
    Namespace/Name    Pods
    default/pvc-data-minio-0    default/minio-0
    minio/obs-testing    minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
    minio/pvc-data-minio-0    minio/minio-0
There are 5 PersistentVolumes:
    Name    Namespace    pvcName    scName    size    key
    pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8    minio-pv    obs-testing    csi-obs    1Gi    pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
    pvc-c7c768aa-373a-4c52-abea-e8b486d23b47    minio-pv    pvc-data-minio-0    csi-disk-sata    10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
    pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05    minio    obs-testing    csi-obs    1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
    pvc-9fd92c99-805a-4e65-9f22-e238130983c8    default    pvc-data-minio-0    csi-disk    10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
    pvc-a22fd877-f98d-4c3d-a04e-191d79883f97    minio    pvc-data-minio-0    csi-disk-sata    10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
    Name    Namespace    ServiceType
    headless-lxprus    default    ClusterIP
    kubernetes    default    ClusterIP
```

```
    minio      default      NodePort
    flink-operator-controller-manager-metrics-service      flink-operator-system      ClusterIP
    flink-operator-webhook-service      flink-operator-system      ClusterIP
    headless-lxprus      minio      ClusterIP
    minio      minio      NodePort
There are 0 Ingresses:
There are 6 Images:
    Name
    gcr.io/flink-operator/flink-operator:v1beta1-6
    flink:1.8.2
    swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
    nginx:stable-alpine-perl
    swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
    gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
    SecretType
    cfe/secure-opaque
    helm.sh/release.v1
```

After the kspider command is executed, the following files are generated in the current directory:

- **cluster-*.json**: This file contains data collected from the source cluster and applications. The data can be used to analyze and plan the migration.
- **preferred-*.json**: This file contains information about the recommended destination cluster. A preliminary evaluation is performed for the source cluster according to its scale and node specifications. The file provides suggestions on the version and scale of the destination cluster.

Step 3 View the data collected from the source cluster and applications.

You can use a text editor or JSON viewer to open the **cluster-*.json** file to view the data. Replace the asterisk (*) in the file name with the actual timestamp or serial number to find and open the correct file.

Description of the **cluster-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Namespaces: number of namespaces. The value is a string.
  Pods: total number of pods. The value is an integer.
  Nodes: node information. The IP address is used as the key to display node information.
    IP addresses
      CPU: CPU. The value is a string.
      Arch: CPU architecture. The value is a string.
      Memory: memory. The value is a string.
      HugePages1Gi: 1 GB hugepage memory. The value is a string.
      HugePages2Mi: 2 MB hugepage memory. The value is a string.
      OS: node OS. The value is a string.
      KernelVersion: OS kernel version. The value is a string.
      RuntimeVersion: running status and version of the node container. The value is a string.
      InternalIP: internal IP address. The value is a string.
      ExternalIP: external IP address. The value is a string.
      MachineID: node ID. The value is a string. Ensure that the CCE ID is the same as the ECS ID.
  Workloads: workload
    Deployment: workload type. The value can be Deployment, StatefulSet, DaemonSet, CronJob, Job, or
LonePod.
      default: namespace name
        Count: quantity. The value is an integer.
        Items: details. The value is an array.
          Name: workload name. The value is a string.
          Namespace: namespace name. The value is a string.
          NodeAffinity: node affinity. The value is of the Boolean type.
          Replicas: number of replicas. The value is an integer.
  Storage: storage
    PersistentVolumes: persistent volume
      pv-name: The PV name is used as the key.
```

```
        VolumeID: volume ID. The value is a string.
        Namespace: namespace. The value is a string.
        PvcName: name of the bound PVC. The value is a string.
        ScName: storage class name. The value is a string.
        Size: size of the space to request. The value is a string.
        Pods: name of the pod that uses the PV. The value is a string.
        NodeIP: IP address of the node where the pod is located. The value is a string.
        VolumePath: path of the node to which the pod is mounted. The value is a string.
    OtherVolumes: volumes of other types
      Type: AzureFile, AzureDisk, GCEPersistentDisk, AWSElasticBlockStore, Cinder, Glusterfs, NFS, CephFS,
FlexVolume, FlexVolume, DownwardAPI
        The volume ID, volume name, and volume shared path are keys.
        Pods: name of the pod. The value is a string.
          NodeIP: IP address of the node where the pod is located. The value is a string.
          Information that uniquely identifies a volume, such as the volume ID, volume name, and volume
shared path. The value is a string.
  Networks: network
    LoadBalancer: load balancing type
      service: network type, which can be Service or Ingress.
        Name: name. The value is a string.
        Namespace: namespace name. The value is a string.
        Type: type. The value is a string.
  ExtraSecrets: extended secret type
    Secret type. The value is a string.
  Images: image
    Image repo. The value is a string.
}
```

Example:


```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
      "ExternalIP": "100.85.xxx.xxx",
      "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
    }
  },
  "Workloads": {
    "CronJob": {},
    "DaemonSet": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-prometheus-node-exporter",
```

```
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 3
        }
      ]
    }
  },
  "Deployment": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-cost-analyzer",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    },
    "kubecost": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-kube-state-metrics",
          "Namespace": "kubecost",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  },
  "Job": {},
  "LonePod": {},
  "StatefulSet": {
    "minio-all": {
      "Count": 1,
      "Items": [
        {
          "Name": "minio",
          "Namespace": "minio-all",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  }
},
"Storage": {
  "PersistentVolumes": {
    "demo": {
      "VolumeID": "demo",
      "Namespace": "fluid-demo-test",
      "PvcName": "demo",
      "ScName": "fluid",
      "Size": "100Gi",
      "Pods": "",
      "NodeIP": "",
      "VolumePath": ""
    },
    "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
      "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
      "Namespace": "minio-all",
      "PvcName": "pvc-data-minio-0",
      "ScName": "csi-disk",
      "Size": "10Gi",
      "Pods": "minio-all/minio-0",
      "NodeIP": "10.1.23.159",
      "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
```

```
    }
   },
   "OtherVolumes": {}
  },
  "Networks": {
   "LoadBalancer": {}
  },
  "ExtraSecrets": [
   "cfe/secure-opaque",
   "helm.sh/release.v1"
  ],
  "Images": [
   "nginx:stable-alpine-perl",
   "ghcr.io/koordinator-sh/koord-manager:0.6.2",
   "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
   "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
   "gcr.io/kubecost1/cost-model:prod-1.91.0",
   "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```


----End

Step 2: Evaluate the Destination Cluster

After the kspider command is executed, in addition to the **cluster-*.json** file, the **preferred-*.json** file is also generated in the current directory. After performing preliminary evaluation for the source cluster according to its scale and node specifications, the file provides the recommended version and scale of the destination cluster. This helps you better plan and prepare for the migration.

Description of the **preferred-*.json** file:

```
{
  K8sVersion: Kubernetes version. The value is a string.
  Scale: cluster scale. The value is a string.
  Nodes: node information
    CPU: CPU. The value is a string.
    Memory: memory. The value is a string.
    Arch: CPU architecture. The value is a string.
    KernelVersion: OS kernel version. The value is a string.
    ProxyMode: cluster proxy mode. The value is a string.
  ELB: whether the ELB service is a dependent service. The value is of the Boolean type.
}
```

Evaluation rules for each field in the preceding file:

Table 10-7 Evaluation rules

Field	Evaluation Rule
K8sVersion	If the version is earlier than 1.21, the main release version of the UCS cluster (for example, 1.21, which changes over time) is recommended. If the version is later than the main release version, the latest version of the UCS cluster is recommended.

Field	Evaluation Rule
Scale	< 25 nodes in the source cluster: Destination cluster of 50 nodes is recommended. 25 ≤ Nodes in the source cluster < 100: Destination cluster of 200 nodes is recommended. 100 ≤ Nodes in the source cluster < 500: Destination cluster of 1000 nodes is recommended. Nodes in the source cluster ≥ 500: Destination cluster of 2000 nodes is recommended.
CPU/Memory	Statistics about the specification of the largest quantity are collected.
Arch	Statistics about the specification of the largest quantity are collected.
KernelVersion	Statistics about the specification of the largest quantity are collected.
ProxyMode	Configure this parameter according to the cluster scale. For a cluster with more than 1000 nodes, **ipvs** is recommended. For a cluster with fewer than 1000 nodes, **iptables** is recommended.
ELB	Check whether the source cluster has a load balancing Service.

Example:

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

CAUTION

The evaluation result is for reference only. You need to determine the version and scale of the destination cluster.

10.6.3 Storage Migration

If your cluster uses EVS disks, you need to migrate the EVS disks to the destination AZ together with the cluster. The migration method is as follows:

You can create an EVS disk backup using Cloud Backup and Recovery (CBR) and use the backup to create another EVS disk. When configuring the EVS disk information, select the destination AZ. For details, see **Creating a Cloud Disk Backup** and **Using a Backup to Create a Disk**.

10.6.4 Application Backup

Application migration across Huawei Cloud clusters of UCS in different AZs consists of two steps: application backup and application migration. This means applications in the source cluster are backed up and then migrated to the destination cluster through data restoration.

k8clone is an easy-to-use Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster.

NOTICE

Back up data during off-peak hours.

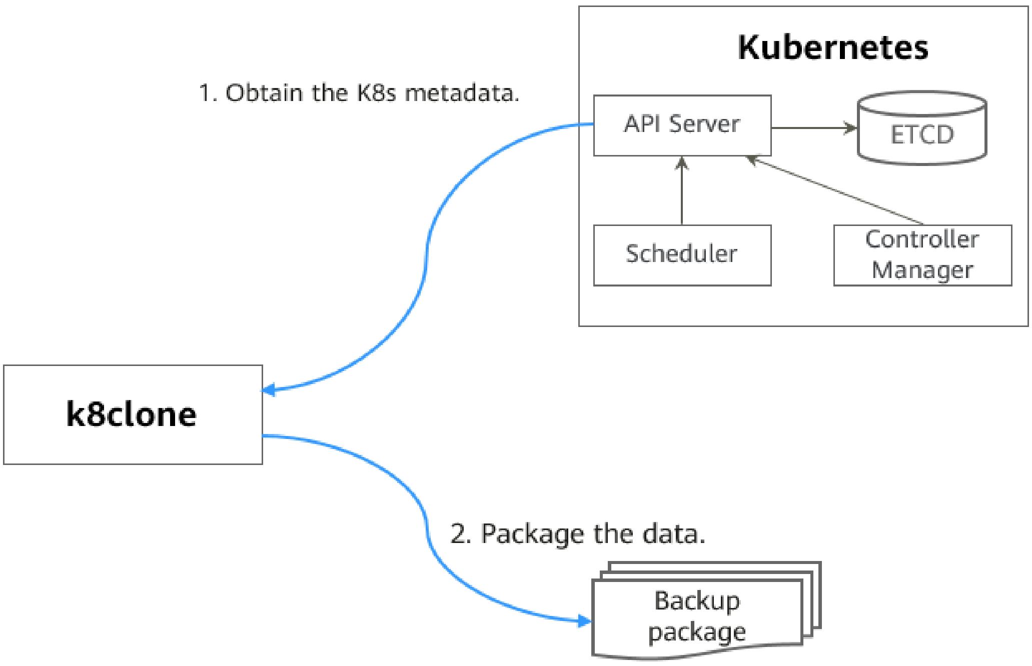
Prerequisites

Ensure that the storage data on which the cloud native application depends has been migrated.

How k8clone Backs Up Data

Data backup process:

Figure 10-17 Data backup process



k8clone Usage for Backup

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 backup -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-n, --namespace**: backs up cloud native applications of a specified namespace. Multiple namespaces are separated by commas (,), for example, ns1,ns2,ns3. The default value is **""**, indicating that the entire cluster is backed up.
- **-e, --exclude-namespaces**: excludes the backup of objects of a specified namespace. This parameter cannot be used together with **--namespace**.
- **-x, --exclude-kind**: excludes the backup of a specified resource type.
- **-i, --include-kind**: specifies the backup of a resource type.
- **-y, --exclude-object**: excludes the backup of a specified resource object.
- **-z, --include-object**: specifies the backup of a resource object.
- **-w, --exclude-having-owner-ref**: excludes the backup of resource objects with ownerReferences. The default value is **false**. The equal sign (=) must be contained in the parameter transfer process, for example, -w=true.
- **-d, --local-dir**: path for storing backup data. The default value is the **k8clone-dump** folder in the current directory.

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
  k8clone backup [flags]

Flags:
  -s, --api-server string             Kubernetes api-server url
  -q, --context string                Kubernetes configuration context
-w, --exclude-having-owner-ref      Exclude all objects having an Owner Reference. The following form is not permitted for boolean flags such as '-w false', please use '-w=false'
  -x, --exclude-kind strings          Resource kind to exclude. Eg. 'deployment'
  -i, --include-kind strings          Ressource kind to include. Eg. 'deployment'
  -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all namespaces and then filters them. Don't use it with the namespace flag.
  -y, --exclude-object strings        Object to exclude. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -z, --include-object strings        Object to include. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
  -h, --help                          help for backup
  -k, --kubeconfig string             The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
```

```
-d, --local-dir string          Where to dump yaml files (default "./k8clone-dump")
-n, --namespace string            Only dump objects from this namespace
```

Examples:

- Backs up objects of the entire cluster. The default path is the **k8clone-dump** folder in the current directory.

  ./k8clone-linux-amd64 backup
- Backs up objects of the entire cluster and specifies the path for storing backup data.

  ./k8clone-linux-amd64 backup -d ./xxxx
- Backs up objects of a specified namespace.

  ./k8clone-linux-amd64 backup -n default
- Excludes the backup of objects of a specified namespace.

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease
- Excludes the backup of specified resource types.

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice
- Specifies the backup of resource types.

  ./k8clone-linux-amd64 backup -x rolebinding
- Excludes the backup of specified resource objects.

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns
- Specifies the backup of resource objects.

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns
- Excludes the backup of resource objects with ownerReferences.

  ./k8clone-linux-amd64 backup -w=true

Procedure

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

Step 2 Go to the directory where k8clone is located and run the backup command to back up data to a local directory and compress the data into a package.

The examples in **k8clone Usage for Backup** provide several common backup methods. You can select a method as required or customize one.

----End

10.6.5 Application Migration

Application migration across Huawei Cloud clusters of UCS in different AZs consists of two steps: application backup and application migration. This means applications in the source cluster are backed up and then migrated to the destination cluster through data restoration.

k8clone is an easy-to-use Kubernetes metadata cloning tool. It can save Kubernetes metadata (objects) as a local package and restore the metadata to the destination cluster.

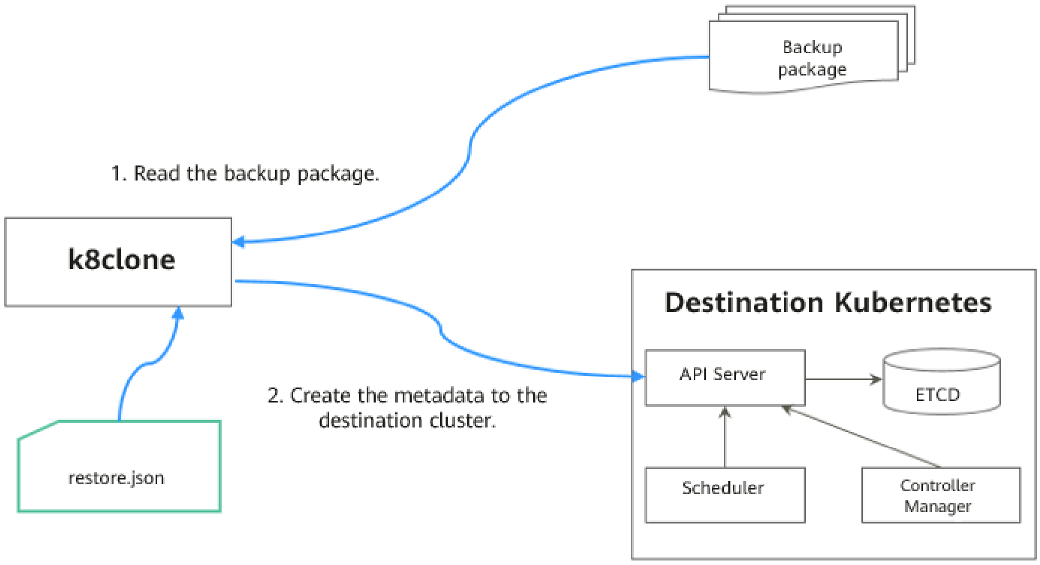
Prerequisites

- Ensure that the storage data on which the cloud native application depends has been migrated.
- Ensure that the metadata backup in the source cluster has been downloaded to the server where k8clone is executed.

How k8clone Restores Data

Data restoration process:

Figure 10-18 Data restoration process



Before the restoration, prepare a data restoration configuration file **restore.json** to automatically change the storage class names of PVC and StatefulSet and the repository address of the image used by the workload during application restoration.

The file content is as follows:

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  // The StorageClassName field of PVC and
StatefulSet can be changed.
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-
north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- **StorageClass**: The storage class names of PVC and VolumeClaimTemplates can be automatically changed based on settings.
- **ImageRepo**: The repository address of the image used by the workload can be changed. The workload can be Deployment (including initContainer), StatefulSet, Orphaned Pod, Job, CronJob, Replica Set, Replication Controller, and DaemonSet.

k8clone Usage for Restoration

NOTE

k8clone can run on Linux (x86 and Arm) and Windows. The usage is similar in both environments. This section uses the Linux (x86) environment as an example.

If Linux (Arm) or Windows is used, replace **k8clone-linux-amd64** in the following command with **k8clone-linux-arm64** or **k8clone-windows-amd64.exe**.

Run **./k8clone-linux-amd64 restore -h** in the directory where k8clone is located to learn about its usage.

- **-k, --kubeconfig**: specifies the location of the kubeconfig file of kubectl. The default value is **$HOME/.kube/config**. The kubeconfig file is used to configure access to the Kubernetes cluster. The kubeconfig file contains the authentication credentials and endpoints (access addresses) required for accessing and registering the Kubernetes cluster. For details, see the **Kubernetes documentation**.
- **-s, --api-server**: Kubernetes API Server URL. The default value is **""**.
- **-q, --context**: Kubernetes Configuration Context. The default value is **""**.
- **-f, --restore-conf**: path of **restore.json**. The default value is the directory where k8clone is located.
- **-d, --local-dir**: path for storing backup data. The default value is the directory where k8clone is located.

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
  k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

Procedure

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

Step 2 Prepare the data restoration configuration file **restore.json**.

Create a **restore.json** file, modify it based on the format, and place it in the directory where k8clone is located.

Example:

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
```

```
 }
}
```

Step 3 Go to the directory where k8clone is located and run the restoration command to restore the backup data to the destination cluster.

Example:

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

----End

11 Pipeline

11.1 Overview

CodeArts Pipeline provides automated release management from building to rollout for UCS container fleets in multi-cloud scenarios. It helps you develop an overall, agile, and efficient application delivery solution.

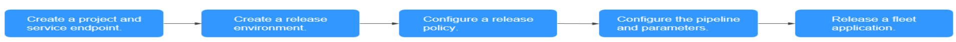
Using pipelines to release container fleets makes it easier to release applications across clouds in a scenario where public, private, and edge clouds coexist.

Prerequisites

You have created a UCS container fleet and enabled cluster federation for the fleet. If not, enable it by referring to **Enabling Cluster Federation**.

Pipeline Release Process

Figure 11-1 Pipeline release process



The pipeline release process is shown in **Figure 11-1**. The details are as follows:

Step 1 Create a project and service endpoint. In this section, you will create a pipeline project for the application and configure cross-service permissions for the project.

Step 2 Create a release environment. In this section, you will create a new code repository for the application and configure the release environment and associated cluster fleets.

Step 3 Configure a release policy. In this section, you will configure an application release policy based on the preset release template.

Step 4 Configure the pipeline and parameters. In this section, you will graphically orchestrate the release process, and select the environment level, release environment, and artifact path through the release plug-in.

Step 5 Release a fleet application. In this section, you will use the pipeline to automate the whole process from building source code and to releasing the application.

----End

11.2 Creating a Project and Service Endpoint

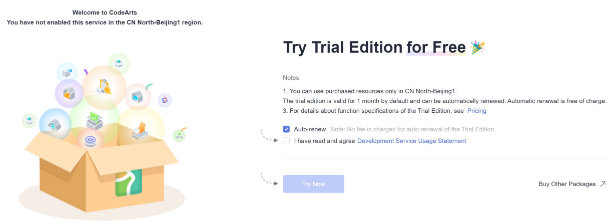
This section describes how to create a pipeline project for an application and how to configure cross-service permissions for the project.

Creating a Scrum Project

Step 1 Log in to the UCS console. In the navigation pane, choose **CICD** > **Pipeline**.

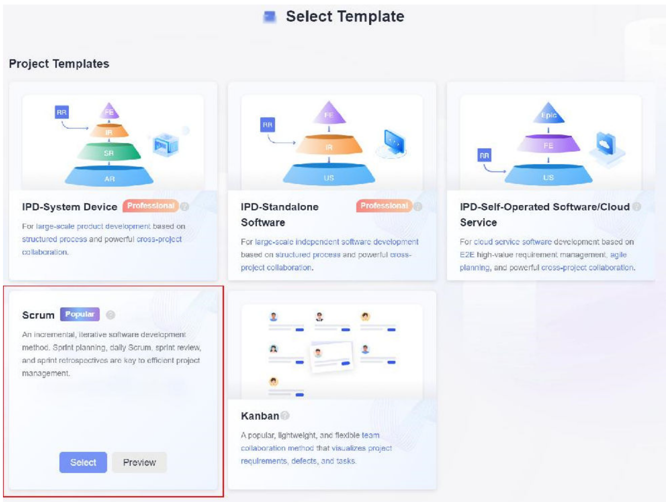
Step 2 Click **Start building your first container fleet pipeline project**. On the displayed CodeArts page, click **Try Now**.

Figure 11-2 Enabling CodeArts



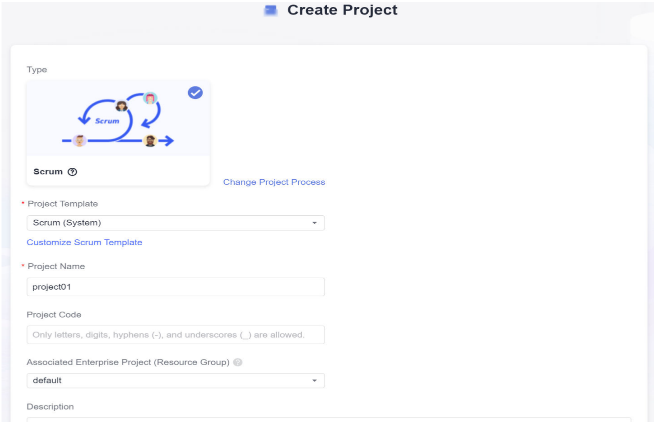
Step 3 Click **Create Project**, select a Scrum project template, and click **Select**.

Figure 11-3 Selecting the Scrum project



Step 4 Enter the project name and other information to create a Scrum project. After the project is created, the Scrum project homepage will be displayed.

Figure 11-4 Creating a Scrum project



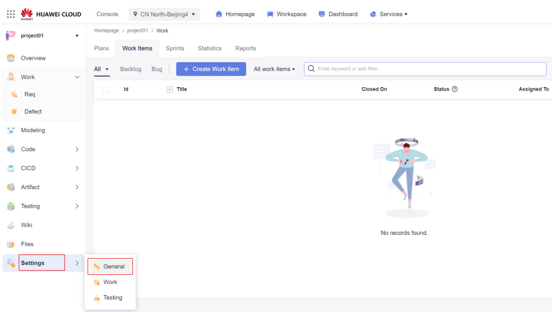
----End

Configuring a Service Endpoint

By creating a service endpoint and configuring IAM authentication information for the account, you can obtain the UCS fleet information of the same cloud service account and cross accounts from CodeArts. This helps you interconnect cross-service permissions and release applications. The procedure for configuring a service endpoint is as follows:

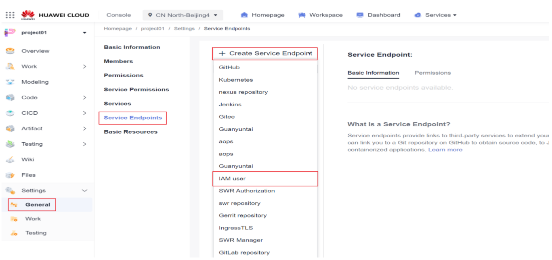
Step 1 In the navigation pane on the left, choose **Settings** > **General**.

Figure 11-5 Choosing general settings



Step 2 Click **Service Endpoints**, click **Create Service Endpoint**, and select **IAM user** from the drop-down list.

Figure 11-6 Configuring a service endpoint



Step 3 Configure IAM information for the service endpoint. For details, see **Table 11-1**.

Figure 11-7 Creating a service endpoint

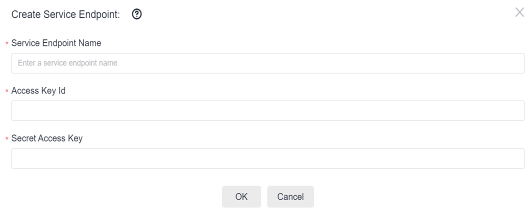


Table 11-1 Parameters for configuring IAM information

Parameter	Description
Service Endpoint Name	Name of the service endpoint This parameter can be customized. Here **iam01** is used as an example.
Access Key Id	The ID of an access key. For details about how to obtain the access key ID, see **How Do I Obtain an Access Key (AK/SK)?**
Secret Access Key	Secret access key. For details about how to obtain the secret access key, see **How Do I Obtain an Access Key (AK/SK)?**

----End

11.3 Creating a Release Environment

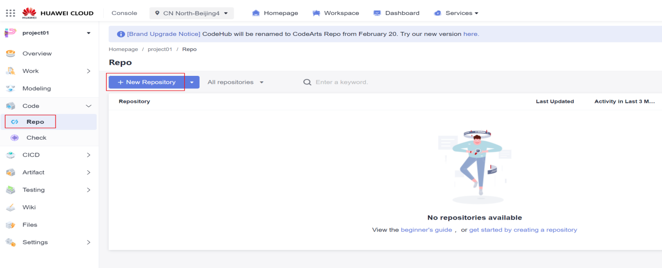
This section describes how to create a code repository for an application and configure the environment and associated UCS cluster fleets.

Creating a Code Repository

Step 1 On the Scrum project homepage, search for the Scrum project created in **Creating a Scrum Project**, and click the project name to access the project.

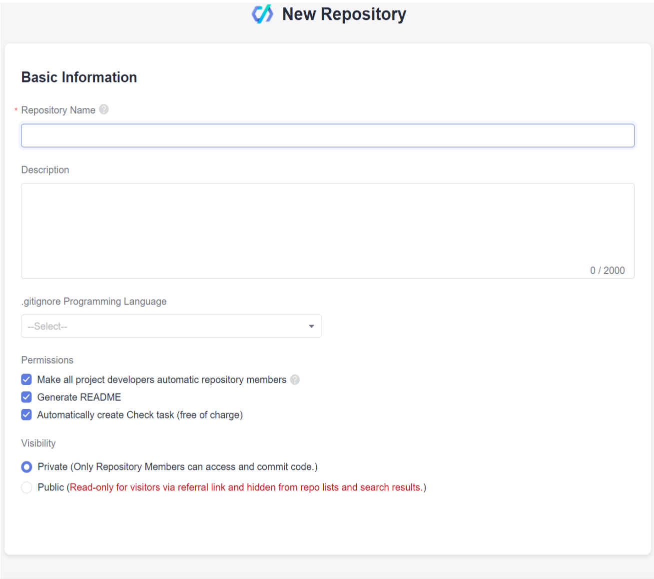
Step 2 In the navigation pane on the left, choose **Code** > **Repo**, and click **New Repository**.

Figure 11-8 Creating a common repository



Step 3 Configure the repository name, permissions, and visibility. For details, see **Figure 11-9**.

Figure 11-9 Configuring the code repository

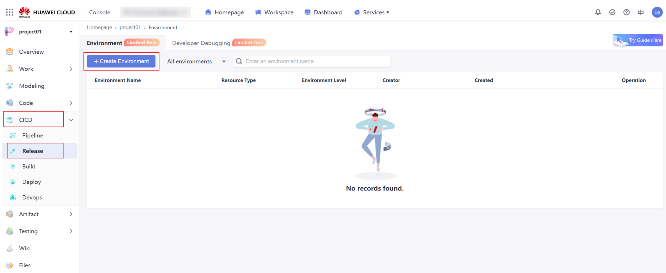


----End

Creating an Environment

Step 1 In the navigation pane on the left, choose **CICD** > **Release**, and click **Environment**. On the displayed page, you can view all environments.

Figure 11-10 Viewing the environments



Step 2 Click **Create Environment** and configure basic information. For details about the parameters, see **Table 11-2**.

NOTICE

- If you set the current user as the publish user, you can directly obtain the UCS fleet information of the account.
- If you set other users as the publish users, you can obtain the UCS fleet information of the account through the IAM service endpoint configured in **Creating a Project and Service Endpoint**.

Figure 11-11 Setting the current user as the publish user

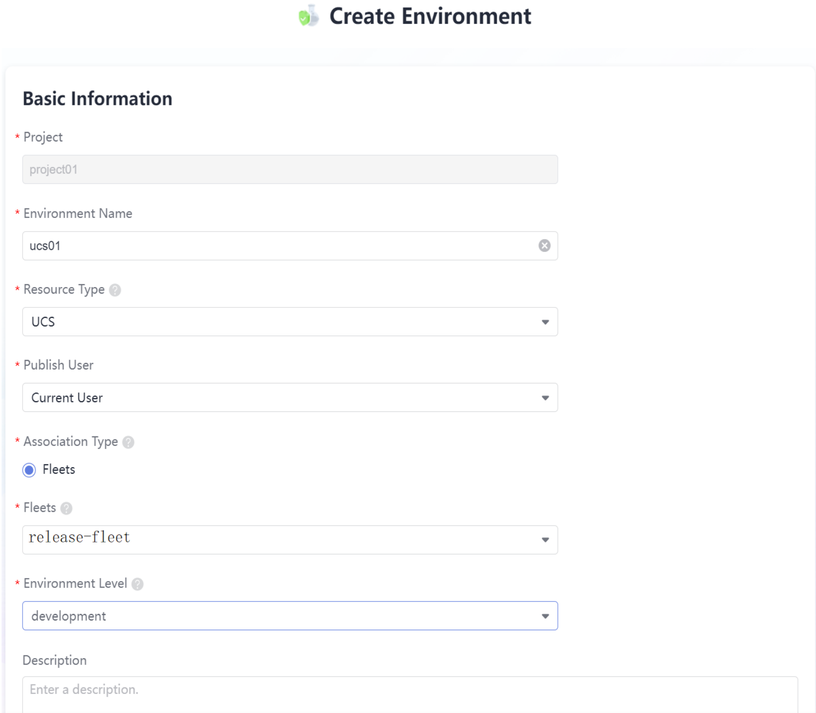


Figure 11-12 Setting other users as the publish users

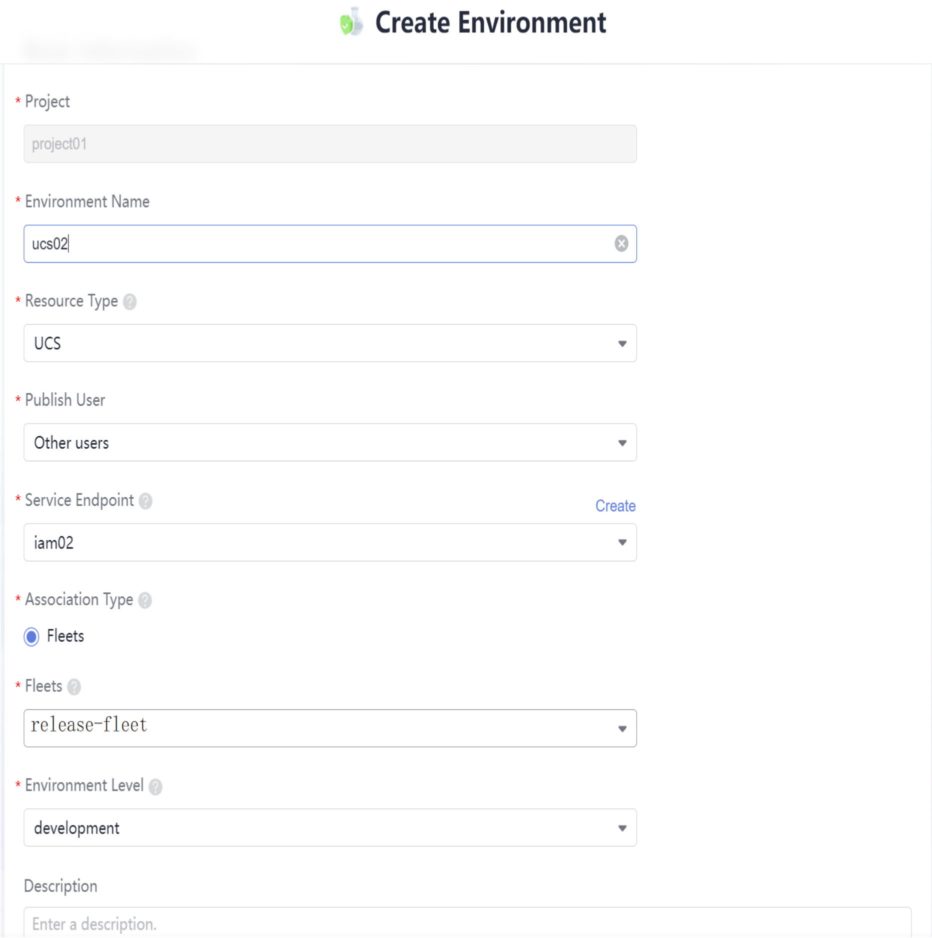


Table 11-2 Parameters for creating an environment

Parameter	Description
Environment Name	Unique ID of an environment. Once created, this parameter cannot be changed.
Resource Type	Multiple types of resources that support different deployment plug-ins.
Publish User	The current user or other users to obtain the fleet information of the account to release applications.
Service Endpoint	Endpoint to obtain UCS resource permissions. For details about how to create a service endpoint, see **How Do I Obtain an Access Key (AK/SK)?**.
Association Type	Associated UCS resource granularity. Currently, only container fleets are supported.

Parameter	Description
Fleets	Cluster fleets for which cluster federation is enabled in the UCS.
Environment Level	Environment types. There are four environment types: development, test, pre-production, and production.
Description	Description of the environment. This parameter is optional.

Step 3 Click **OK**. The environment details page is displayed.

----End

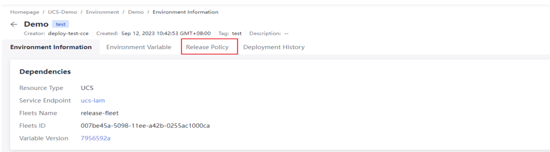
11.4 Configuring a Release Policy

The rolling upgrade template is preset in release management. This section describes how to add the rolling upgrade plug-in and configure a release policy by using the preset rolling upgrade template.

Currently, the UCS pipeline only presets the rolling upgrade template.

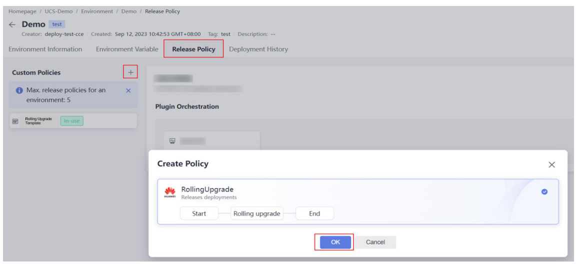
Step 1 On the environment details page, click **Release Policy**.

Figure 11-13 Release policy



Step 2 On the right of **Custom Policies**, click +. In the displayed dialog box, select a policy template as needed, and click **OK**.

Figure 11-14 Creating a policy



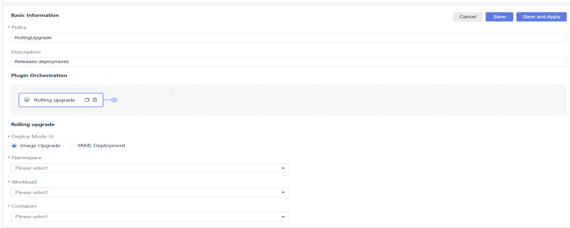
Step 3 Configure basic information and add plug-ins to customize the template.

There are two deployment modes for rolling upgrade: image upgrade and YAML deployment.

Image Upgrade

When you select image upgrade, you need to select the related namespace, workload, and container. During deployment, the pipeline will change the image to the container image of the workload of the namespace.

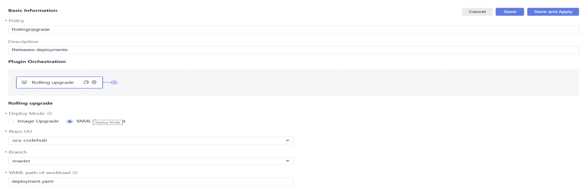
Figure 11-15 Image upgrade



YAML Deployment

You need to create a YAML file in the code repository and enter the YAML path of workload.

Figure 11-16 YAML deployment

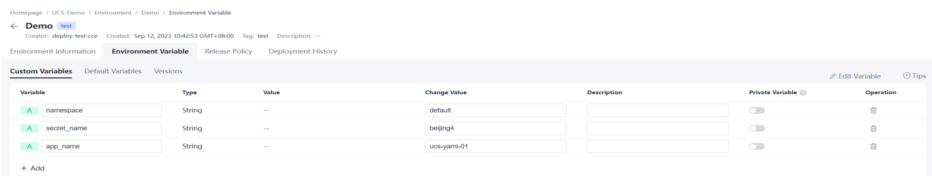


If a private image is pulled, perform the following steps:

In UCS, configure a Docker image repository key for the corresponding cluster and record the key name. For details, see **Secrets**.

Choose **Release** > **Environment** > **Environment Variable** to set environment variables. You can reference environment variables in the format of **{{}}** in YAML files.

Figure 11-17 Configuring environment variables



Example YAML file:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: {{app_name}}
  namespace: {{namespace}}
spec:
  replicas: 3
  selector:
    matchLabels:
      app: {{app_name}}
      version: v1
```

```
template:
  metadata:
    labels:
      app: {{app_name}}
      version: v1
  spec:
    containers:
      - name: container-1
        image: {{ARTIFACT}}
        env:
          - name: PAAS_APP_NAME
            value: {{app_name}}
          - name: PAAS_NAMESPACE
            value: {{namespace}}
          - name: PAAS_PROJECT_ID
            value: {{PROJECT_ID}}
        resources:
          limits:
            cpu: 250m
            memory: 512Mi
          requests:
            cpu: 250m
            memory: 512Mi
    imagePullSecrets:
      - name: {{secret_name}}
    schedulerName: default-scheduler
```

NOTE

- YAML deployment only supports one YAML file.
- The code repositories and their branches of YAML files are those configured in release management.
- The YAML path is a relative path. The current directory is the root directory of the code branch.
- You can use *${variable name}* in a YAML path to reference an environment variable, and *{{variable name}}* in a YAML file to reference an environment variable.

----End

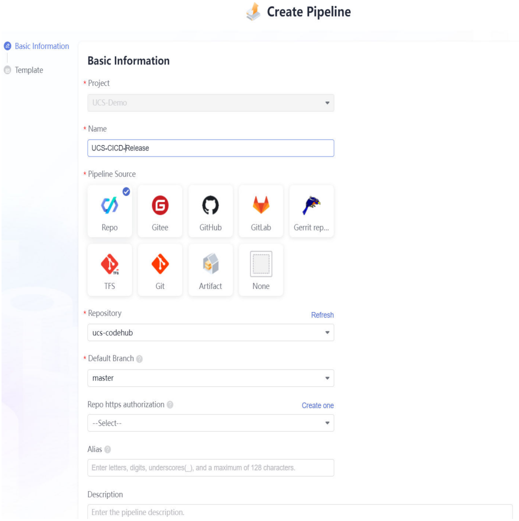
11.5 Configuring the Pipeline and Parameters

This section describes how to graphically orchestrate the release process and how to select the environment level, release environment, and artifact path through the release plug-in.

Step 1 In the navigation pane, choose **CICD** > **Pipeline**. The Pipeline page is displayed.

Step 2 Click **Create Pipeline** and select the code repository created in **Creating a Code Repository**.

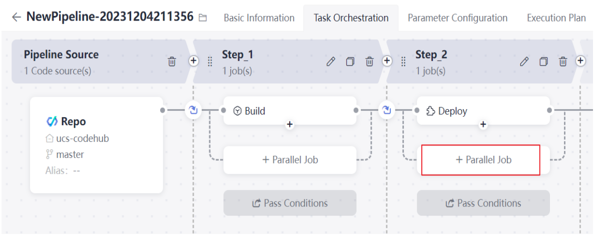
Figure 11-18 Creating a pipeline



Step 3 Click **Next**. Select **Get-Started** from system templates. The task orchestration page is displayed.

Step 4 Configure the stage name based on the service requirements, and configure the execution content and orchestration details of each job.

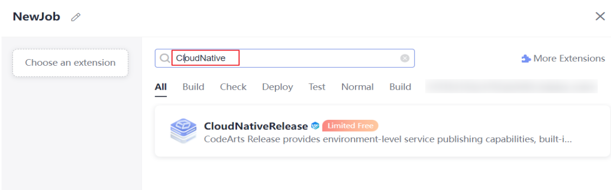
Figure 11-19 Task orchestration



In task orchestration, set the name of the **Build** stage to **Step_1** and the job type to **Build**. This stage is to build an image for deploying an application based on the application source code. For details, see **Using Node.js to Create a Docker Image**.

Set the name of the rightmost stage to **Step_2** and the job type to **Cloud Native Release**. This stage is to deploy the application to the UCS fleet based on the defined delivery resources of the YAML file.

Figure 11-20 Adding a cloud native release job

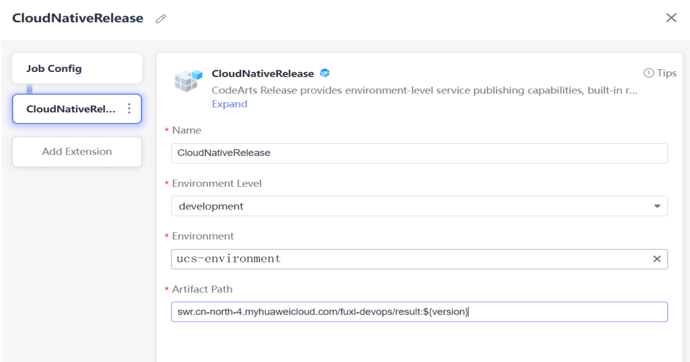


Step 5 Choose **CICD** > **Pipeline**, select the extension to be released, and configure the environment level, environment, and artifact path.

The artifact path refers to the image generated by compiling source code in **Step_1 Build** configured in **Step 4** and pushed to the SoftWare Repository for

Container (SWR). When configuring the artifact path, you can directly enter the artifact path and the version number of the referenced image, or use environment variables in the format of **${*variable name*}** to reference the built artifact.

Figure 11-21 Setting a cloud native release job

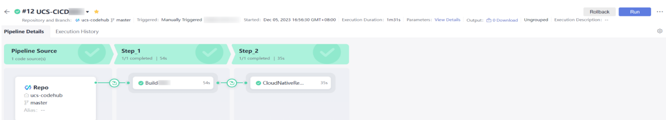


Step 6 Modify the YAML file of workload by referencing the default ARTIFACT variable in the **image** field. The artifact path is rendered to the **image** field of the YAML file of workload through the default ARTIFACT variable.

```
image:{{ARTIFACT}}
```

After the pipeline is configured, the pipeline details page is displayed, as shown in **Figure 11-22**.

Figure 11-22 Pipeline configured successfully



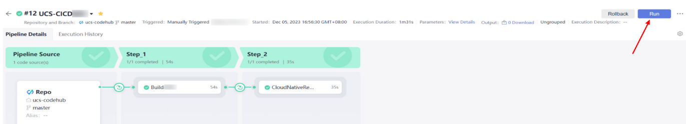
----End

11.6 Releasing a Fleet Application

This section describes how to use a pipeline to automate the whole process from building source code and to releasing the application.

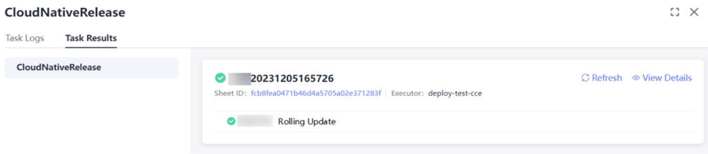
Step 1 After the pipeline, parameters, and artifact path are configured, click **Run** to execute the pipeline to build code and implement cloud native release.

Figure 11-23 Executing the pipeline



Step 2 Click **CloudNativeRelease** of stage 2. In the displayed dialog box, click **Task Results** to view the release sheet.

Figure 11-24 Viewing the release ticket

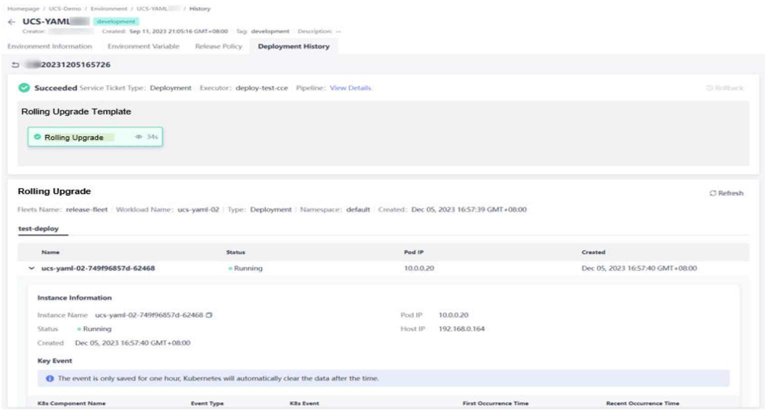


- The basic information about the release is displayed, including the ticket name, ticket ID, and the release task step.
- On the details page, the release process is displayed. You can view the running status of a specified workload in each cluster of the current container fleet, and retry or cancel the release.

Step 3 Click **View Details** to go to the details page of the service ticket.

On this page, you can view the release process and details of each cluster in the fleet. You can also view the instance information, creation time, Kubernetes events, and more of the workload in Huawei Cloud clusters, attached clusters, and on-premises clusters of the current container fleet.

Figure 11-25 Service ticket details



Step 4 After the fleet application is released, log in to the UCS console and choose **Fleets** in the navigation pane. On the displayed page, click the cluster name to go to the details page. Choose **Workloads** in the navigation pane and check whether the Deployment has been released to the corresponding cluster and is running normally.

----End

12 Error Codes

If an exception occurs during the execution of an operation request and the request is not processed, an error message is returned. The error message contains the error code and error description.

Table 12-1 Error code description

Error Code	Status Code	Message	Description
UCS.00000001	400	Failed to obtain the user information.	Obtain user information failed.
UCS.00000003	400	Failed to obtain the federation information.	Obtain the federation information failed.
UCS.00000004	403	Request forbidden.	Forbidden request.
UCS.00000005	500	Database operation failed.	Database error.
UCS.00000006	500	Server internal error.	Internal server error.
UCS.00000007	500	Data transform error.	Data conversion failed.
UCS.00000008	500	Error add event.	Add the event failed.
UCS.00000009	500	Data unmarshal error.	Deserialize data failed.
UCS.00000010	500	Data marshal error.	Serialize data failed.
UCS.00000011	400	Bad query parameter value.	Invalid request parameter.

Error Code	Status Code	Message	Description
UCS.00000012	400	Invalid request body.	Invalid request body.
UCS.00000013	404	No requested resources found.	The requested resource cannot be found.
UCS.00000014	500	Failed to encrypt data.	Data encryption failed.
UCS.00000015	500	Failed to decrypt data.	Data decryption failed.
UCS.00000016	400	Invalid header value.	Invalid request header.
UCS.00000017	400	Insufficient quota	Insufficient quota.
UCS.00000018	401	Authorization failed.	Authorization failed.
UCS.00010001	500	Failed to get iam connection.	IAM connection failed.
UCS.00010002	403	Sub-user has no authority to create agency.	The sub-user does not have the permission for creating an agency.
UCS.00010003	400	Failed to create agency.	Create agency failed.
UCS.00010004	500	Failed to get role id for te_admin.	Obtain the te_admin role failed.
UCS.00010005	500	Failed to get admin token from iam.	Obtain the admin token failed.
UCS.00010006	500	Failed to get agency list from iam.	Obtain the agency list failed.
UCS.00010007	500	Failed to get agency grants from iam.	Obtain the grants agency failed.
UCS.00010008	500	Failed to update agency role.	Update the role agency failed.
UCS.00010009	400	Failed to get project token by agency	Obtain the project token through the agency failed.

Error Code	Status Code	Message	Description
UCS.00010010	400	Failed to get op_svc account domain token	Obtain the token of the op account failed.
UCS.00010011	400	Failed to get project id by project name.	Obtain the project ID failed.
UCS.00010012	400	IAM agency quota insufficient, please expand agency quota	IAM agency quota exceeded.
UCS.00010013	400	fail to get iam pdp authorize result	Obtain the PDP authentication result failed.
UCS.00010014	403	iam pdp authentication denied	PDP authentication rejected.
UCS.00010015	403	iam rbac authentication denied	RBAC authentication rejected.
UCS.00020001	500	Failed to get aeskey.	Obtain the aeskey failed.
UCS.00020002	500	Failed to get certs.	Obtain certificate failed.
UCS.00020003	500	Failed to create certs.	Create certificate failed.
UCS.00020003	500	Failed to delete certs.	Delete certificate failed.
UCS.00030001	404	Cluster Not Found.	No clusters found.
UCS.00030002	400	Failed to obtain the cluster information.	Obtain the cluster information failed.
UCS.00030003	400	Failed to get resourceJob info with cluster status	Obtain the resource job failed.
UCS.00040001	400	Failed to obtain the mesh information.	Obtain the mesh information failed.
UCS.00090001	500	Failed to create DNSRecord	Record set creation failed.
UCS.00100001	400	Failed to publish message to smn.	Publish messages to SMN failed.
UCS.00100002	400	smn topic error.	Incorrect SMN topic.

Error Code	Status Code	Message	Description
UCS.00100003	400	smn subscription error.	SMN subscription error.
UCS.00110001	400	SDR failed to get billing raw data	Obtain billing data failed.
UCS.00110002	400	Formatting raw billing data to SDR format error	Format billing data failed.
UCS.00120001	400	CBC failed to update resources status	Update the CBC resource status failed.
UCS.00130001	400	Get UCS Agency info error	Obtain the UCS agency failed.
UCS.00140001	400	Create ClusterRole failed	Create a ClusterRole failed.
UCS.00140002	400	Delete ClusterRole failed	Delete a ClusterRole failed.
UCS.00140003	400	Update ClusterRole failed	Update a ClusterRole failed.
UCS.00140004	400	Get ClusterRole failed	Obtain the ClusterRole information failed.
UCS.00140005	400	Create ClusterRoleBinding failed	Create a ClusterRoleBinding failed.
UCS.00140006	400	Delete ClusterRoleBinding failed	Delete a ClusterRoleBinding failed.
UCS.00140007	400	Update ClusterRoleBinding failed	Update a ClusterRoleBinding failed.
UCS.00140008	400	Get ClusterRoleBinding failed	Obtain the ClusterRoleBinding information failed.
UCS.00140009	400	Create Role failed	Create a role failed.
UCS.00140010	400	Delete Role failed	Delete a role failed.
UCS.00140011	400	Update Role failed	Update a role failed.

Error Code	Status Code	Message	Description
UCS.00140012	400	Get Role failed	Obtain role information failed.
UCS.00140013	400	Create RoleBinding failed	Create a RoleBinding failed.
UCS.00140014	400	Delete RoleBinding failed	Delete a RoleBinding failed.
UCS.00140015	400	Update RoleBinding failed	Update a RoleBinding failed.
UCS.00140016	400	Get RoleBinding failed	Obtain the RoleBinding information failed.
UCS.00150001	400	Cluster policy validate failed.	Cluster policy verification failed.
UCS.00150002	400	ClusterGroup policy validate failed.	Cluster group policy verification failed.
UCS.00150003	400	Cluster has enable policy.	Policy enabled for the cluster.
UCS.00150004	400	ClusterGroup has enable policy.	Policy enabled for the cluster group.
UCS.00150005	400	Cluster not enable policy.	Policy not enabled for the cluster.
UCS.00150006	400	ClusterGroup not enable policy.	Policy not enabled for the cluster group.
UCS.00150007	500	Get policy job failed.	Obtain the policy task failed.
UCS.01000001	400	Failed to obtain the user information.	Obtain the user information failed.
UCS.01000002	429	The throttling threshold has been reached.	Throttling threshold reached.
UCS.01000003	401	Authorization failed.	Authorization failed.
UCS.01000004	403	Request forbidden.	Forbidden request.
UCS.01000005	500	Database operation failed.	Database error.
UCS.01000006	500	Server internal error.	Internal server error.

Error Code	Status Code	Message	Description
UCS.01000007	500	Data transform error.	Data conversion failed.
UCS.01000008	500	Error add event.	Add the event failed.
UCS.01000009	500	Data unmarshal error.	Deserialize data failed.
UCS.01000010	500	Data marshal error.	Serialize data failed.
UCS.01000011	400	Bad query parameter value.	Invalid request parameter.
UCS.01000012	400	Invalid request body.	Invalid request body.
UCS.01000013	404	No requested resources found.	The requested resource cannot be found.
UCS.01000014	500	Failed to encrypt data.	Data encryption failed.
UCS.01000015	500	Failed to decrypt data.	Data decryption failed.
UCS.01000016	400	Invalid header value.	Invalid request header.
UCS.01000017	400	Insufficient quota	Insufficient quota.
UCS.01000018	400	Quota info validate failed	Quota parameter verification failed.
UCS.01000019	500	Quota update failed	Quota update failed.
UCS.01010001	500	Failed to get iam connection.	IAM connection failed.
UCS.01010002	500	Failed to get project token by agency	Obtain the project token through the agency failed.
UCS.01010003	403	No access permission. Please contact the administrator.	No permissions.
UCS.01010004	400	get deployment region's projectID error	Obtain the project ID failed.
UCS.01010005	400	get IAM agency's token error	Obtain the agency token failed.

Error Code	Status Code	Message	Description
UCS.01010006	400	fail to get iam pdp authorize result	Obtain the PDP authentication result failed.
UCS.01010007	403	iam pdp authentication denied	PDP authentication rejected.
UCS.01010008	403	iam rbac authentication denied	RBAC authentication rejected.
UCS.01020001	500	Failed to get aeskey.	Obtain the aeskey failed.
UCS.01020002	500	Failed to get certs.	Obtain certificate failed.
UCS.01020003	500	Failed to create certs.	Create certificate failed.
UCS.01020004	500	Failed to delete certs.	Delete certificate failed.
UCS.01030001	404	Cluster Not Found.	No clusters found.
UCS.01030002	400	Failed to obtain the cluster information.	Obtain the cluster information failed.
UCS.01030003	409	The same cluster already exists.	The cluster name already exists.
UCS.01030004	400	Cluster status is unavailable, please fix cluster first.	Cluster status is Unavailable.
UCS.01030005	403	No authorization for cluster	Authorize the cluster failed.
UCS.01030006	400	Create resource job for cluster error	Create a resource job in the cluster failed.
UCS.01030007	400	Create on-demand order for cluster error	Create the pay-per-use order failed.
UCS.01030008	400	Cluster kubeconfig format error.	Incorrect kubeconfig format of the cluster.
UCS.01030009	400	This cluster does not support unregister	The cluster does not support unregistration.

Error Code	Status Code	Message	Description
UCS.01030010	400	Failed to obtain cce cluster information.	Obtain the CCE cluster information failed.
UCS.01030011	400	Cluster category not supported	Cluster type not supported.
UCS.01030012	400	Register cce cluster error	CCE cluster registration failed.
UCS.01030013	400	Register attached cluster error	Attached cluster registration failed.
UCS.01030014	400	Register on-premise cluster error	On-premises cluster registration failed.
UCS.01030015	100	Register multi cloud cluster error	Multi-cloud cluster registration failed.
UCS.01030016	400	Cluster has been frozen	Cluster frozen.
UCS.01050001	400	RecordSet create failed.	Record set creation failed.
UCS.01080001	400	Failed to obtain the federation information.	Obtain the federation information failed.
UCS.01080002	400	Cluster group has federalized.	Federation enabled for the fleet.
UCS.01080003	500	Cluster group federation failed.	Federation operation failed.
UCS.01080004	400	Cluster group federation validate failed.	Enable federation verification failed.
UCS.01080005	400	Retry join all clusters to federation failed.	Retry federating all clusters failed.
UCS.01080006	400	Cluster group has not been federalized.	Federation not enabled for the fleet.
UCS.01080007	400	Retry join cluster to federation failed.	Retry federating the cluster failed.
UCS.01090001	400	Failed to obtain the mesh information.	Obtain the mesh information failed.
UCS.01100001	403	No authorization for cluster group	Fleet unauthorized.

Error Code	Status Code	Message	Description
UCS.01100002	400	associate cluster with clustergroup error	Add the cluster to the fleet failed.
UCS.01100003	400	associate cluster with rule error	Associate the permission policy with the fleet failed.
UCS.01100004	409	The same clustergroup already exists.	The fleet name already exists.
UCS.01100005	404	ClusterGroup Not Found.	The fleet does not exist.
UCS.01100006	400	Cluster number in fleet exceed limit.	Too many clusters in the fleet.
UCS.01100007	400	Update associated clusters validate failed	Verify the update of the associated cluster failed.
UCS.01110001	400	resource notification to SMN error	Send notifications to SMN failed.
UCS.01120001	400	Create ClusterRole failed	Create a ClusterRole failed.
UCS.01120002	400	Delete ClusterRole failed	Delete a ClusterRole failed.
UCS.01120003	400	Update ClusterRole failed	Update a ClusterRole failed.
UCS.01120004	400	Get ClusterRole failed	Obtain the ClusterRole information failed.
UCS.01120005	400	Create ClusterRoleBinding failed	Create a ClusterRoleBinding failed.
UCS.01120006	400	Delete ClusterRoleBinding failed	Delete a ClusterRoleBinding failed.
UCS.01120007	400	Update ClusterRoleBinding failed	Update a ClusterRoleBinding failed.
UCS.01120008	400	Get ClusterRoleBinding failed	Obtain the ClusterRoleBinding information failed.
UCS.01120009	400	Create Role failed	Create a role failed.

Error Code	Status Code	Message	Description
UCS.01120010	400	Delete Role failed	Delete a role failed.
UCS.01120011	400	Update Role failed	Update a role failed.
UCS.01120012	400	Get Role failed	Obtain role information failed.
UCS.01120013	400	Create RoleBinding failed	Create a RoleBinding failed.
UCS.01120014	400	Delete RoleBinding failed	Delete a RoleBinding failed.
UCS.01120015	400	Update RoleBinding failed	Update a RoleBinding failed.
UCS.01120016	400	Get RoleBinding failed	Obtain the RoleBinding information failed.
UCS.01130001	400	policy management create reconcile job failed	Create a coordination job in policy management failed.
UCS.01130002	400	policy management create disable job failed	Create a disabling job in policy management failed.
UCS.01130003	400	cluster policy validate failed.	Cluster policy verification failed.
UCS.01130004	400	clusterGroup policy validate failed.	Cluster group policy verification failed.
UCS.01130005	400	cluster policy management is in installing or closing status	Cluster policy management is being installed or has been disabled.
UCS.01130006	400	cluster group policy management is in installing or closing status	Cluster group policy management is being installed or has been disabled.