## Ubiquitous Cloud Native Service

# Best Practices

| | |
|---|---|
| **Issue** | 02 |
| **Date** | 2024-11-01 |

**HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.**

## Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

## Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

# Contents



# 1 Permission Configuration

## 1.1 Granting UCS Permissions to IAM Users

### Application Scenarios

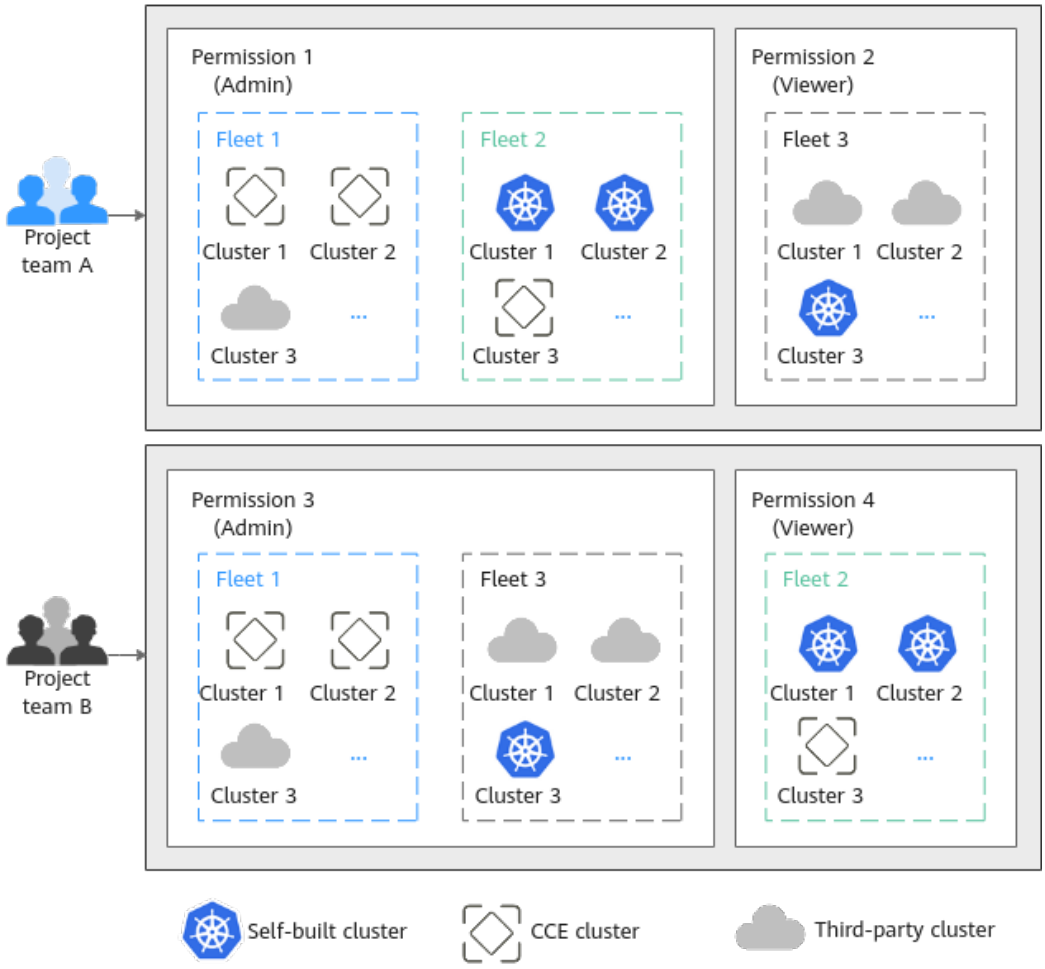
UCS allows you to grant cluster permissions to IAM users and user groups under your account, so that departments or projects can be isolated by permission policy or cluster group.

Assume that you have two project teams, each involving multiple members. **Figure 1-1** shows how their permissions are granted.

- During the development, project team A needs the Admin permission on fleets 1 and 2 and the Viewer permission on fleet 3.
- During the development, project team B needs the Admin permission on fleets 1 and 3 and the Viewer permission on fleet 2.

**Figure 1-1** Permission design




## Solutions

To implement the preceding permission isolation, IAM system policies and UCS permission management must be used together. IAM system policies control the user operations allowed on the UCS console, and UCS permission management controls the fleet and cluster resources that allow for user operations.

As shown in **Figure 1-2**, authorization consists of the following steps:

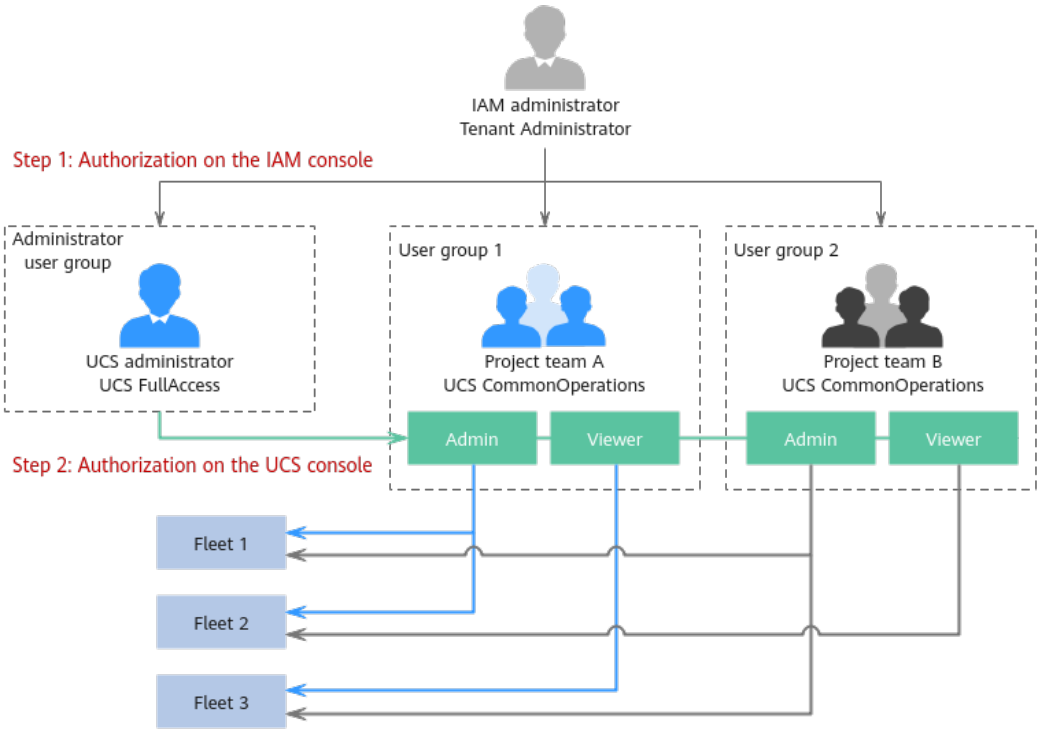
- Step 1: Authorization on the IAM console. The IAM administrator with the Tenant Administrator permission needs to create three user groups. One is the administrator user group, and the other two are user groups (group 1 and group 2) of two project teams (team A and team B). The **UCS FullAccess** and **UCS CommonOperations** permissions are granted to the user groups, respectively.
- Step 2: Authorization on the UCS console. The UCS administrator with the **UCS FullAccess** permission creates the Admin and Viewer permission policies for user groups 1 and 2, and associates the permission policies with the fleet as follows:

  The Admin permission policy of user group 1 is associated with fleets 1 and 2, and the Viewer permission policy is associated with fleet 3. The Admin

permission policy of user group 2 is associated with fleets 1 and 3, and the Viewer permission policy is associated with fleet 2.

**Figure 1-2** Authorization scheme




## Prerequisites

- The account has subscribed to UCS and fleet and cluster resources have been available according to **Figure 1-1**.
- The permission data is prepared according to **Figure 1-2**.

**Table 1-1** Data preparation on the IAM console

| User Group | User | Permission |
| --- | --- | --- |
| Administrator user group: UCS_Group_admin | UCS_Group_admin_User1 | UCS FullAccess |
| User group 1: UCS_Group_1 | UCS_Group_1_User1, UCS_Group_1_User2, ... | UCS CommonOperations |
| User group 2: UCS_Group_2 | UCS_Group_2_User1, UCS_Group_2_User2, ... | UCS CommonOperations |

**Table 1-2** Data preparation on the UCS console

| User Group | User | Permission Type | Permission |
| --- | --- | --- | --- |
| User group 1 | UCS_Group_1_User1, UCS_Group_1_User2, ... | Admin | ucs-group-1-admin |
| | | Viewer | ucs-group-1-readonly |
| User group 2 | UCS_Group_2_User1, UCS_Group_2_User2, ... | Admin | ucs-group-2-admin |
| | | Viewer | ucs-group-2-readonly |

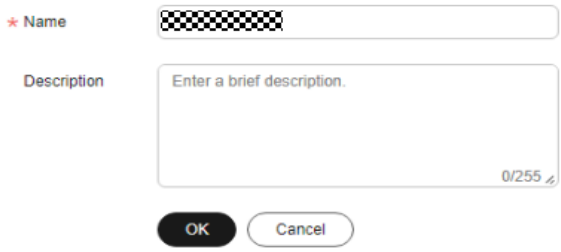
## Step 1: Authorizing the IAM Administrator

**Step 1** Log in to the IAM console as the IAM administrator.

**Step 2** In the navigation pane, choose **User Groups**. In the upper right corner, click **Create User Group**.

**Step 3** In the displayed dialog box, enter the administrator user group name and description, and click **OK**.

**Figure 1-3** Creating a user group

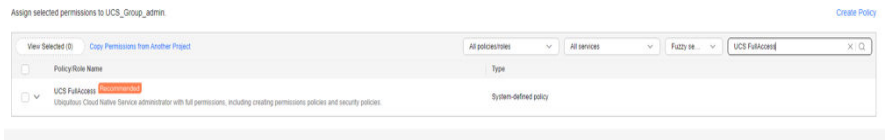



**Step 4** In the user group list, click **Authorize** in the row that contains the target user group.

**Figure 1-4** Granting permissions to the user group


**Step 5** Search for and select the permission policy **UCS FullAccess**.

**Figure 1-5** Selecting the permission policy



**Step 6** Click **Next** and select a scope.

The default option **All resources** is selected, indicating that the IAM user will be able to use all resources, including those in enterprise projects, region-specific projects, and global services under your account based on granted permissions.

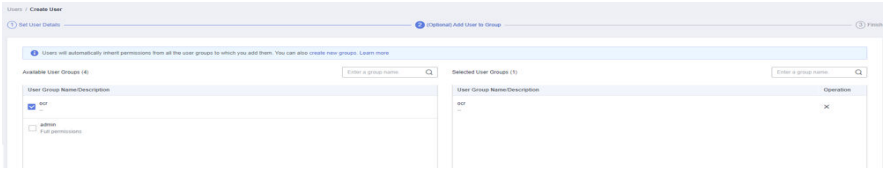
**Step 7** Click **OK**.

**Step 8** In the navigation pane, choose **Users**. In the upper right corner, click **Create User**.

Enter the username and initial password. For details about other parameters, see **Creating an IAM User**.

**Step 9** Click **Next** and select the user group authorized in **Step 4**.

**Figure 1-6** Adding the user to the user group



**Step 10** Click **Create**.

**Step 11** Repeat the preceding steps to create and authorize other user groups and users in **Table 1-1**.

**----End**

## Step 2: Authorizing the UCS Administrator

**Step 1** Log in to the UCS console as the UCS administrator. In the navigation pane, choose **Permissions**.

**Step 2** In the upper right corner, click **Create Permission Policy**.

**Step 3** Configure the parameters as follows:

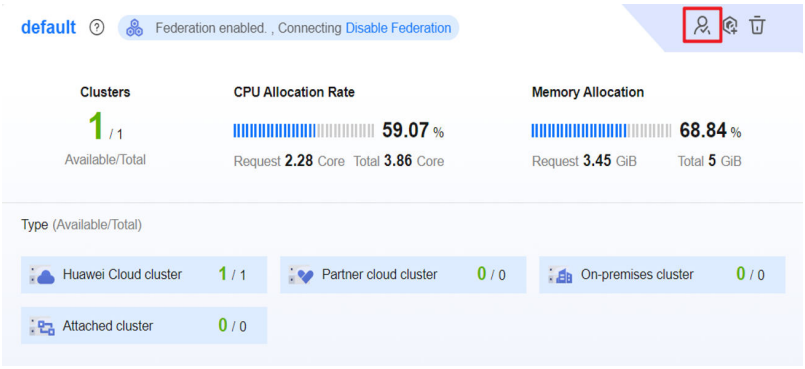
- **Policy Name**: Enter a name, starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
- **User**: Select the user associated with the permission policy, that is, the IAM user created in **Step 8**. In practice, a user group may contain multiple users. When creating a permission policy, you can select all users in the user group to grant permissions in batches.
- **Type**: Select **Admin**. It indicates the read-write permissions on all cluster resource objects.

**Step 4** Click **OK**.

**Step 5** After the permission policy is created, go to the **Fleets** page and click in the upper right corner of the target fleet.

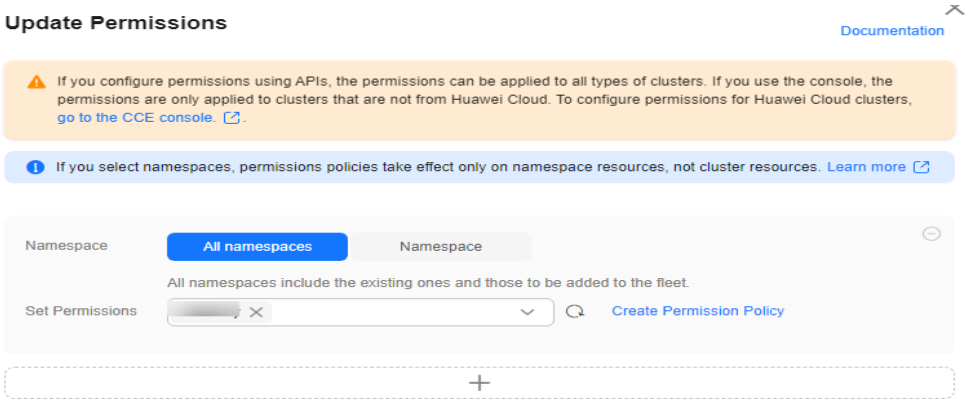
  

**Figure 1-7** Associating the permission policy with the fleet




**Step 6** In the window that slides from the right, click **Set Permissions**. In **Update Permissions**, associate the permission policy created in **Step 3** with all namespaces of the fleet.

**Figure 1-8** Updating permissions




**Step 7** Click **OK**. The IAM user can now log in to the UCS console to use the functions allowed by the permission policy.

**Step 8** Repeat the preceding steps to create other permission policies in **Table 1-2** and associate them with the fleet.

**----End**

# 2 On-Premises Clusters

## 2.1 Creating VPC Endpoints for Connecting to On-Premises Clusters over Private Networks

### Application Scenarios

If you have Kubernetes clusters in your on-premises data center, you can connect your on-premises data center to UCS and enable Container Intelligent Analysis (CIA) to communicate with SWR and OBS. If the public network is unavailable, you can connect your on-premises data center to Huawei Cloud VPC through VPN and then use VPC endpoints to enable VPC to access UCS, SWR, DNS, OBS, and CIA over private networks.

### Preparations

| Service | Domain Name | IP Address (If Any) | Port |
| --- | --- | --- | --- |
| SWR | swr.cn-north-4.myhuaweicloud.com | Obtain the value from VPCEP. | 443 |
| OBS | op-svc-swr-b051-10-38-19-62-3az.obs.cn-north-4.myhuaweicloud.com | N/A | 443 and 80 |

| Service | Domain Name | IP Address (If Any) | Port |
| --- | --- | --- | --- |
| CIA | cie-{First eight digits in the ID of the CIA instance}{First eight digits in the ID of the selected VPC subnet}.cn-north-4.myhuaweicloud.com | Obtain the value from VPCEP. | 443 |
| DNS | N/A | Create a VPC endpoint and select the corresponding IP address. | 53 |

The following table lists the domain names of SWR and OBS in other regions.

| Region | SWR Domain Name | OBS Domain Name |
| --- | --- | --- |
| CN North-Beijing4 | swr.cn-north-4.myhuaweicloud.com | op-svc-swr-b051-10-38-19-62-3az.obs.cn-north-4.myhuaweicloud.com |
| CN East-Shanghai2 | swr.cn-east-2.myhuaweicloud.com | obs.cn-east-2.myhuaweicloud.com |
| CN East-Shanghai1 | swr.cn-east-3.myhuaweicloud.com | op-svc-swr-b051-10-147-7-14-3az.obs.cn-east-3.myhuaweicloud.com |
| CN South-Guangzhou | swr.cn-south-1.myhuaweicloud.com | op-svc-swr-b051-10-230-33-197-3az.obs.cn-south-1.myhuaweicloud.com |
| CN Southwest-Guiyang1 | swr.cn-southwest-2.myhuaweicloud.com | op-svc-swr-b051-10-205-14-19-3az.obs.cn-southwest-2.myhuaweicloud.com |

| Region | SWR Domain Name | OBS Domain Name |
| --- | --- | --- |
| CN North-Ulanqab1 | swr.cn-north-9.myhuaweicloud.com | obs.cn-north-9.myhuaweicloud.com |
| AP-Singapore | swr.ap-southeast-3.myhuaweicloud.com | op-svc-swr-b051-10-38-34-172-3az.obs.ap-southeast-3.myhuaweicloud.com |
| CN-Hong Kong | swr.ap-southeast-1.myhuaweicloud.com | obs.ap-southeast-1.myhuaweicloud.com |
| LA-Mexico City1 | swr.na-mexico-1.myhuaweicloud.com | obs.na-mexico-1.myhuaweicloud.com |
| LA-Mexico City2 | swr.la-north-2.myhuaweicloud.com | obs.la-north-2.myhuaweicloud.com |

## Procedure

**Step 1** Configure a VPN by referring to **Connecting an On-Premises Data Center to a VPC Through a VPN**..

If a VPN has been configured, go to **Step 7**.

NOTE

- The private CIDR block of your on-premises data center cannot overlap with the VPC CIDR block used for connecting to the VPN on Huawei Cloud.
- 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. 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.

**Step 2** **Create a VPN gateway** on Huawei Cloud.

Log in to the Huawei Cloud console and choose **Virtual Private Network**. In the navigation pane, choose **Enterprise – VPN Gateways**. On the displayed page, click the **S2C VPN Gateways** tab. In the upper right corner, click **Buy S2C VPN Gateway**.

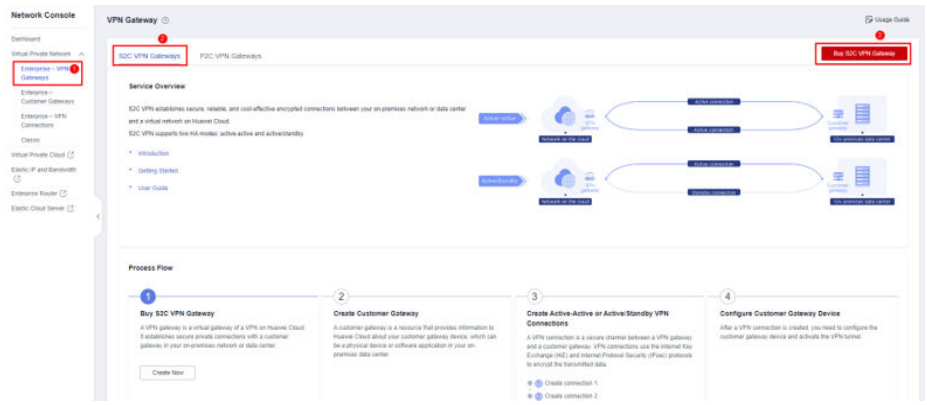


**Table 2-1** Planned data

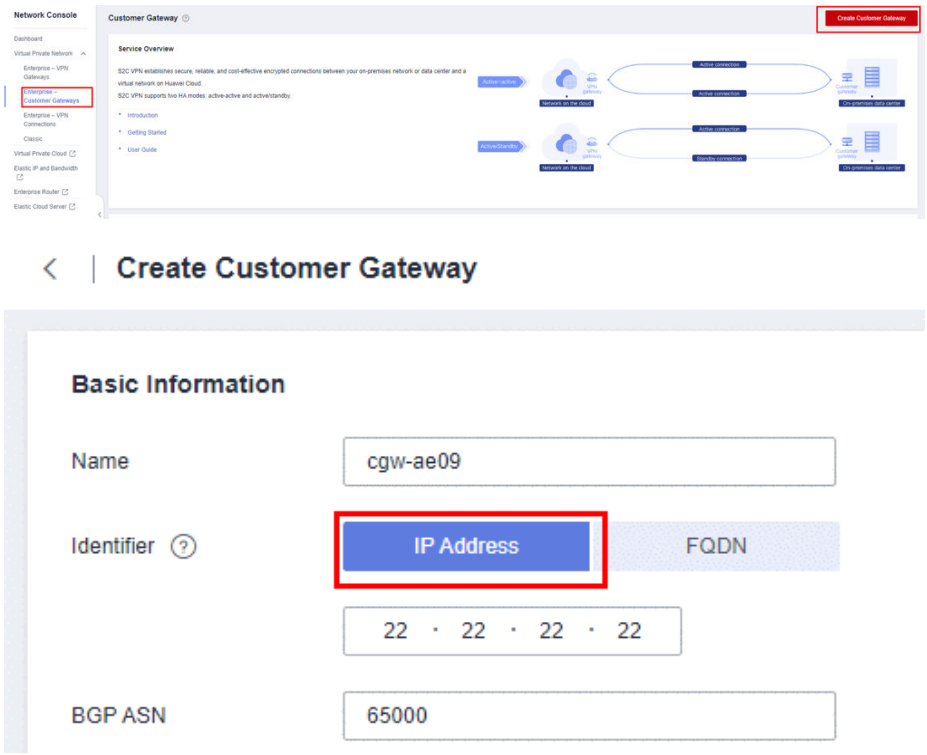
| Category | Planned Item | Planned Value |
| --- | --- | --- |
| VPC | Subnets that need to access the VPC | 10.188.1.0/24 and 100.64.0.0/10 (the CIDR blocks of SWR and OBS) |
| VPN gateway | Interconnection subnet | This subnet is used for communication between the VPN gateway and VPC. The subnet cannot overlap with the existing VPC subnets. |
| | | 10.188.2.0/24 |
| | EIPs | EIPs are automatically generated when you buy EIPs. By default, a VPN gateway uses two EIPs. In this example, the following EIPs are generated: |
| | | Active EIP: 11.xx.xx.11 |
| | | Standby EIP: 11.xx.xx.12 |
| VPN connections | Tunnel interface address | This address is used by a VPN gateway to establish an IPsec tunnel with a customer gateway. At the two ends of the IPsec tunnel, the configured local and remote tunnel interface addresses must be reversed. |
| | | VPN connection 1: 169.254.70.1/30 |
| | | VPN connection 2: 169.254.71.1/30 |

**Step 3** **Create a customer gateway**.

In the navigation pane, choose **Enterprise – Customer Gateways**. On the displayed page, click **Create Customer Gateway**.

Set **Identifier** to **IP Address** and enter the public IP address of the on-premises data center.




**Step 4** **Create a VPN connection**.

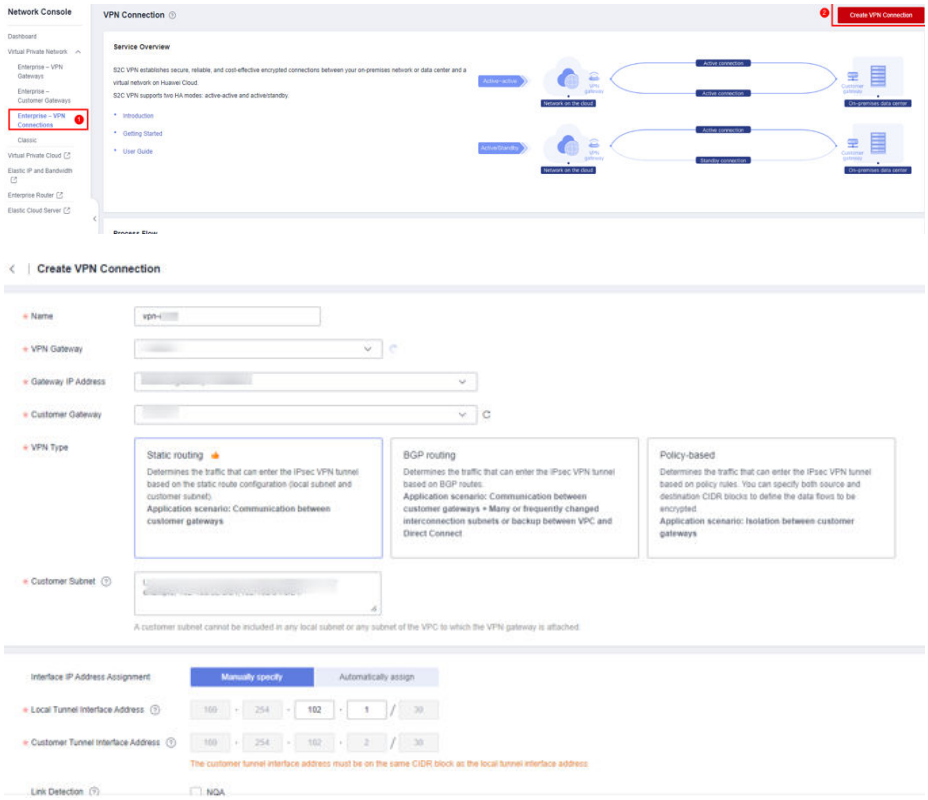


**Table 2-2** Parameters for creating a VPN connection

| Parameter | Description | Example Value |
| --- | --- | --- |
| Name | Enter a VPN connection name. | vpn-xxx |
| VPN Gateway | Select the VPN gateway created in **Step 2**. | vpngw-xxx |
| Gateway IP Address | Select the active EIP of the VPN gateway. | 11.xx.xx.11 |
| Customer Gateway | Select the customer gateway created in **Step 3**. | cgw-xxx |
| VPN Type | Select **Static routing**. | Static routing |
| Customer Subnet | Enter the subnet of the on-premises data center that needs to access the VPC.<br>**NOTE**<br>• The customer subnet can overlap with the local subnet but cannot be the same as the local subnet.<br>• A customer subnet cannot be included in the existing subnets of the VPC associated with the VPN gateway. It also cannot be the destination address in the route table of the VPC associated with the VPN gateway.<br>• Customer subnets cannot be the reserved CIDR blocks of VPCs, for example, 100.64.0.0/10 or 214.0.0.0/8.<br>• If the interconnection subnet is associated with an ACL rule, ensure that the ACL rule permits the TCP port for traffic between all local and customer subnets. | 172.16.0.0/16 |
| Interface IP Address Assignment | The options are **Manually specify** and **Automatically assign**. | Manually specify |
| Local Tunnel Interface Address | Configure the tunnel IP address of the VPN gateway.<br>**NOTE**<br>The local and remote interface addresses configured on the customer gateway device must be the same as the values of **Customer Tunnel Interface Address** and **Local Tunnel Interface Address**, respectively. | 169.254.70.2/30 |
| Customer Tunnel Interface Address | Specify the tunnel interface address configured on the customer gateway device. | 169.254.70.1/30 |

| Parameter | Description | Example Value |
| --- | --- | --- |
| Link Detection | This function is used for route reliability detection in multi-link scenarios.<br>**NOTE**<br>When enabling this function, ensure that the customer gateway supports ICMP and is correctly configured with the customer interface IP address of the VPN connection. Otherwise, VPN traffic will fail to be forwarded. | Select **NQA**. |
| PSK/Confirm PSK | Specify the negotiation key of the VPN connection.<br>The PSKs configured on the VPN console and the customer gateway device must be the same. | Test@123 |
| Policy Settings | There are **IKE Policy** and **IPsec Policy**, which specifies the encryption and authentication algorithms of a VPN tunnel.<br>The policy settings on the VPN console and the customer gateway device must be the same. | Default |

**Step 5** **Configure the customer gateway device**.

**Step 6** Verify the network connectivity.

1. Log in to the management console.
2. Click  in the upper left corner and select a region and a project.
3. Click **Service List** and choose **Compute** > **Elastic Cloud Server**.
4. Log in to the ECS.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.

   In this example, use VNC provided on the management console to log in to an ECS.
5. Run the following command on the ECS console:

   ping 172.16.0.100

   **172.16.0.100** is the IP address of a server in the on-premises data center. Replace it with an actual server IP address.

   If information similar to the following is displayed, the client can communicate with the ECS:

```
Reply from xx.xx.xx.xx: bytes=32 time=28ms TTL=245
Reply from xx.xx.xx.xx: bytes=32 time=28ms TTL=245
Reply from xx.xx.xx.xx: bytes=32 time=28ms TTL=245
Reply from xx.xx.xx.xx: bytes=32 time=27ms TTL=245
```

**Step 7** Create VPC endpoints on Huawei Cloud.

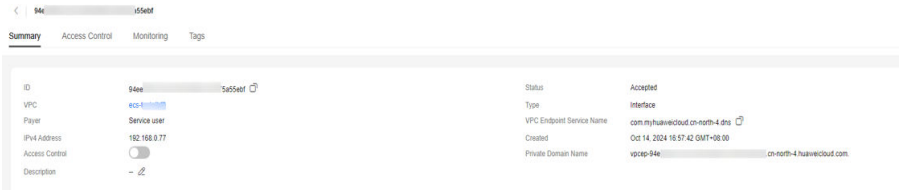
  

To enable an on-premises data center to access DNS, SWR, OBS, and UCS on Huawei Cloud, you need to create their endpoints in the VPC that communicates with the on-premises data center.

### Creating a VPC Endpoint for DNS

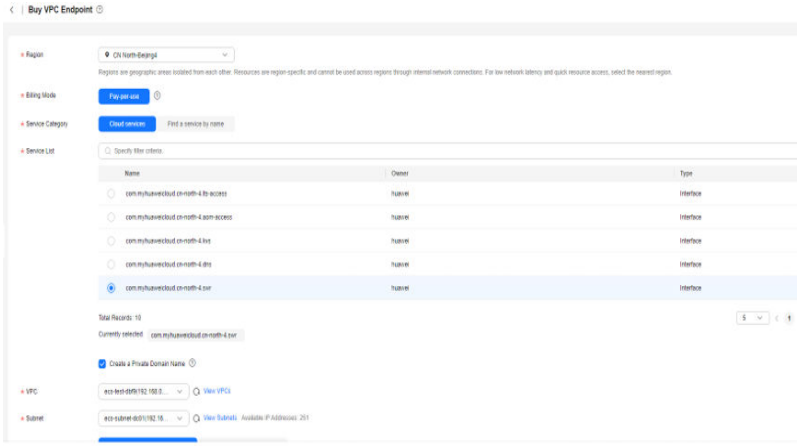
Click **Service List** and choose **Networking** > **VPC Endpoint**.

1. In the navigation pane, choose **VPC Endpoint** > **VPC Endpoints**.
2. On the displayed page, click **Buy VPC Endpoint**.
3. Set **Service Category** to **Cloud services** and select **com.myhuaweicloud.cn-north-4.dns** from **Service List**.
4. Select the VPC that has been connected in **Step 2**.
5. Click the generated VPC endpoint name to view the IP address.

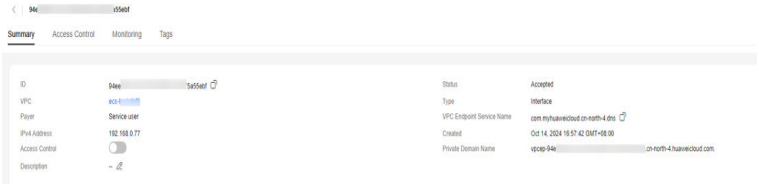


### Creating a VPC Endpoint for SWR

1. Click **Service List** and choose **Networking** > **VPC Endpoint**.
2. In the navigation pane, choose **VPC Endpoint** > **VPC Endpoints**.
3. On the displayed page, click **Buy VPC Endpoint**.
4. Set **Service Category** to **Cloud services** and select **com.myhuaweicloud.cn-north-4.swr** from **Service List**.
5. Select the VPC that has been connected in **Step 2**.



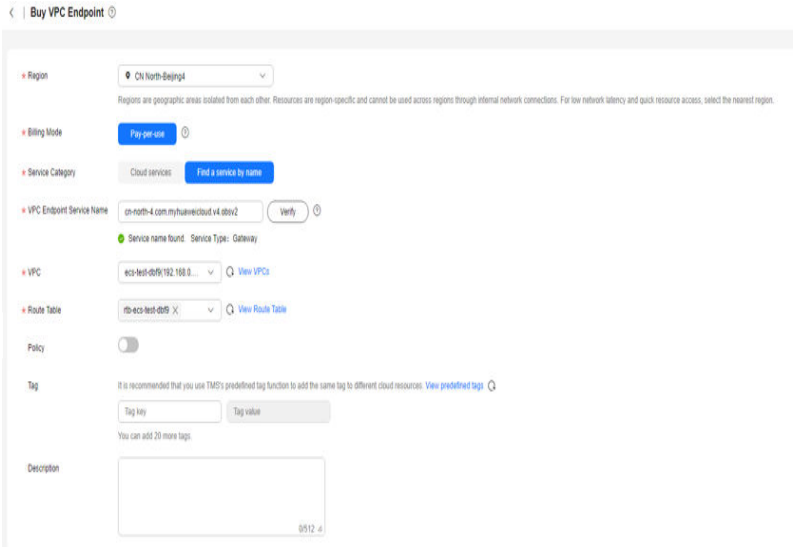
6. Click the generated VPC endpoint name to view the IP address.



### Creating a VPC Endpoint for OBS

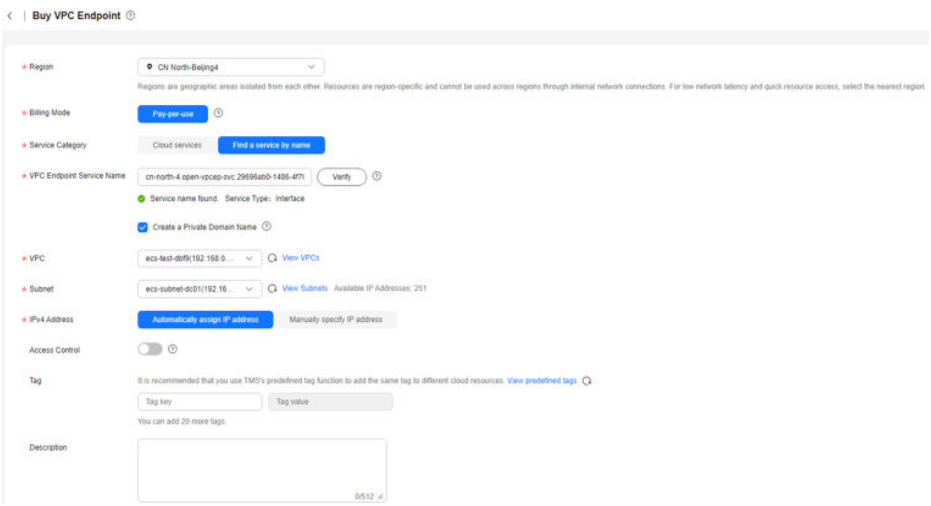
  

1. Click **Service List** and choose **Networking** > **VPC Endpoint**.
2. In the navigation pane, choose **VPC Endpoint** > **VPC Endpoints**.
3. On the displayed page, click **Buy VPC Endpoint**.
4. Set **Service Category** to **Find a service by name** and **VPC Endpoint Service Name** to **cn-north-4.com.myhuaweicloud.v4.obsv2**. Then, click **Verify**.
5. Select the VPC that has been connected in **Step 2**.




### Creating a VPC Endpoint for UCS

1. Click **Service List** and choose **Networking** > **VPC Endpoint**.
2. In the navigation pane, choose **VPC Endpoint** > **VPC Endpoints**.
3. On the displayed page, click **Buy VPC Endpoint**.
4. Set **Service Category** to **Find a service by name** and **VPC Endpoint Service Name** to **cn-north-4.open-vpcep-svc.29696ab0-1486-4f70-ab35-a3f6b1b37c02**. Then, click **Verify**.
5. Select the VPC that has been connected in **Step 2**.




**Step 8** Add the Huawei Cloud DNS forwarder to the DNS server in the on-premises data center.

1. Add DNS records on the DNS server in your on-premises data center to forward requests for resolving the private domain name of Huawei Cloud to the DNS VPC endpoint.

   Take DNS Bind as an example. In **/etc/named.conf**, add the DNS forwarder configuration and set **forwarders** to the IP address of the VPC endpoint for accessing DNS. *{xx.xx.xx.xx}* represents the IP address of the VPC endpoint for accessing DNS in **Step 7**.

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

2. Configure static DNS resolution and add the IP addresses of SWR and CIE instances. The IP addresses can be obtained from the CIA instance.

   Take CN North-Beijing4 as an example. If **dnsmasq** is used, add the following static resolution to **/etc/dnsmasq.conf**:

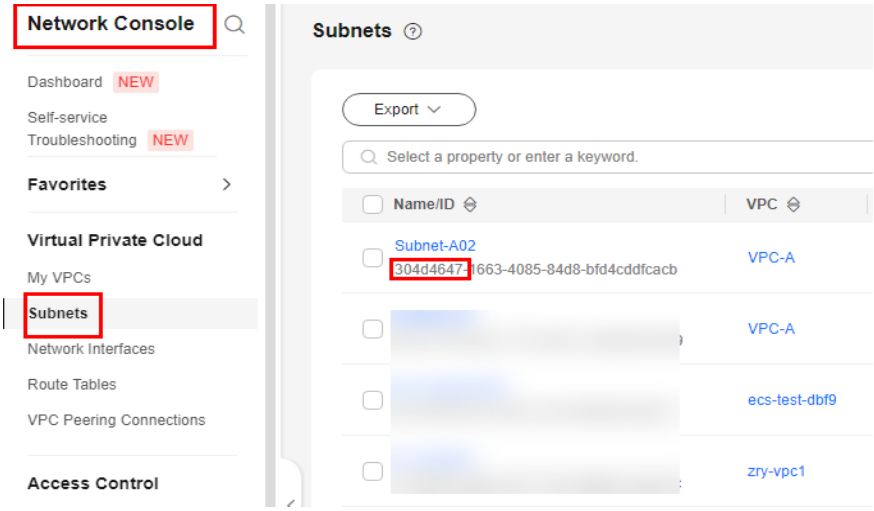
   ```
   address=/swr.cn-north-4.myhuaweicloud.com/xx.xx.xx.xx
   ```

   *xx.xx.xx.xx* represents the IP address of the VPC endpoint for accessing SWR in **Step 7**.

   address=/cie-*{First eight digits in the ID of the CIA instance}{First eight digits in the ID of the VPC subnet}*.cn-north-4.myhuaweicloud.com

   Obtains the first eight digits in the ID of the CIA instance.

   Obtains the first eight digits in the ID of the VPC subnet.




**Step 9** Register an on-premises cluster with UCS as follows: Prepare the kubeconfig file of the cluster to be accessed. Ensure that the value of the **server** field in this file is a private IP address (not a public IP address or domain name). Log in to the UCS console. In the navigation pane, choose **Fleets**. In the **On-premises cluster** card, click **Register Cluster**. Select a cluster service provider and configure cluster parameters as prompted. For details, see **Preparing for Installation**.

After a cluster is connected, you need to configure an endpoint for the cluster to access the network so that the cluster can be taken over by UCS. Click **Private access** and select the VPC that connects to the on-premises data center through the VPN.

**NOTE**

The VPC can be selected only when the configuration in **Step 7** is complete.

Download the configuration file of the cluster agent and upload it to the Kubernetes cluster in the on-premises data center. Run the following command to deploy the agent in the cluster to be connected:

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

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
```

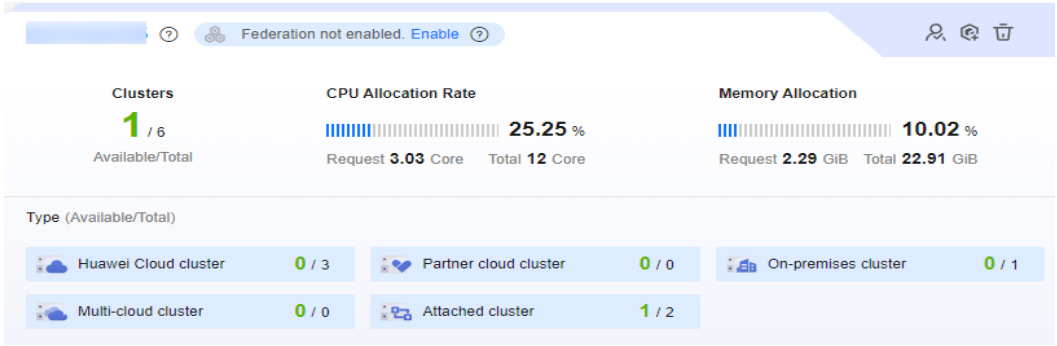
Check the status of the cluster agent.

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

Expected log output for normal running:

```
Start serving
```

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




**Step 10** Connect the Kubernetes cluster to CIA.

1. Log in to the UCS console and choose **Container Intelligent Analysis** in the navigation pane. Select a CIA instance and click **Enable Monitoring** in the upper right corner. Select a cluster to be connected in the on-premises data center and click **Next: Configure Connection**.
2. Set **Data Access** to **Private access**. **Private access**: Select the VPC that has been connected to the on-premises data center through a VPN.
3. Complete the add-on configuration.

   The system provides default add-on settings, including the add-on specifications, collection period, and storage. If you want to change the default values, click ▼ next to the add-on parameters to expand the configuration items.

   **Add-on Specifications**: There are **Demo (≤ 100 containers)** and other options. Different specifications have different requirements on cluster resources such as CPU and memory. UCS preliminarily checks whether an add-on can be installed on the cluster node. If no, a message will be displayed.

- **Storage**: used to temporarily store Prometheus data.
- **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.

Wait till the cluster is connected. After 2 to 3 minutes, the cluster is in the low-risk, medium-risk, or high-risk state, and monitoring data is displayed.

**----End**

# 2.2 Using Workload Identities to Securely Access Cloud Services

## Application Scenarios

With workload identities, your workloads in a cluster can access cloud services like IAM users without using the AK/SK, reducing security risks.

This section describes how to use workload identities in UCS.

## Solution Process

**Figure 2-1** shows the process of using workload identities.

**Step 1** Assign authorization in advance.

1. **Obtain the JSON Web Key Set (JWKS) issued by the private key of an on-premises cluster** from UCS. The JWKS is used to verify the token issued by this cluster for a ServiceAccount.
2. **Create an identity provider (IdP)** for the on-premises cluster in IAM.
3. Add the public key of this cluster for the IdP. When a workload uses a token to send requests, IAM will use this public key to verify the token.
4. Add a rule to map the ServiceAccount to the IAM account. After the configuration, the ServiceAccount has the permissions of the IAM account.

**Step 2** Configure the token.

1. Deploy a workload and configure a ServiceAccount.
2. Mount the token of the ServiceAccount to the workload.

**Step 3** Verify the token.

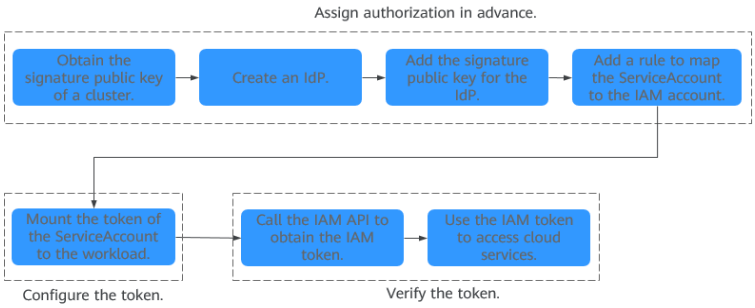
1. Call the IAM API to obtain the IAM token.

2. Use the IAM token to access cloud services.

**----End**

**Figure 2-1** Process of using workload identities




## Obtaining the JWKS of an On-Premises Cluster

**Step 1** Use kubectl to access the on-premises cluster.

**Step 2** Run the following command to obtain the public key:

**kubectl get --raw /openid/v1/jwks**

A json string is returned, containing the public key of the on-premises cluster for accessing the IdP.

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "Ew29q....",
      "alg": "RS256",
      "n": "peJdm...."
    }
  ]
}
```

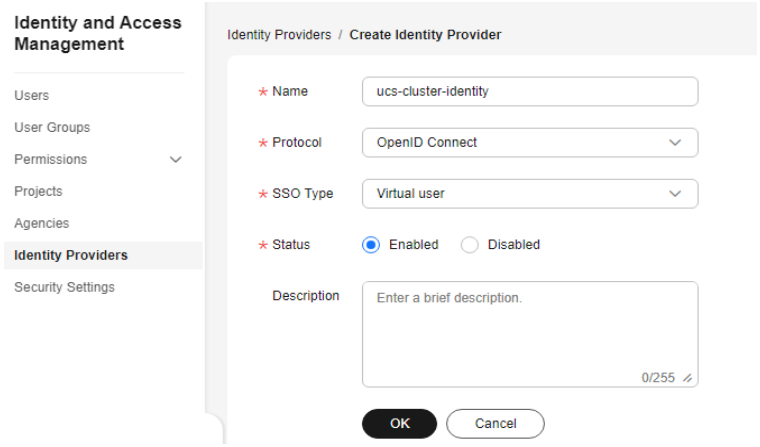
**----End**

## Creating an IdP

**Step 1** Log in to the IAM console, create an IdP, and select **OpenID Connect** for **Protocol**.

**Figure 2-2** Creating an IdP




**Step 2** Click **OK**. Then, modify the IdP information as described in **Table 2-3**. If you need an identity conversion rule, click **Create Rule**.

**Figure 2-3** Modifying IdP information

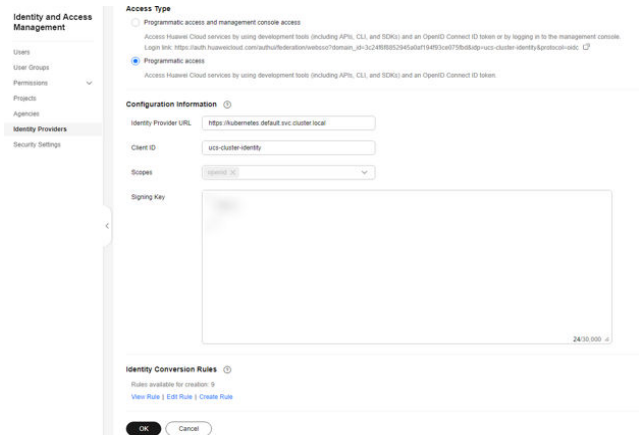


**Figure 2-4** Creating an identity conversion rule

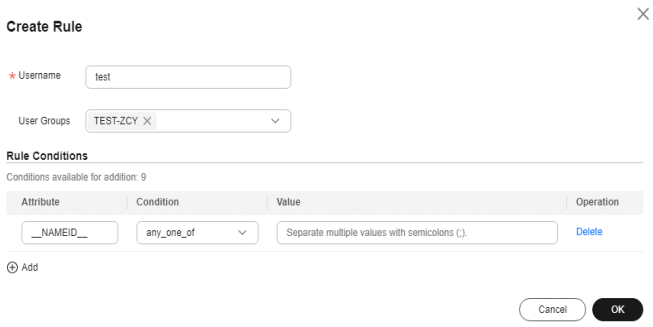



**Table 2-3** IdP parameters

| Parameter | Description |
| --- | --- |
| Access Type | Select **Programmatic access**. |

| Parameter | Description |
| --- | --- |
| Configuration Information | • **Identity Provider URL**: Enter **https://kubernetes.default.svc.cluster.local**.<br>• **Client ID**: Enter **ucs-cluster-identity**.<br>• **Signing Key**: Enter the JWKS of the on-premises cluster obtained in **Obtaining the JWKS of an On-Premises Cluster**. |
| Identity Conversion Rules | An identity conversion rule maps the ServiceAccount of a workload to an IAM user group.<br>For example, create a ServiceAccount named *xxx* in namespace **default** of the cluster and map it to user group **demo**. If you use the IdP ID to access cloud services, you have the permissions of the **demo** user group.<br>Value format: **system:serviceaccount:***Namespace***:***ServiceAccountName*. |

**Step 3** Click **OK**.

**----End**

## Obtaining an IAM Token

**Step 1** Create a ServiceAccount, whose name must be the value of **ServiceAccountName** set in **Step 2**.

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: test_sa_name # The value must be the same as that in the identity conversion rule.
```

**Step 2** Add the ServiceAccount and volume configurations to the workload.

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

```
    containers:
    - name: container-1
      image: nginx:latest
      volumeMounts
      - mountPath: "/var/run/secrets/tokens" # Mount the ServiceAccountToken generated by Kubernetes to /var/run/secrets/tokens/token_path.
        name: token-volume
    imagePullSecrets:
    - name: default-secret
    serviceAccountName: test_sa_name    # Name of the ServiceAccount created in the previous step
    volumes:
    - name: token-volume
      projected:
        defaultMode: 420
        sources:
        - serviceAccountToken:
            audience: ucs-cluster-identity   # The value must be the client ID of the IdP.
            expirationSeconds: 7200     # Expiration time
            path: token_path     # Path name, which can be customized
```

**Step 3** After the creation, log in to the container to obtain the token.


**Step 4** Construct request body data. For details, see **Obtaining a Project ID**.

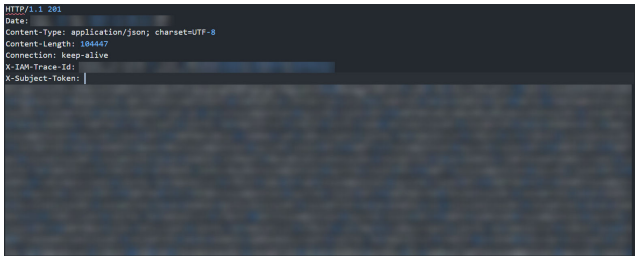
```
{
   "auth" : {
     "id_token" : {
       "id" : "eyJhbGciOiJSUzIXXXXX"   // Token obtained in the previous step
     },
     "scope": {
       "project" : {
         "id" : "05495693df80d3c92fa1c01795c2be02",  // Project ID
         "name" : "cn-north-7"
       }
     }
   }
 }
```

**Step 5** Call the IAM API to obtain the IAM token. For details about the IAM endpoint, see **Regions and Endpoints**.

```
curl -i --location --request POST 'https://{{iam endpoint}}/v3.0/OS-AUTH/id-token/tokens'  --header 'X-Idp-Id: {{workload_identity}}'  --header 'Content-Type: application/json'  --data @token_body.json
```

- Replace *{workload_identity}* with the name of the IdP registered in **Step 1**. In this example, the name is **ucs-cluster-identity**.
- **token_body.json** is the constructed request body data file.

**Step 6** Obtain the IAM token from the response body. The value of **X-Subject-Token** in the response header is the IAM token.




**----End**

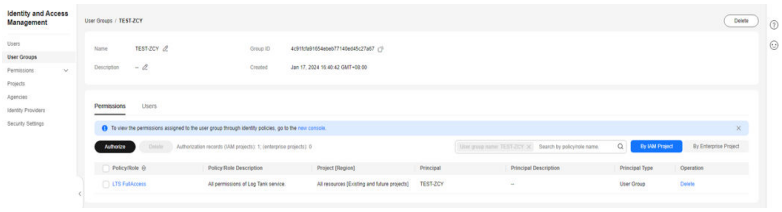
  

## Using an IAM Token to Access Cloud Services

This section uses LTS as an example.

**Step 1** Before using an IAM token to access LTS, you need to configure permissions for the user group.

**Step 2** To call LTS, you need to add the LTS FullAccess permissions to the user group.

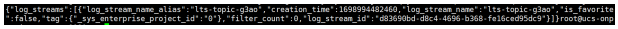


**Step 3** Run the following command to call the service API:

```
curl --location --request GET 'https://ltsperform.cn-north-7.myhuaweicloud.com/v2/{{Project ID}}/groups/{{Log group ID}}/streams' \--header 'Content-Type: application/json;charset=utf-8' \--header 'X-Auth-Token: {{IAM token obtained in the previous step}}' \--data-raw ''
```

The value of *{Log group ID}* can be obtained in LTS.

The following figure shows the expected result.



**----End**

# 3 Cluster Federation

## 3.1 Using Cluster Federation to Implement Multi-Active DR for Applications

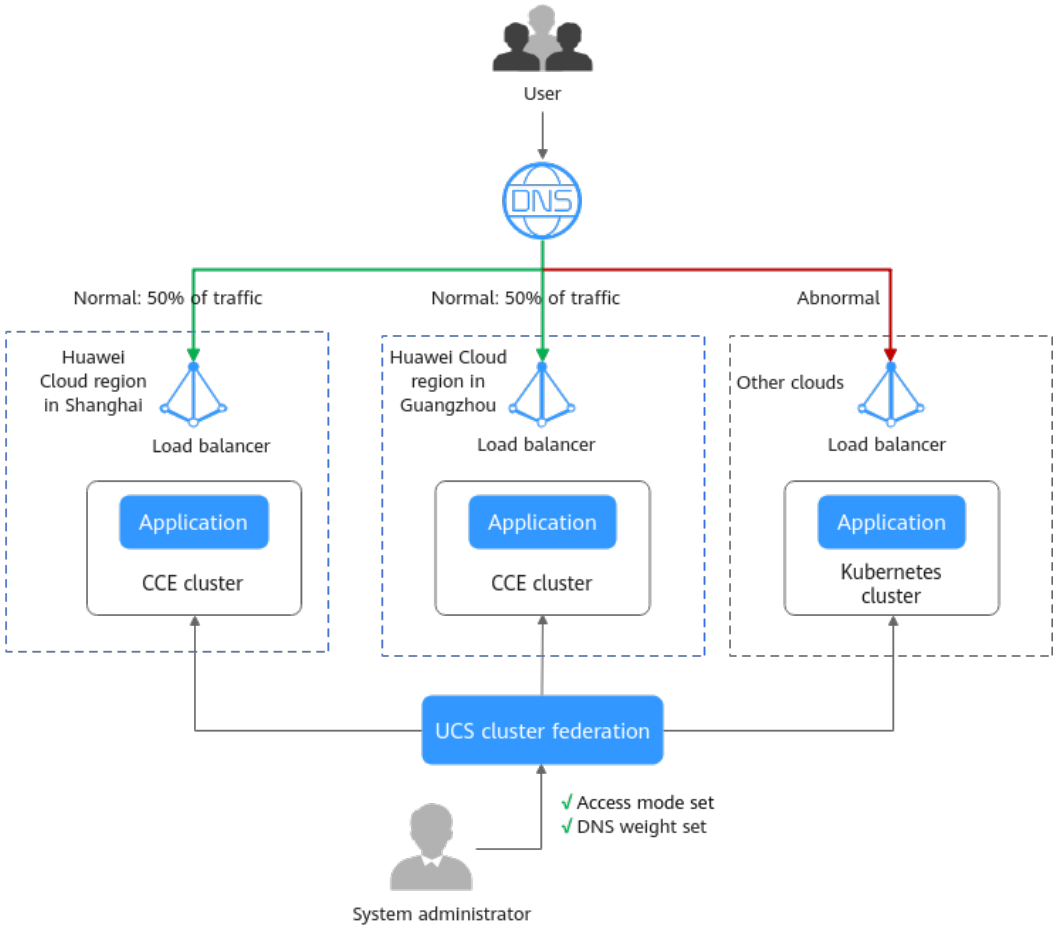
### Application Scenarios

To tackle single points of failure (SPOFs), UCS allows instances of an application to run on multiple clouds. When one of the clouds is down, cluster federation will migrate instances to other clouds and switch over traffic within seconds, significantly improving service reliability.

**Figure 3-1** shows the multi-active DR solution in UCS. Under DNS policies, instances of an application are distributed to three Kubernetes clusters: two Huawei Cloud CCE clusters (deployed in different regions) and one third-party cloud cluster.

**Figure 3-1** Multi-active DR for multi-cloud clusters




## Prerequisites

- You have created a cluster. The following is an example of creating a CCE cluster (guide: **Buying a CCE cluster** in two regions (CN South-Guangzhou and CN East-Shanghai1). The Kubernetes version must be 1.19 or later, and each cluster must have at least one available node.

  NOTE

  In your production environment, you can deploy clusters in different regions, AZs, or even clouds to implement multi-active DR.

- You have created a public zone in Huawei Cloud DNS. For details, see **Routing Internet Traffic to a Website**.

## Setting Up the Basic Environment

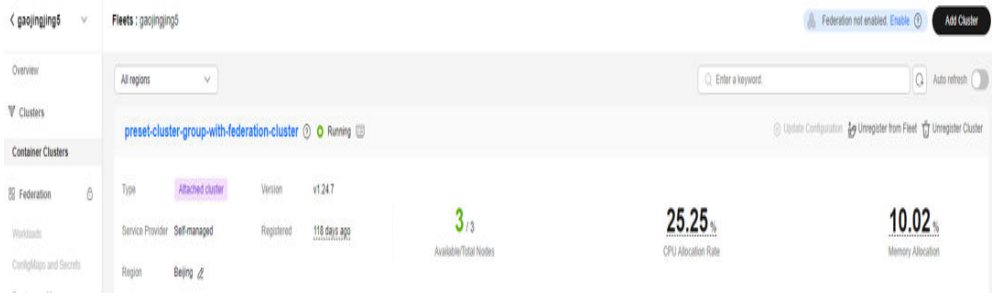
**Step 1** Register clusters to UCS and configure cluster access. For details, see **Registering a Cluster**.

For example, register clusters **ccecluster01** and **ccecluster02** to the fleet **ucs-group** of UCS and check whether the clusters are running normally.

**Step 2** Enable cluster federation for the fleet and ensure that the clusters have been connected to a federation. For details, see **Cluster Federation**.

**Figure 3-2** Clusters




**Step 3** Creating Workloads

To show the traffic switchover effect, the container image versions of the two clusters in this section are different. (This difference does not exist in the actual production environment.)

- Cluster **ccecluster01**: If the example application uses the image **nginx:gz**, the message "ccecluster01 is in Guangzhou." will be returned.
- Cluster **ccecluster02**: If the example application uses the image **nginx:sh**, the message "ccecluster02 is in Shanghai." will be returned.

Before the operation, upload the images of the example applications to the SWR image repository in the region where the clusters are located. That is, upload the image **nginx:gz** to CN South-Guangzhou and the image **nginx:sh** to CN East-Shanghai1. Otherwise, the workloads will malfunction because it cannot pull the images.

**NOTE**

In this example, example clusters and workloads are not limited in terms of cloud service providers, regions, and quantity.

1. Log in to the UCS console. In the navigation pane, choose **Fleets**.
2. Click the name of the fleet for which cluster federation has been enabled. The fleet console is displayed.
3. In the navigation pane, choose **Federation** > **Workloads**. In the upper right corner, click **Create from Image**.
4. Enter the basic information and configure container parameters. The image name can be user-defined. Click **Next: Scheduling and Differentiation**.
5. Configure the cluster scheduling policy, complete differentiated cluster configuration, and click **Create Workload**.
   - **Scheduling**: Select **Cluster weight** and set the weight of each cluster to 1.
   - **Differentiated Settings**: Click on the left of the cluster to enable differentiated settings. Set the image name of **ccecluster01** to **swr.cn-south-1.myhuaweicloud.com/kubernetes-test2/nginx:gz** (address of the image **nginx:gz** in the SWR image repository) and that of **ccecluster02** to **swr.cn-east-3.myhuaweicloud.com/kubernetes-test2/nginx:sh**.

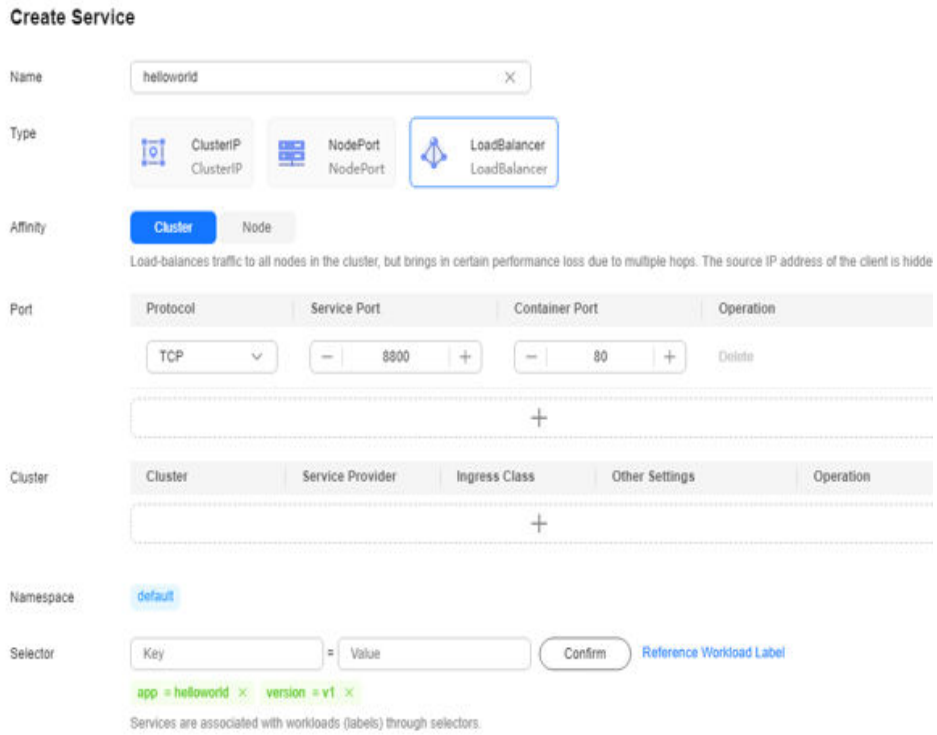
**Figure 3-3** Scheduling and differentiation

**Step 4** Create a LoadBalancer access.

1. Log in to the Huawei Cloud UCS console. In the navigation pane, choose **Fleets**.
2. Click the name of the fleet for which cluster federation has been enabled. The fleet console is displayed.
3. In the navigation pane, choose **Federation** > **Services and Ingresses**. In the upper right corner, click **Create Service**.
4. Configure the parameters and click **OK**.
   - **Service Type**: Select **LoadBalancer**.
   - **Port**: Select **TCP** for **Protocol**, and enter the service port and container port, for example, **8800** and **80**.
   - **Cluster**: Click + to add clusters **ccecluster01** and **ccecluster02** in sequence. Select a shared load balancer for **LoadBalancer**. The load balancer must be in the VPC of each cluster. If no load balancer is available in the list, click **Create Load Balancer** to create one on the ELB console. Retain default values for other parameters.
   - **Selector**: Services are associated with workloads through selectors. In this example, a workload label is referenced to add a label.

**Figure 3-4** Creating a Service



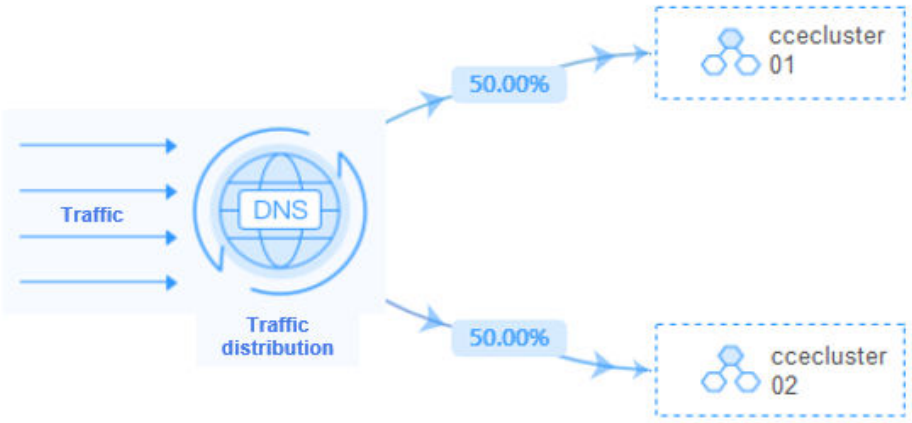

**Step 5** Create a DNS policy.

1. Log in to the Huawei Cloud UCS console. In the navigation pane, choose **Fleets**.
2. Click the name of the fleet for which cluster federation has been enabled. The fleet console is displayed.

3. In the navigation pane, choose **Federation** > **DNS Policies**. Then, add a root domain name.
4. In the upper right corner, click **Create DNS Policy**. Then, configure the parameters.
   - **Target Service**: Select the Service created in **Step 4**.
   - **Distribution Mode**: Select **Adaptive**. Traffic will be automatically distributed based on the number of pods in each cluster. In this example, both **ccecluster01** and **ccecluster02** contain one pod, so each cluster receives 50% of the traffic.

**Figure 3-5** Traffic ratio topology




**----End**

## Verifying Multi-Active DR

You have deployed applications in clusters **ccecluster01** and **ccecluster02** and allowed external access via LoadBalancer Services. After the DNS policy in **Step 5** is created, the system automatically adds a resolution record for the selected root domain name and generates a unified external access path (domain name address) on UCS. This allows you to access the domain name address to verify traffic distribution.

**Step 1** Obtain the domain name address.

1. Log in to the UCS console. In the navigation pane, choose **Fleets**.
2. Click the name of the fleet for which cluster federation has been enabled. The fleet console is displayed.
3. In the navigation pane, choose **Federation** > **DNS Policies**. The value of **Domain Name Address** in the list is the domain name address.

**Step 2** Run the following command on a host that has been connected to the public network to continuously access the domain name address and check the cluster application processing status.

- Generally, applications in both clusters receive traffic and each cluster processes 50% of the traffic.

```
while true;do wget -q -O- helloworld.default.mcp-xxx.svc.xxx.co:8800; done
ccecluster01 is in Guangzhou.
ccecluster02 is in Shanghai.
```

```
ccecluster01 is in Guangzhou.
ccecluster02 is in Shanghai.
ccecluster01 is in Guangzhou.
ccecluster02 is in Shanghai.
...
```

- When an application exception occurs on **ccecluster01** (simulating an application exception by shutting down a cluster node), the system routes all traffic to **ccecluster02**, so that users are unaware of the exception.

  ```
  while true;do wget -q -O- helloworld.default.mcp-xxx.svc.xxx.co:8800; done
  ccecluster02 is in Shanghai.
  ccecluster02 is in Shanghai.
  ccecluster02 is in Shanghai.
  ccecluster02 is in Shanghai.
  ccecluster02 is in Shanghai.
  ccecluster02 is in Shanghai.
  ...
  ```

  Return to the UCS console. You can see that the cluster traffic ratio in the domain name list has changed. **ccecluster02** takes over 100% traffic, which is consistent with the configured traffic ratio and what we have observed.

**----End**

# 3.2 Using a VPC Peering Connection to Connect CCE Clusters

## Application Scenarios

Before creating an MCS object, ensure connectivity of both inter-cluster nodes and containers. You can create a VPC peering connection to connect CCE clusters across VPCs.

This section describes how you can create a VPC peering connection for connectivity of both inter-cluster nodes and containers.

## Configuring Cluster Network Types

Set the network type to underlay for inter-cluster pod communication. The following table lists the types of CCE clusters that support underlay networks.

**Table 3-1** Types of CCE clusters that support underlay networks

| CCE Cluster Type | Network Type | Support Underlay Network |
| --- | --- | --- |
| CCE clusters | Container tunnel network | No |
| | VPC network | Yes |
| CCE Turbo clusters | Cloud native network 2.0 | Yes |

## Creating a VPC Peering Connection

**Step 1** Go to the VPC peering connection list page.

**Step 2** In the upper right corner of the page, click **Create VPC Peering Connection**. In the displayed dialog box, configure parameters as prompted. For details about the parameters, see **Table 3-2**.

**Figure 3-6** Creating a VPC peering connection

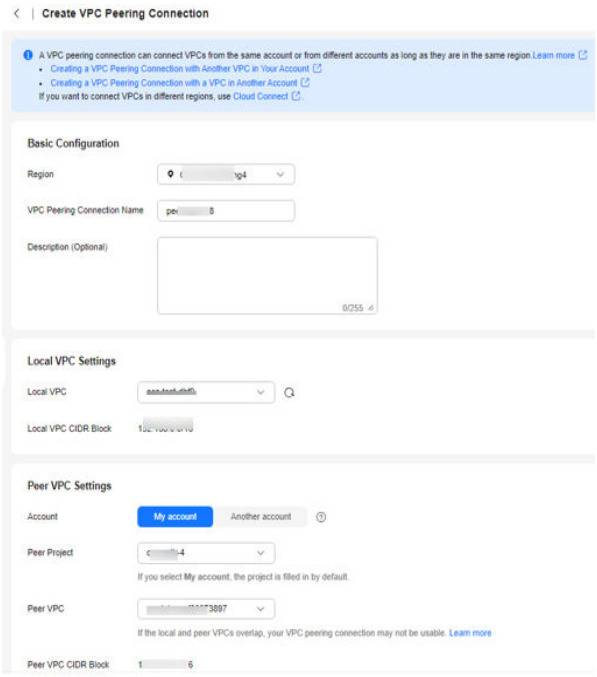



**Table 3-2** Parameters for creating a VPC peering connection

| Parameter | Mandatory | Description |
| --- | --- | --- |
| VPC Peering Connection Name | Yes | Name of the VPC peering connection.<br>The name can contain a maximum of 64 characters, including letters, digits, hyphens (-), and underscores (_). |
| Local VPC | Yes | VPC of the local cluster. Select one from the drop-down list. |
| Local VPC CIDR Block | Yes | CIDR block of the local VPC. |
| Account | Yes | Select **My account** or **Another account**. In this example, **My account** is selected.<br>• **My account**: The local and peer VPCs are from the same account.<br>• **Another account**: The local and peer VPCs are from different accounts. |

| Parameter | Mandatory | Description |
| --- | --- | --- |
| Peer Project | Yes | The system fills in the corresponding project by default when **Account** is set to **My account**.<br>For example, if two VPCs (VPC-A and VPC-B) are in account A in region A, the system fills in the corresponding project of account A in region A by default. |
| Peer VPC | Yes | VPC of the peer cluster. Select one from the drop-down list. |
| Peer VPC CIDR Block | Yes | CIDR block of the peer VPC.<br>The local and peer VPCs cannot have identical or overlapping CIDR blocks. Otherwise, the routes added for the VPC peering connection may not take effect. |
| Description | No | Description of the connection. Enter up to 255 characters. Angle brackets (< or >) are not allowed. |

**Step 3** Click the VPC peering connection name. On the displayed page, click **Add Route**.

As shown in **Figure 3-7**, you need to configure VPC CIDR blocks for local and peer clusters. For details, see **Table 3-3**.

**Figure 3-7** Adding a route

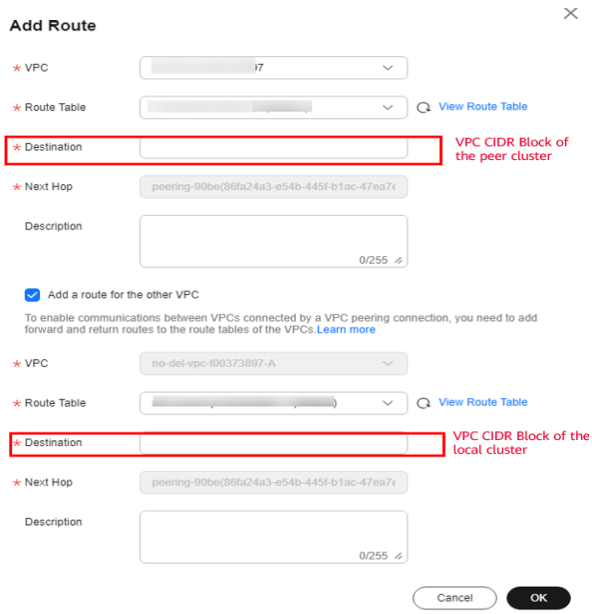
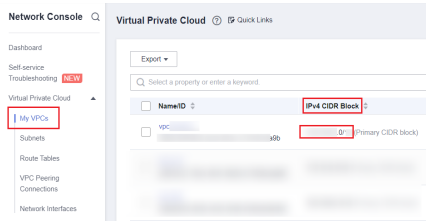


**Table 3-3** Route parameters

| Parameter | Mandatory | Description |
| --- | --- | --- |
| Destination | Yes | Enter the VPC CIDR block for the peer cluster.<br>To query this CIDR block:<br>1. Log in to the VPC console.<br>2. In the navigation pane, choose **Virtual Private Cloud** > **My VPCs**. On the displayed page, locate the peer VPC and copy its IPv4 CIDR block.<br><br>**Figure 3-8** Querying the VPC CIDR block of the peer cluster<br>  |
| Destination | Yes | Enter the VPC CIDR block for the local cluster.<br>**CAUTION**<br>The destination of each route must be unique. |
| Description | No | Supplementary information about the route.<br>Enter up to 255 characters. Angle brackets (< or >) are not allowed. |

**Step 4** On the VPC peering connection details page, click **Add Route**.

As shown in **Figure 3-9**, you need to configure container CIDR blocks for local and peer clusters. For details, see **Table 3-4**.

**Figure 3-9** Adding a route

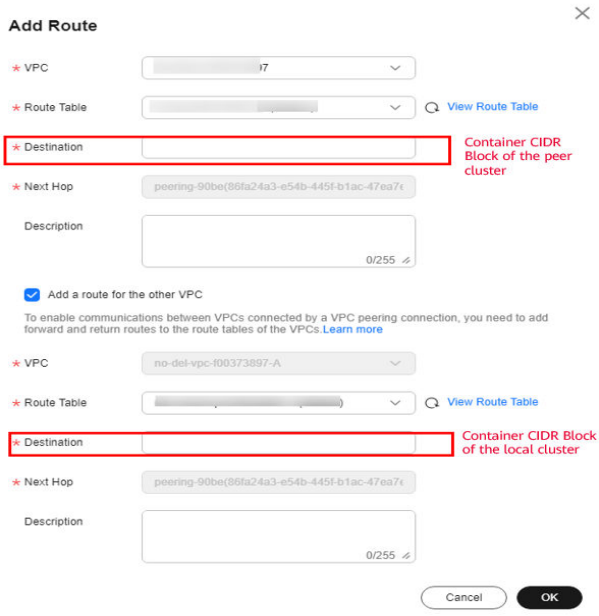
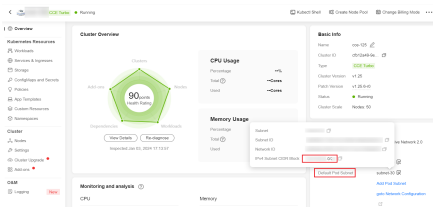



**Table 3-4** Route parameters

| Parameter | Mandatory | Description |
| --- | --- | --- |
| Destination | Yes | Enter the container CIDR block of the peer cluster.<br>To query this CIDR block:<br>1. Log in to the CCE console.<br>2. Click the name of the target cluster to access the cluster console. In the **Networking Configuration** area, hover over the name of **Default Pod Subnet** and copy the IPv4 CIDR block.<br>**CAUTION**<br>If there are multiple CIDR blocks, create a route for each CIDR block for communication between containers.<br>**Figure 3-10** Querying the container CIDR block of the peer cluster<br>  |
| Destination | Yes | Enter the container CIDR block of the local cluster.<br>**CAUTION**<br>The destination of each route must be unique. |

| Parameter | Mandatory | Description |
| --- | --- | --- |
| Description | No | Supplementary information about the route.<br>Enter up to 255 characters. Angle brackets (< or >) are not allowed. |

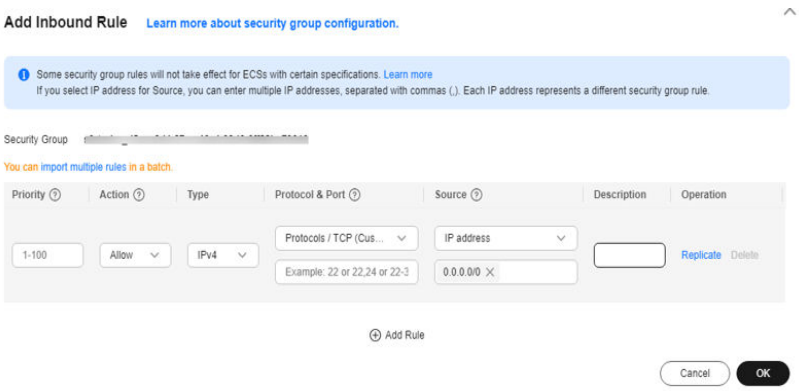
**----End**

## Changing a Security Group

Change the security group for the node in the local cluster to allow the node in the peer cluster to access over the local container port in the inbound rule.

Set **Protocol & Port** to the container port of the local cluster and **Source** to the IP address or CIDR block of the node in the peer cluster, as shown in **Figure 3-11**. For details about how to change the security group, see **Changing the Default Security Group of a Node**.

**Figure 3-11** Changing a security group




## Verifying Connectivity Between Clusters

**Step 1** Log in to the node in the local cluster and run the following command to verify the communication between the nodes in the local and peer clusters:

**ping** *IP address of the node in the peer cluster*

If the ping succeeds, the cluster connectivity is normal.

**Step 2** Access the container in the local cluster and run the following command to verify the communication between the containers in the local and peer clusters:

**curl** *IP address of the pod in the peer cluster*

If the access succeeds, the container connectivity is normal.

**----End**

# 3.3 Using Multi-Cluster Workload Scaling to Scale Workloads

## Application Scenarios

There are predictable and unpredictable traffic peaks for some services in complex scenarios. If you only use the standard FederatedHPA, it takes a long time to scale pods in workloads, which may make services unavailable during the expected peak hours. To abstract away this complexity, UCS provides two scaling policies, FederatedHPA and CronFederatedHPA, to automatically scale pods in workloads based on metric changes or at regular intervals.

This section uses hpa-example as an example to describe how you can use both FederatedHPA and CronFederatedHPA to scale workloads.

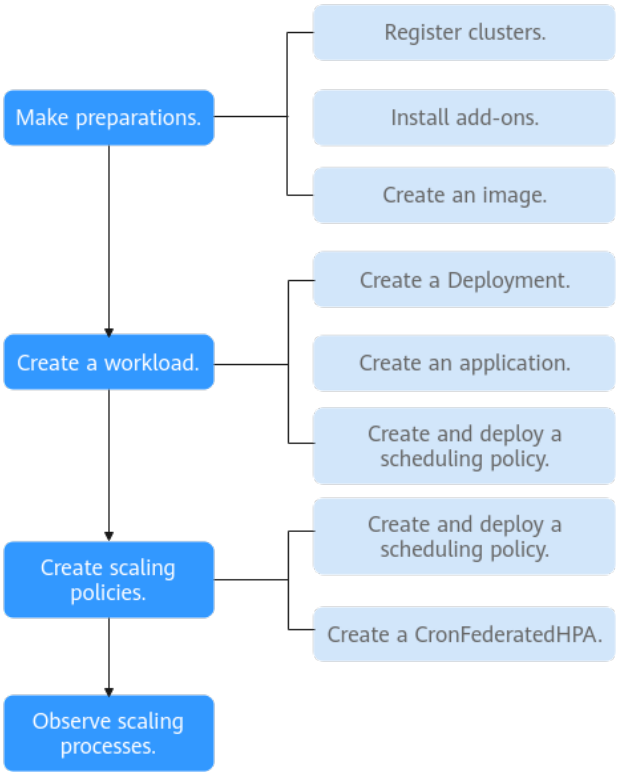
## Solution Process

**Figure 3-12** shows how to use both FederatedHPA and CronFederatedHPA.

1. Make preparations. Before creating workload scaling policies, prepare two Huawei Cloud clusters that have been registered with UCS, install Kubernetes Metrics Server for each cluster, and create an image named **hpa-example**.
2. Create a workload. Create a Deployment using the prepared image, create an application, and create and deploy a scheduling policy for the Deployment.
3. Create scaling policies. Use the command line tool to create a FederatedHPA and a CronFederatedHPA.
4. Observe scaling processes. View the number of pods in the Deployment and observe the effects of the scaling policies.

**Figure 3-12** Process of using both FederatedHPA and CronFederatedHPA




## Making Preparations

- Register two Huawei Cloud clusters (cluster01 and cluster02) with UCS. For details about how to register Huawei Cloud clusters with UCS, see **Huawei Cloud Clusters**.
- Install Kubernetes Metrics Server for the clusters. For details about how to install this add-on, see **Kubernetes Metrics Server**.
- Log in to the cluster node and deploy a compute-intensive application. When a user sends a request, the result needs to be calculated before being returned to the user. The following describes the details.

  a. Create a PHP file named **index.php** to calculate the square root of the request for 1,000,000 times before "OK!" is displayed.

  **vi index.php**

  The following provides an example **index.php**:

  ```
  <?php
    $x = 0.0001;
    for ($i = 0; $i <= 1000000; $i++) {
      $x += sqrt($x);
    }
    echo "OK!";
  ?>
  ```

  b. Compile a Dockerfile to create an image.

  **vi Dockerfile**

  The following provides an example Dockerfile:

  ```
  FROM php:5-apache
  COPY index.php /var/www/html/index.php
  RUN chmod a+rx index.php
  ```

c. Create an image named **hpa-example** with the **latest** tag.

**docker build -t hpa-example:latest .**

d. (Optional) Log in to the SWR console. In the navigation pane, choose **Organizations**. In the upper right corner, click **Create Organization**. Skip this step if you already have an organization.

e. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog box, click **Generate a temporary login command**. Then, click  to copy the command.

f. Run the login command copied in the previous step on the node. If the login is successful, "Login Succeeded" will be displayed.

g. Add a tag to the **hpa-example** image.

**docker tag** *{Image name 1:Tag 1} {Image repository address}/{Organization name}/{Image name 2:Tag 2}*

**Table 3-5** Tag parameters

| Parameter | Description |
| --- | --- |
| {Image name 1:Tag 1} | Replace them with the name and tag of the image to be uploaded. |
| {Image repository address} | Replace it with the domain name at the end of the login command in **e**. |
| {Organization name} | Replace it with the organization name created in **d**. |
| {Image name 2:Tag 2} | Replace them with the image name and tag to be displayed in the SWR image repository. |

The following is a command example:

**docker tag hpa-example:latest swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest**

h. Push the image to the image repository.

**docker push** *{Image repository address}/{Organization name}/{Image name 2:Tag 2}*

The following is a command example:

**docker push swr.ap-southeast-1.myhuaweicloud.com/cloud-develop/hpa-example:latest**

Check whether the following information is returned. If yes, the image push is successful.

```
6d6b9812c8ae: Pushed
...
fe4c16cbf7a4: Pushed
latest: digest: sha256:eb7e3bbd*** size: **
```

i. To view the pushed image, go to the SWR console and refresh the **My Images** page.

## Creating a Workload

**Step 1** Use the **hpa-example** image to create a Deployment with one pod. The image path varies with the SWR repository and needs to be replaced with the actual value.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hpa-example
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hpa-example
  template:
    metadata:
      labels:
        app: hpa-example
    spec:
      containers:
      - name: container-1
        image: 'hpa-example:latest'   # Replace it with the path of the image you uploaded to SWR.
        resources:
          limits:                     # Keep the value same as that of requests to prevent flapping during scaling.
            cpu: 500m
            memory: 200Mi
          requests:
            cpu: 500m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```

**Step 2** Create a Service with the port number being 80.

```
kind: Service
apiVersion: v1
metadata:
  name: hpa-example
spec:
  ports:
    - name: cce-service-0
      protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 31144
  selector:
    app: hpa-example
  type: NodePort
```

**Step 3** Create a scheduling policy for the Deployment and Service and deploy the Deployment and Service in cluster01 and cluster02, with the weight of each cluster being 1 to ensure that each cluster has the same priority.

```
apiVersion: policy.karmada.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: hpa-example-pp
  namespace: default
spec:
  placement:
    clusterAffinity:
      clusterNames:
      - cluster01
      - cluster02
```

```
  replicaScheduling:
    replicaDivisionPreference: Weighted
    replicaSchedulingType: Divided
    weightPreference:
      staticWeightList:
      - targetCluster:
          clusterNames:
          - cluster01
        weight: 1
      - targetCluster:
          clusterNames:
          - cluster02
        weight: 1
preemption: Never
propagateDeps: true
resourceSelectors:
- apiVersion: apps/v1
  kind: Deployment
  name: hpa-example
  namespace: default
- apiVersion: v1
  kind: Service
  name: hpa-example
  namespace: default
```

**----End**

## Creating Scaling Policies

**Step 1** Create a FederatedHPA.

**vi hpa-example-hpa.yaml**

As described in the YAML file, this policy is associated with the Deployment named **hpa-example**. The stabilization window is 0 seconds for a scale-out and 100 seconds for a scale-in. The maximum number of pods is 100 and the minimum number of pods is 2. This policy contains a system metric rule in which the desired CPU usage is 50%.

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: FederatedHPA
metadata:
  name: hpa-example-hpa                       # FederatedHPA name
  namespace: default                        # Namespace where the Deployment resides
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-example                       # Deployment name
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 100             # The stabilization window is 100 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: cpu                           # CPU-based scaling metrics
        target:
          type: Utilization                 # The metric type is resource usage.
          averageUtilization: 50            # Desired average resource usage
```

**Step 2** Create a CronFederatedHPA.

**vi cron-federated-hpa.yaml**

As described in the YAML file, this policy works with the FederatedHPA named **hpa-example-hpa** to scale out 10 pods at 08:30 and scale in 2 pods at 10:00 for the Deployment daily.

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: CronFederatedHPA
metadata:
  name: cron-federated-hpa                  # CronFederatedHPA name
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: FederatedHPA                      # CronFederatedHPA runs based on FederatedHPA.
    name: hpa-example-hpa                   # FederatedHPA name
  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 10 * * *                    # Time when the policy is triggered
    targetReplicas: 2                       # Desired number of pods, which is a non-negative integer
    timeZone: Asia/Shanghai                 # Time zone
```

**----End**

## Observing Scaling Processes

**Step 1** View the FederatedHPA. You can see that the CPU usage of the Deployment is 0%.

**kubectl get FederatedHPA hpa-example-hpa**

```
NAME              REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
hpa-example-hpa   Deployment/hpa-example   0%/50%    1         100       1          6m
```

**Step 2** Access the Deployment. In the following command, *{ip:port}* indicates the access address of the Deployment obtained from its details page.

**while true;do wget -q -O- http://*{ip:port}*; done**

**Step 3** Observe the automatic scale-out process of the Deployment.

**kubectl get federatedhpa hpa-example-hpa --watch**

View the FederatedHPA. You can see that the CPU usage of the Deployment is 200% at 6m23s, which exceeds the target value. In this case, the FederatedHPA is triggered to expand four pods for the Deployment. In the subsequent several minutes, the CPU usage does not decrease until 8m16s. This is because the new pods may not be successfully created. The possible cause is that resources are insufficient and the pods are in the **Pending** state. During this period, nodes are added.

At 8m16s, the CPU usage decreases, indicating that the pods are successfully created and start to bear traffic. The CPU usage decreases to 81% at 8m, still greater than the target value and beyond the tolerance range. So, 7 pods are added at 9m31s, and the CPU usage decreases to 51%, which is within the tolerance range. From then on, the number of pods remains 7.

```
NAME              REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-example-hpa   Deployment/hpa-example   0%/50%     1         100       1          6m
hpa-example-hpa   Deployment/hpa-example   200%/50%   1         100       1          6m23s
hpa-example-hpa   Deployment/hpa-example   200%/50%   1         100       4          6m31s
hpa-example-hpa   Deployment/hpa-example   210%/50%   1         100       4          7m16s
hpa-example-hpa   Deployment/hpa-example   210%/50%   1         100       4          7m16s
hpa-example-hpa   Deployment/hpa-example   90%/50%    1         100       4          8m16s
```

```
hpa-example-hpa   Deployment/hpa-example   85%/50%   1   100   4   9m16s
hpa-example-hpa   Deployment/hpa-example   51%/50%   1   100   7   9m31s
hpa-example-hpa   Deployment/hpa-example   51%/50%   1   100   7   10m16s
hpa-example-hpa   Deployment/hpa-example   51%/50%   1   100   7   11m
```

View the scaling event of the FederatedHPA, from which you can see the effective time of this policy.

**kubectl describe federatedhpa hpa-example-hpa**

**Step 4** Stop accessing the Deployment and observe its automatic scale-in process.

View the FederatedHPA. You can see that the CPU usage is 21% at 13m. The number of pods is reduced to 3 at 18m and then to 1 at 23m.

**kubectl get federatedhpa hpa-example-hpa --watch**

```
NAME              REFERENCE                TARGETS     MINPODS   MAXPODS   REPLICAS   AGE
hpa-example-hpa   Deployment/hpa-example   50%/50%     1         100       7          12m
hpa-example-hpa   Deployment/hpa-example   21%/50%     1         100       7          13m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       7          14m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       7          18m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          18m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          19m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          19m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          19m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          19m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          23m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       3          23m
hpa-example-hpa   Deployment/hpa-example   0%/50%      1         100       1          23m
```

View the scaling event of the FederatedHPA, from which you can see the effective time of this policy.

**kubectl describe federatedhpa hpa-example-hpa**

**Step 5** When the triggering time of the CronFederatedHPA arrives, observe the automatic scaling process of the Deployment.

The number of pods is increased to 4 at 118m and then to 10 at 123m.

**kubectl get cronfederatedhpa cron-federated-hpa --watch**

```
NAME                 REFERENCE                TARGETS     MINPODS   MAXPODS   REPLICAS   AGE
cron-federated-hpa   Deployment/hpa-example   50%/50%     1         100       1          112m
cron-federated-hpa   Deployment/hpa-example   21%/50%     1         100       1          113m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       4          114m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       4          118m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       4          118m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       4          119m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       7          119m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       7          119m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       7          119m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       7          123m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       10         123m
cron-federated-hpa   Deployment/hpa-example   0%/50%      1         100       10         123m
```

View the scaling event of the CronFederatedHPA, from which you can see the effective time of this policy.

**kubectl describe cronfederatedhpa cron-federated-hpa**

**----End**

# 3.4 Using MCI to Distribute Traffic Across Clusters

## Application Scenarios

Distributed clusters are often deployed on the clouds or regions nearest to users for low latency. However, if a cluster in a region is faulty, services in that region will be affected. MCI can be used to distribute traffic across clusters in different regions for cross-region failovers.

## Preparations

- Prepare two CCE Turbo clusters of v1.21 or later or Kubernetes clusters whose network model is underlay and deploy them in different regions.
- Plan the regions where applications are to be deployed and purchase a load balancer for each region. To ensure cross-region DR, the load balancers must be deployed across regions. Each load balancer must be a dedicated one of the application type (HTTP/HTTPS) and support the private network (with a private IP address), with the cross-VPC backend function enabled. For details, see **Creating a Dedicated Load Balancer**.
- Connect ELB VPCs to Kubernetes clusters so that load balancers can communicate with pods and the CIDR blocks of member clusters do not conflict with each other.
- Prepare Deployments and Services available in the federation. If no Deployment or Service is available, create ones by referring to **Deployments** and **ClusterIP**.

## Cross-Region Failover Through MCI

This section uses CCE Turbo clusters cce-cluster01 and cce-cluster02 as an example to describe how to enable public network access to services across regions and verify the cross-region DR of applications. This will be achieved by MCI objects that are associated with load balancers in different regions and DNS resolution provided by Huawei Cloud.

**Step 1** Register clusters with UCS, connect them to the network, and add them to a fleet. For details, see **Registering a Cluster**.

**Step 2** Enable cluster federation for the fleet and ensure that the clusters have been connected to a federation. For details, see **Cluster Federation**.

**Step 3** Create workloads and configure Services.

The following uses the **nginx** image as an example to describe how to deploy **nginx** workloads in clusters **cce-cluster01** and **cce-cluster-02** and configure Services.

**Step 4** Create a load balancer in each region.

In the network configuration, enable the IP backend (cross-VPC backend) function, select the VPC where **cce-cluster01** resides, and create an EIP. Record the ID of each load balancer.

**Step 5** Obtain the project ID of each region.

On the Huawei Cloud console, choose the account name in the upper right corner and click **My Credentials** to query the project ID of each region.

**Step 6** Use kubectl to connect to the federation. For details, see **Using kubectl to Connect to a Federation**.

**Step 7** Create and edit the **mci.yaml** file of each region.

Create MCI objects. The file content is defined as follows. For details about the parameters, see **Using MCI**.

**kubectl apply -f mci.yaml**

```
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterIngress
metadata:
  name: nginx-ingress-region1
  namespace: default
  annotations:
    karmada.io/elb.id: xxxxxxx # ID of the load balancer in region 1
    karmada.io/elb.port: " 80" # Listener port of the load balancer in region 1
    karmada.io/elb.projectid: xxxxxxx # Project ID of the tenant in region 1
    karmada.io/elb.health-check-flag: " on" # Health check is enabled for traffic switchover.
spec:
  ingressClassName: public-elb
  rules:
  - host: demo.localdev.me
    http:
      paths:
      - backend:
          service:
            name: nginx
            port:
              number: 8080
        path: /
        pathType: Prefix
---
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterIngress
metadata:
  name: nginx-ingress-region2
  namespace: default
  annotations:
    karmada.io/elb.id: xxxxxxx # ID of the load balancer in region 2
    karmada.io/elb.port: " 801" # Listener port of the load balancer in region 2
    karmada.io/elb.projectid: xxxxxxx # Project ID of the tenant in region 2
    karmada.io/elb.health-check-flag: " on" # Health check is enabled for traffic switchover.
spec:
  ingressClassName: public-elb
  rules:
  - host: demo.localdev.me
    http:
      paths:
      - backend:
          service:
            name: nginx
            port:
              number: 8080
        path: /
        pathType: Prefix
```

**Step 8** Check whether the backend server group is attached to the ELB listener, whether the backend instance is running, and whether the health check is normal.

> **CAUTION**
>
> Enable the security group for containers in advance. Take a CCE Turbo cluster as an example. Choose **Overview** > **Network Configuration** > **Default Security Group** and enable the CIDR block of the load balancer in the other region.

**----End**

## Configuring DNS Access

This section uses the private DNS server on Huawei Cloud as an example. You can also configure the DNS server by yourself.

**Step 1** Create a private DNS server and access the corresponding service over the public network on the ECS console. Associate an EIP or NAT gateway with the ECS instance to allow this ECS to access the public network.

- Create a private domain name in the same VPC as the ECS. The domain name is specified in the MCI object.
- Add the EIP of each load balancer to the record set of each cluster.

**Step 2** On the ECS console, use **curl demo.localdev.me** to access the corresponding service. If 200 is returned, the service access is normal.

**----End**

## Verifying the Cross-Region Failover

The example applications are deployed in clusters ccecluster-01 and ccecluster-02 and EIPs are provided for accessing corresponding services.

**Fault simulation**

The following uses the fault in region 1 as an example. Perform the following operations to simulate a single-region fault:

**Step 1** Hibernate the cce-cluster01 cluster in region 1 and stop the nodes in the cluster.

**Step 2** Disassociate EIP 1 from the load balancer in region 1.

**----End**

**DR verification**

**Step 1** On the DNS resolution page, manually delete the IP address associated with the load balancer in region 1 from the record set.

**Step 2** Check whether backend servers whose health check results are abnormal are displayed.

**Step 3** Access the corresponding service on the ECS console and check whether the service can be accessed and whether 200 is returned.

**----End**

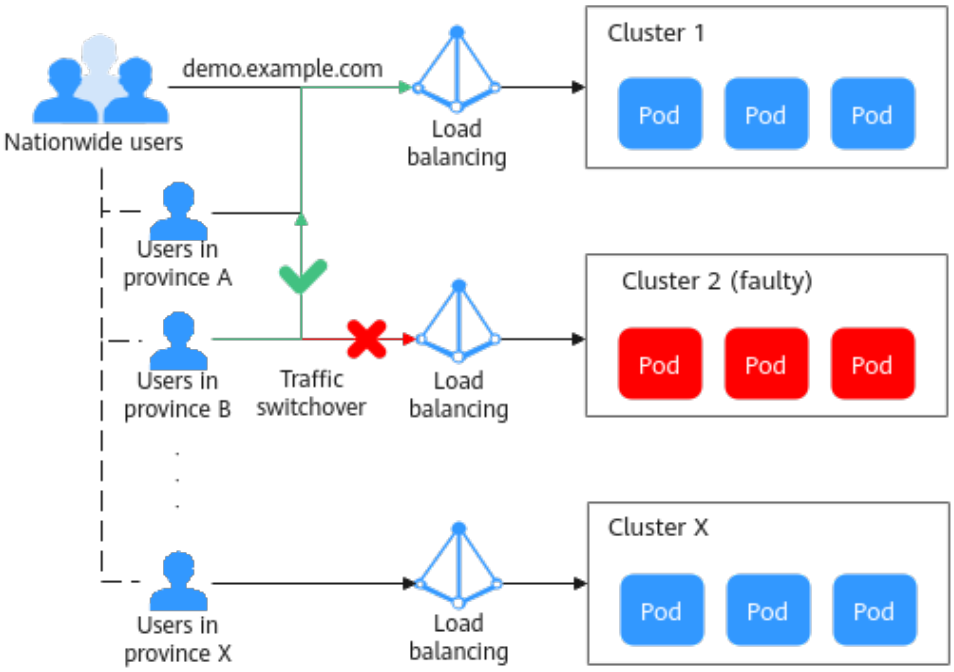
# 4 Traffic Distribution

## 4.1 Using Traffic Distribution to Implement Traffic Switchover

### Application Scenarios

Distributed clusters are often deployed on the clouds or regions nearest to users for low latency. However, if a cluster in a region is faulty, service access in that region will be affected. UCS allows you to manage application traffic and data for traffic switchover, scheduling, and migration across clouds and clusters. **Figure 4-1** shows how UCS switches traffic from a faulty cluster to ensure service continuity.

**Figure 4-1** Traffic switchover in multi-cloud clusters



## Constraints

- You have two available clusters of version 1.19 or later, and each cluster must have at least one available node.
- You have added a public zone to Huawei Cloud DNS. For details, see **Routing Internet Traffic to a Website**.

## Setting Up the Environment

**Step 1** Register clusters to UCS and configure cluster access. For details, see **Registering a Cluster**.

For example, register **ccecluster01** and **ccecluster02** to UCS and check whether they are running normally.

**Step 2** Create a workload in each connected cluster.

NOTE

In this example, container images with different tags are used to create workloads, aiming to show you more clearly how application traffic is switched.

- **ccecluster01**: Image tag 1.0.0 is used.
- **ccecluster02**: Image tag 2.0.0 is used.

**Step 3** Create a LoadBalancer Service for the workloads in each cluster.

NOTE

Only LoadBalancer Services are supported and displayed.

**Step 4** Use a browser to access the IP of the LoadBalancer Service to check the deployment result.

**----End**

## Verifying Traffic Switchover

You have deployed applications in clusters **ccecluster01** and **ccecluster02** and allowed external access via LoadBalancer Services.

The following describes how to distribute application traffic to realize traffic switchover and ensure high availability.

NOTE

In this example, example clusters and workloads are not limited in terms of service providers, regions, and quantity.

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

**Step 2** Click **Create Traffic Policy**. In the window that slides from the right, enter the domain name (use **demo.example.com** here).

**Figure 4-2** Creating a traffic policy


**Step 3** Add scheduling policies for the two clusters and click **OK**.

In this example, three scheduling policies are added to simulate cluster application deployment in different regions:

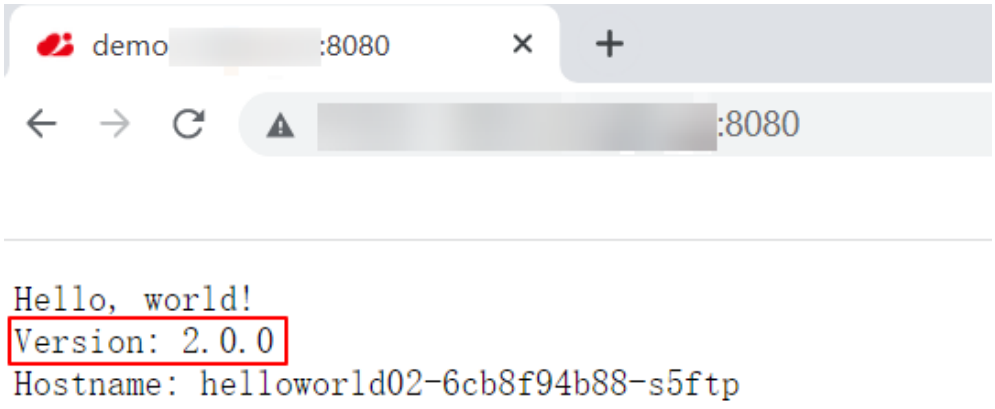
- For **ccecluster01**, set **Line Type** to **Region line - Global/Asia-Pacific/Singapore**.
- For **ccecluster02**, set **Line Type** to **Region line - Chinese Mainland/South China/Guangdong**.
- Add a default line for the domain name. For **ccecluster01**, set **Line Type** to **Default**. If no default line is added for the domain name, users in regions beyond the specified lines cannot access the applications.

**Step 4** View the scheduling policies. Three policies have been added for **demo.example.com**. User traffic can access applications in the two clusters based on the configured line type and weight.

- Users in Singapore will access application 1.0.0 in **ccecluster01**.
- Users in Guangdong will access application 2.0.0 in **ccecluster02**.
- Users in other regions will access application 1.0.0 in **ccecluster01** by default.

**Step 5** A user in Guangdong visits **demo.example.com** to access the application. The response shows that the user is accessing application 2.0.0 in **ccecluster02**.

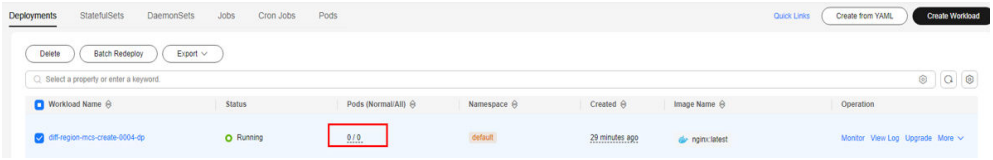
**Figure 4-3** Checking the access result




**Step 6** Manually stop the application in **ccecluster02** and change the number of pods to 0 to simulate a fault.

**Figure 4-4** Adjusting the pod quantity




**Step 7** When a user in Guangdong accesses the application, the request is still sent to **ccecluster02** and an error is returned.

In this case, click **Suspend** for the corresponding scheduling policy of **ccecluster02** on the **Traffic Distribution** page to perform traffic switchover.

After that, the subsequent access requests of this user will be sent to **ccecluster01** through the default line, and the access becomes normal. After the faulty cluster is recovered, you can click **Enable** to enable the policy again.

**----End**