Cloud Container Instance (CCI 2.0)

User Guide

Issue	01
Date	2025-08-15

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 Permissions Management.. 1
1.1 Permissions Management for CCI 2.0.. 1
1.2 Creating a User and Granting Permissions.. 1
1.3 CCI 2.0 Custom Policies.. 2
1.4 Delegating a Federated User to Manage Resources.. 4
2 Environment Configuration..6
3 Namespaces.. 11
3.1 Creating a Namespace.. 11
3.2 Creating a Namespace in a Shared VPC.. 15
4 Using CCI Through the Console.. 17
4.1 Workload Management.. 17
4.1.1 Deployments.. 17
4.1.2 Pods.. 24
4.1.2.1 Overview.. 24
4.1.2.2 Creating a Pod.. 25
4.1.2.3 Pod Flavor.. 29
4.1.2.3.1 Upgrading Pod Flavors.. 29
4.1.2.3.2 Reserved System Overhead.. 34
4.1.2.3.3 Increasing Reserved System Overhead.. 36
4.1.3 Viewing Resource Usages.. 38
4.1.4 Lifecycle.. 38
4.1.4.1 Startup Command.. 38
4.1.4.2 PostStart/PreStop Processing.. 40
4.1.5 Health Check.. 41
4.1.6 Web Terminal.. 44
4.1.7 Workload Upgrade.. 45
4.1.8 Workload Scaling.. 46
4.2 Network Management.. 49
4.2.1 Services.. 49
4.2.1.1 Service Overview.. 49
4.2.1.2 Private Network Access.. 50
4.2.1.3 Public Network Access.. 52

4.2.1.4 Configuring HTTP/HTTPS for a LoadBalancer Service 53
4.2.1.4.1 Configuring HTTP for a LoadBalancer Service 54
4.2.1.4.2 Configuring HTTPS for a LoadBalancer Service 57
4.2.1.5 Configuring an Access Policy for a Service 60
4.2.2 Gateways 62
4.2.2.1 PoolBindings 62
4.2.2.1.1 PoolBinding Overview 63
4.2.2.1.2 Associating a Deployment with a PoolBinding 64
4.2.2.1.3 Associating a Service with a PoolBinding 66
4.2.3 Specifying a Subnet for a Pod 68
4.3 Auto Scaling 69
4.3.1 Horizontal Scaling 69
4.4 Storage Management 71
4.4.1 Overview 71
4.4.2 Ephemeral Storage 72
4.4.3 SFS Turbo Volumes 74
4.4.4 OBS Parallel File System Volumes 80
4.4.4.1 Parallel File System Overview 80
4.4.4.2 Setting Access Keys (AK/SK) for Mounting a Parallel File System Volume 82
4.4.4.3 Creating a Volume from an Existing Parallel File System 84
4.4.4.4 Updating the Access Keys (AK/SK) for a Parallel File System Volume 90
4.4.4.5 Setting Mount Options for a Parallel File System Volume 91
4.5 Configuration Center 93
4.5.1 ConfigMaps 93
4.5.2 Secrets 95
4.6 Images 97
4.6.1 Image Snapshots 97
4.6.1.1 Overview 97
4.6.1.2 Creating an Image Snapshot 98
4.6.1.3 Using an Image Snapshot 101
4.6.1.4 Managing Image Snapshots 104
4.6.2 Pulling an Image from a Self-Managed Image Repository 104
4.7 Monitoring 107
4.8 Event Management 115
4.8.1 Enabling Event Reporting 115
4.8.2 Configuring Alarm Rules on the LTS Console 118
4.8.3 Event List 120
4.9 Log Management 124
4.9.1 Log Collection 124
4.9.2 Mounting Standard Output Logs 133
5 Using CCI with CCE 136
5.1 Overview 136

5.2 Quick Start.. 143
5.3 Scheduling Workloads to CCI 2.0.. 147
5.4 Rounding Up Pod Resources... 154
5.5 Images.. 155
5.6 Storage... 158
5.7 Logging.. 162
5.7.1 Collecting Logs Using LTS.. 162
5.7.2 Collecting Logs Using a Sidecar Container.. 169
5.8 Configuring a ClusterExtensionProfile or ExtensionProfile... 172
5.9 Using a Sidecar Container.. 175
5.9.1 Configuring a Sidecar Container.. 175
5.9.2 Replicating Environment Variables and Volume Mount Points of a Service Container........................ 178
5.10 Networking... 182
5.10.1 Configuring Networking... 182
5.10.2 Specifying Default DNS Servers.. 189
5.10.3 Configuring Subnets... 190
5.11 Monitoring... 192
5.12 Auto Scaling.. 192
5.13 Virtual Node Configurations.. 195
5.14 Troubleshooting.. 196

6 O&M Management.. 199

6.1 Configuring Core Dump.. 199
6.2 Configuring a Header for Images.. 200

7 Cost Tag Management... 202

8 Auditing... 205

8.1 CCI Operations Supported by CTS.. 205
8.2 Viewing a Trace.. 207

1 Permissions Management

1.1 Permissions Management for CCI 2.0

CCI 2.0 permissions management allows you to grant permissions to your IAM users and user groups. It works with Identity and Access Management (IAM) to provide a variety of authorization methods, including IAM fine-grained authorization, IAM token authorization, namespace authorization, and resource authorization in namespaces.

- **CCI permissions:** permissions granted based on IAM fine-grained authorization. You can authorize users to perform operations on namespaces, such as creating and deleting namespaces.

1.2 Creating a User and Granting Permissions

This section describes how to use **IAM** to implement fine-grained permissions control for your CCI 2.0 resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing CCI 2.0 resources.
- Grant users only the permissions required to perform a given task based on their job responsibilities.
- Entrust an account or cloud service to perform efficient O&M on your CCI 2.0 resources.

If your account does not require individual IAM users, skip this section.

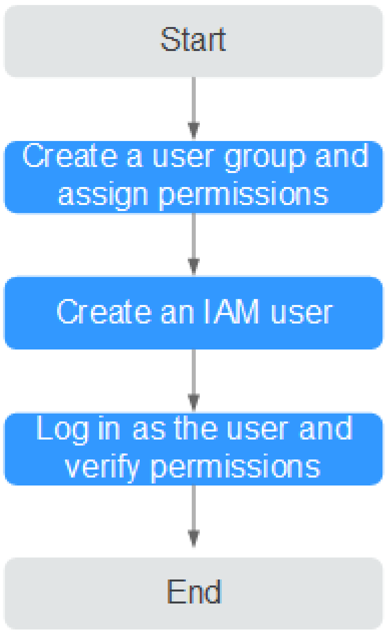
The following is the procedure for granting permissions (see **Figure 1-1**).

Prerequisites

You have learned about the permissions supported by CCI 2.0.

Process Flow

Figure 1-1 Process of granting CCI 2.0 permissions


Start
Create a user group and assign permissions
Create an IAM user
Log in as the user and verify permissions
End

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

   Create a user group (for example, **Developers**) on the IAM console and assign the **CCI CommonOperations** policy to the group. CCI 2.0 is a project-level service. When assigning system-defined policies to users, you also need to assign the **IAM ReadOnlyAccess** policy to the users.

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

   Create a user (for example, **James**) on the IAM console and add the user to the group created in **1**.

3. **Log in as the user you created** and verify permissions.

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

   – Choose **Service List** > **Cloud Container Instance 2.0**. In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create Deployment**. If the Deployment is created successfully, the **CCI CommonOperations** policy has taken effect.

   – Choose **Service List** > **Cloud Container Instance 2.0**. In the navigation pane, choose **Namespaces**. On the page displayed, click **Create Namespace**. If the namespace cannot be created, the **CCI CommonOperations** policy has taken effect.

1.3 CCI 2.0 Custom Policies

You can create custom policies to supplement the system-defined policies of CCI 2.0.

To create a custom policy, choose either visual editor or JSON.

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a policy in the JSON format from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. The following section contains examples of common CCI custom policies.

Example Custom Policies

- Example 1: Updating a namespace

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cci:namespace:update"
            ]
        }
    ]
}
```

- Example 2: Denying namespace deletion

  A policy with only "Deny" permissions must be used in conjunction with other policies for it to take effect. If you assign both "Allow" and "Deny" to a user, the "Deny" permissions take precedence over the "Allow" permissions.

  The following method can be used if you need to assign permissions of the **CCIFullAccess** policy to a user but you want to prevent the user from deleting namespaces (**cci:namespace:delete**). Create a custom policy for denying namespace deletion, and attach both policies to the group that the user belongs to. Then, the user can perform all operations on CCI 2.0 except deleting namespaces. The following is an example of a deny policy:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "cci:namespace:delete"
            ],
            "Effect": "Deny"
        }
    ]
}
```

- Example 3: Defining permissions for multiple services in a policy

  A custom policy can contain the actions of multiple services that are of the global or project-level type. The following is an example policy containing actions of multiple services:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ecs:cloudServers:resize",
                "ecs:cloudServers:delete",
                "ims:images:list",
                "ims:serverImages:create"
            ],
            "Effect": "Allow"
        }
```

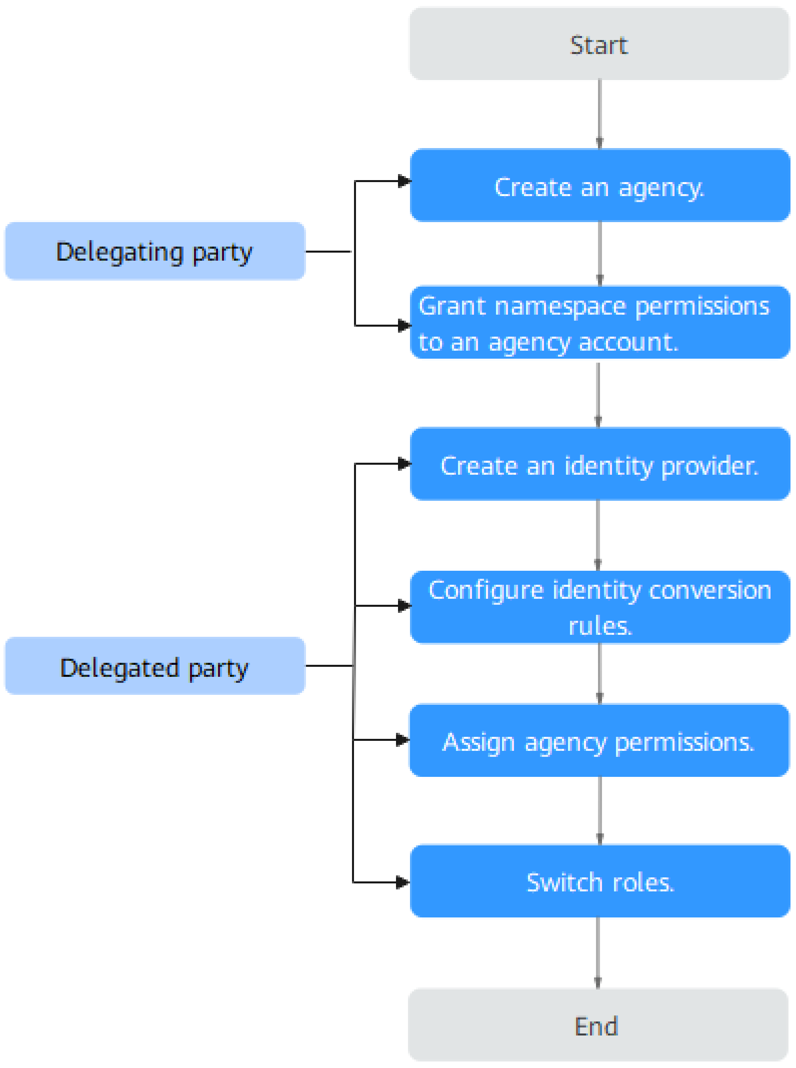
```
  ]
}
```

1.4 Delegating a Federated User to Manage Resources

If you want to delegate a federated user of another account (account B) to manage resources in your account (account A), log in using account A, create an agency for account B, and grant namespace permissions to account B. Then, log in using account B and perform federated identity authentication for it. After the authentication is complete, account B assigns the agency permissions to the federated user so that the federated user can switch to the agency of account A. Log in to Huawei Cloud as the federated user and switch the role to manage resources in account A.

This section describes how to delegate federated users to manage resources. **Figure 1-2** shows the operation process.

Figure 1-2 Process of delegating a federated user to manage resources


Start
Create an agency.
Delegating party
Grant namespace permissions to an agency account.
Create an identity provider.
Configure identity conversion rules.
Delegated party
Assign agency permissions.
Switch roles.
End

Procedure

To delegate account B to manage resources in account A as a federated user, take the following steps:

Step 1 Create an agency (by the delegating party).

Log in to the IAM console as the delegating party (account A). Create an agency, enter the account name of the delegated party (account B), and grant permissions of the **CCIFullAccess** policy to the delegated party. Users granted these permissions can create, delete, query, and update all CCI 2.0 resources.

Step 2 Perform the federated identity authentication as the delegated party.

Log in to as the delegated party (account B) and perform federated identity authentication.

Before delegating a federated user to manage resources, you need to perform federated identity authentication on the delegated party. The authentication process consists of two steps: establish a trust relationship and create an identity provider, and then configure identity conversion rules.

NOTE

After an identity provider is created, a default identity conversion rule is also created. You need to click **Edit Rule** to update or delete the default rule and create one. If you add a new rule without deleting the default rule, the default rule may be matched, and the new rule does not take effect.

Step 3 Assign permissions to a user (by the delegated party).

If a user under the delegated party (account B) wants to manage account A's resources, the delegated party (account B) must assign agency permissions to the user. To enable a federated user to manage resources of the delegating party (account A), the delegated party (account B) needs to assign the permissions of the custom policy **federation_agency** to the user group (**federation_group**) that the federated user belongs to. **federation_group** is the user group that the federated user belongs to. It is user-defined and written into the identity conversion rules.

Step 4 Switch roles (by the delegated party).

Account B and the federated user with agency permissions can switch their roles to the delegating party (account A) to manage its resources.

----End

2 Environment Configuration

Purchasing VPC Endpoints

VPC endpoints are required for accessing cloud services that use the network segment starting with 100.

- To pull images from a repository of SWR Enterprise Edition, you need to purchase a VPC endpoint for accessing OBS.
- To pull images from an SWR public image repository, you need a VPC endpoint for accessing SWR and a VPC endpoint for accessing OBS in the VPC where the workload is deployed.

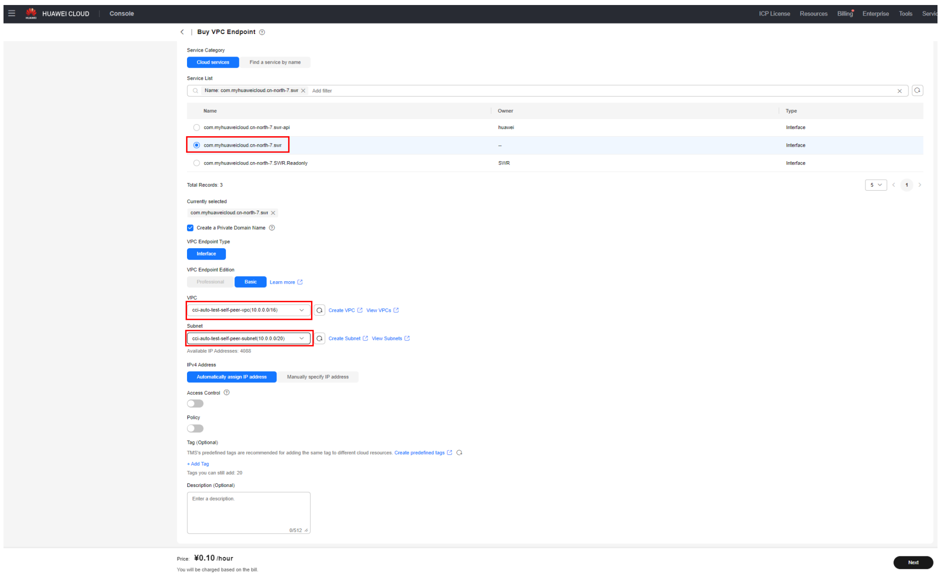
Step 1 Go to the **VPC endpoint list** page.

Step 2 On the **VPC Endpoints** page, click **Buy VPC Endpoint**.

The **Buy VPC Endpoint** page is displayed.

Step 3 Configure the parameters. Both the VPC endpoint for accessing SWR and the VPC endpoint for accessing OBS work only in the VPC where they are created.

Figure 2-1 Buying a VPC endpoint for accessing SWR



Buy VPC Endpoint

Service Category: Cloud services

Service List

Name: com.myhuaweicloud.cn-north-7.swr

	Name	Owner	Type
○	com.myhuaweicloud.cn-north-7.swr-api	huawei	Interface
●	com.myhuaweicloud.cn-north-7.swr	--	Interface
○	com.myhuaweicloud.cn-north-7.SWR.Readonly	SWR	Interface

Total Records: 3

Currently selected
com.myhuaweicloud.cn-north-7.swr

✓ Create a Private Domain Name

VPC Endpoint Type: Interface

VPC Endpoint Edition: Basic

VPC: cci-auto-test-self-peer-vpc(10.0.0.0/16)

Subnet: cci-auto-test-self-peer-subnet(10.0.0.0/20)

Available IP Addresses: 4088

IPv4 Address: Automatically assign IP address

Access Control

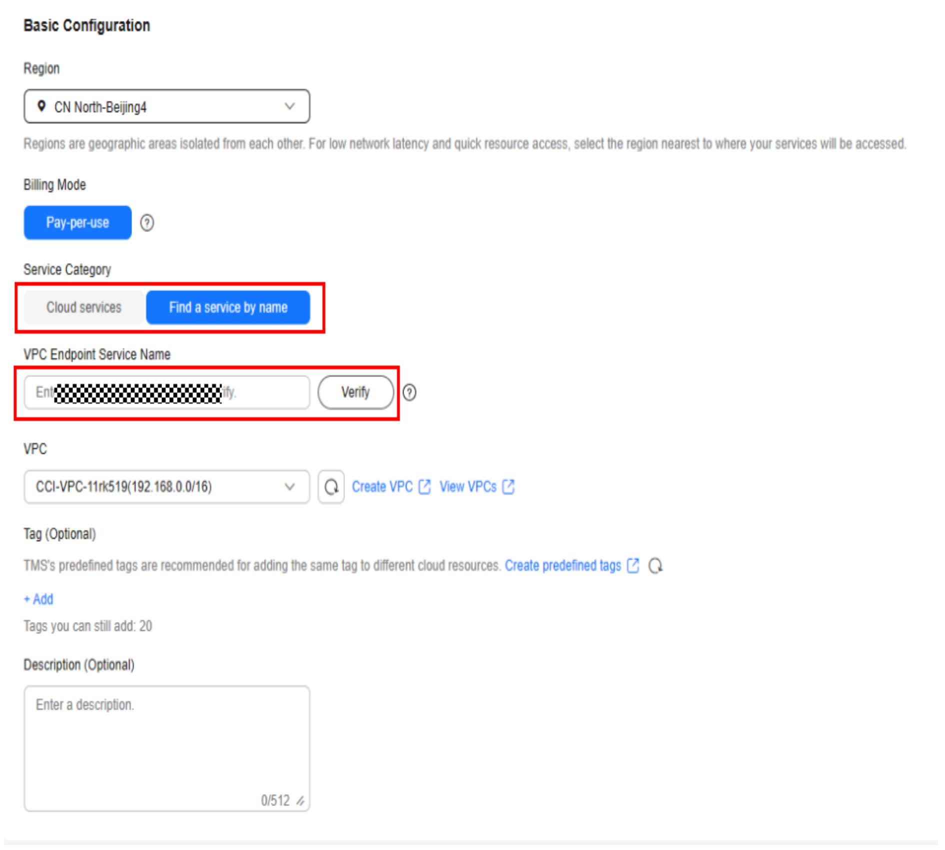
Policy

Tag (Optional)
TMS's predefined tags are recommended for adding the same tag to different cloud resources.
Tags you can still add: 20

Description (Optional)
Enter a description.
0/512

Price: ¥0.10 /hour
You will be charged based on the bill.

Figure 2-2 Buying a VPC endpoint for accessing OBS



Basic Configuration

Region

CN North-Beijing4

Regions are geographic areas isolated from each other. For low network latency and quick resource access, select the region nearest to where your services will be accessed.

Billing Mode

Pay-per-use

Service Category

Cloud services | Find a service by name

VPC Endpoint Service Name

Ent[redacted]ify. | Verify

VPC

CCI-VPC-11rk519(192.168.0.0/16) | Create VPC | View VPCs

Tag (Optional)

TMS's predefined tags are recommended for adding the same tag to different cloud resources. Create predefined tags

+ Add

Tags you can still add: 20

Description (Optional)

Enter a description.

0/512

Table 2-1 VPC endpoint parameters

Parameter	Example	Description
Region	CN-Hong Kong	Specifies the region where the VPC endpoint will be used to connect a VPC endpoint service. Resources in different regions cannot communicate with each other over an intranet. For lower latency and quicker access, select the region nearest to your on-premises data center.
Billing Mode	Pay-per-use	Specifies the billing mode of the VPC endpoint. VPC endpoints can be used or deleted at any time. VPC endpoints support only pay-per-use billing based on the usage duration.

Parameter	Example	Description
Service Category	Cloud services	There are two options: ● **Cloud services**: Select this option if the VPC endpoint service to be accessed is a cloud service. ● **Find a service by name**: Select this option if the VPC endpoint service to be accessed is a private service of your own. **CAUTION** ● Select **Cloud services** when you buy a VPC endpoint for accessing SWR. ● Select **Find a service by name** when you buy a VPC endpoint for accessing OBS.
Service List	-	This parameter is available only when you select **Cloud services** for **Service Category**. VPC endpoint services have been created. You can select one of them. **NOTE** If you select **Find a service by name** for **Service Category** when you buy a VPC endpoint for accessing OBS, submit a service ticket to get the service name.
VPC	-	Specifies the VPC where the VPC endpoint is to be deployed.
Subnet	-	Specifies the subnet where the VPC endpoint is to be deployed.
Route Table	-	This parameter is available only when you create a VPC endpoint for connecting to a gateway VPC endpoint service. **NOTE** This parameter is available only in the regions where the route table function is enabled. You are advised to select all route tables. Otherwise, the access to the gateway VPC endpoint service may fail. Select a route table required for the VPC where the VPC endpoint is to be deployed. For details about how to add a route, see **Adding Routes to a Route Table** in the *Virtual Private Cloud User Guide*.

Parameter	Example	Description
Policy	-	Specifies the VPC endpoint policy. VPC endpoint policies are a type of resource-based policies. You can configure a policy to control which principals can use the VPC endpoint to access VPC endpoint services.
Tag	example_key1 example_value1	Specifies the tag that is used to classify and identify the VPC endpoint. The tag settings can be modified after the VPC endpoint is purchased
Description	-	Provides supplementary information about the VPC endpoint.

Table 2-2 Tag requirements for VPC endpoints

Parameter	Requirement
Tag key	● Cannot be left blank. ● Must be unique for each resource. ● Can contain a maximum of 36 characters. ● Cannot start or end with a space or contain special characters =*<>\,|/ ● Can contain only letters, digits, hyphens (-), and underscores (_).
Tag value	● Cannot be left blank. ● Can contain a maximum of 43 characters. ● Cannot start or end with a space or contain special characters =*<>\,|/ ● Can contain only letters, digits, hyphens (-), and underscores (_).

Step 4 Confirm the settings and click **Next**.

- If the configuration is correct, click **Submit**.
- If any parameter is incorrect, click **Previous** to modify it as needed and then click **Submit**.

Step 5 Click **Back to VPC Endpoint List** after the task is submitted.

Step 6 View the endpoint details by clicking each endpoint ID.

----End

Logging In to the CCI 2.0 Console

Log in to the CCI 2.0 console and grant CCI the permissions to access other cloud services.

Step 1 Log in to the management console.

Step 2 Click in the upper left corner to select the desired region.

CCI 2.0 is available in CN-Hong Kong, AP-Jakarta, TR-Istanbul, AF-Johannesburg, ME-Riyadh, LA-Mexico City2, LA-Sao Paulo1, AP-Bangkok, and AP-Singapore.

NOTE

CCI 2.0 does not allow you to create resources in sub-projects.

Step 3 Choose **Service List** > **Containers** > **Cloud Container Instance 2.0**.

Switch to the CCI 2.0 console.

Step 4 If this is the first time you are logging in to the CCI 2.0 console, click **Agree** to grant CCI 2.0 the permissions to access other cloud services.

After the permissions are granted, an agency named **cci_admin_trust** is created. You can view the agency on the IAM console.

----End

(Optional) Uploading Images

The cloud platform provides the SoftWare Repository for Container (SWR) service for you to upload container images to the image repository. You can easily pull these images when creating workloads on CCI 2.0. For details about how to upload images, see **Pushing an Image**.

NOTICE

- After **Enterprise Project Management Service (EPS)** is enabled, if an IAM user needs to use private images in your account, you need to log in to the CCI 2.0 console using the account, choose **Image Repository**, and grant the required permissions to the user on the SWR console.
- You can use either of the following methods to grant permissions to an IAM user:
  - On the details page of an image, click the **Permissions** tab, click **Add Permission**, and then grant the read, write, and manage permissions to the user. For details, see **Granting Permissions for a Specific Image**.
  - On the details page of an organization, click the **Users** tab, click **Add Permission**, and then grant the read, write, and manage permissions to the user. For details, see **Granting Permissions for an Organization**.

3 Namespaces

3.1 Creating a Namespace

Namespaces are used to logically divide your resources into different groups, especially in scenarios where a large number of users from multiple teams work on different projects.

CCI 2.0 provides general computing resources and allows you to create pods with vCPUs for general computing.

NOTE

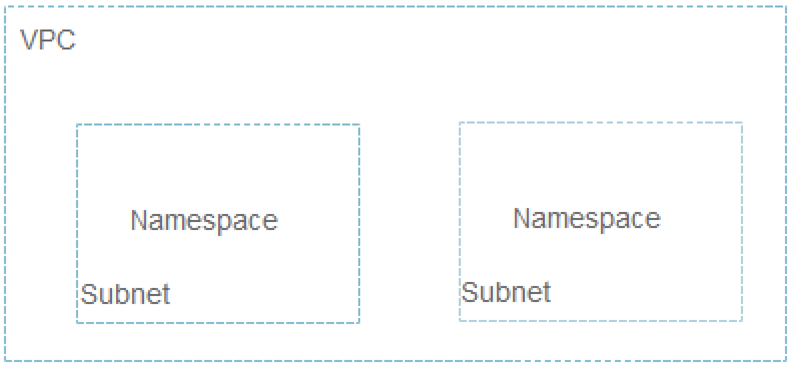
- One account can create a maximum of five namespaces in a region.
- x86 images are supported.

Relationship Between Namespaces and Networks

Each namespace requires a separate subnet, as shown in **Figure 3-1**. When you create a namespace, you need to associate it with a VPC. A subnet will be created for the namespace in the VPC. Containers and other resources created in this namespace will run in the VPC and subnet you select.

If you want to run resources of multiple services in the same VPC, you need to consider network planning, including subnet CIDR block division and IP address planning.

Figure 3-1 Relationship between namespaces and VPC subnets



Application Scenarios

Namespaces can implement partial environment isolation. If you have a large number of projects and personnel, you can create different namespaces based on project attributes, such as production, test, and development.

Creating a Namespace

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Namespaces**.

Step 3 On the **Namespaces** page, click **Create Namespace** in the upper right corner.

Step 4 Enter a name for the namespace.

NOTE

- The name of each namespace must be unique.
- Enter 1 to 63 characters starting and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.

Step 5 (Optional) Specify monitoring settings.

Parameter	Description
AOM (Optional)	If this option is enabled, you need to select an AOM instance.

Step 6 Configure the network plane.

Table 3-1 Network plane settings

Parameter	Description
IPv6	If this option is enabled, IPv4/IPv6 dual stack is supported.

Parameter	Description
VPC	Select the VPC where the workloads are running. If no VPC is available, create one first. The VPC cannot be changed once selected. Recommended CIDR blocks: 10.0.0.0/8-22, 172.16.0.0/12-22, and 192.168.0.0/16-22 **NOTICE** ● You cannot set the VPC CIDR block and subnet CIDR block to 10.247.0.0/16, because this CIDR block is reserved for workloads. If you select this CIDR block, there may be IP address conflicts, which may result in workload creation failure or service unavailability. If you do not need to access pods through workloads, you can select this CIDR block. ● After the namespace is created, you can choose **Namespaces** in the navigation pane and view the VPC and subnet in the **Subnet** column.
Subnet	Select the subnet where the workloads are running. If no subnet is available, create one first. The subnet cannot be changed once selected. ● A certain number of IP addresses (**10** by default) in the subnet will be warmed up for the namespace. ● You can set the number of IP addresses to be warmed up in **Advanced Settings**. ● If warming up IP addresses for the namespace is enabled, the VPC and subnet can only be deleted after the namespace is deleted. **NOTE** Ensure that there are sufficient available IP addresses in the subnet. If IP addresses are insufficient, workload creation will fail.
Security Group	Select a security group. If no security group is available, create one first. The security group cannot be changed once selected.

Step 7 (Optional) Specify advanced settings.

Each namespace provides an IP pool. You can specify the pool size to reduce the duration for assigning IP addresses and speed up the workload creation.

For example, 200 pods are running routinely, and 200 IP addresses are required in the IP pool. During peak hours, the IP pool instantly scales out to provide 65,535 IP addresses. After a specified interval (for example, 23 hours), the IP addresses that exceed the pool size (65535 – 200 = 65335) will be recycled.

Table 3-2 (Optional) Advanced namespace settings

Parameter	Description
IP Pool Warm-up for Namespace	• An IP pool is provided for each namespace, with the number of IP addresses you specify here. IP addresses will be assigned in advance to accelerate workload creation. • An IP pool can contain a maximum of 65,535 IP addresses. • When using general-computing pods, you are advised to configure an appropriate size for the IP pool based on service requirements to accelerate workload startup. • Configure the number of IP addresses to be assigned properly. If the number of IP addresses exceeds the number of available IP addresses in the subnet, other services will be affected.
IP Address Recycling Interval (h)	Pre-assigned IP addresses that become idle can be recycled within the duration you specify here. **NOTE** Recycling mechanism: • Recycling time: The **yangtse.io/warm-pool-recycle-interval** field configured on the network determines when the IP addresses can be recycled. If **yangtse.io/warm-pool-recycle-interval** is set to **24**, the IP addresses can only be recycled 24 hours later. • Recycling rate: A maximum of 50 IP addresses can be recycled at a time. This prevents IP addresses from being repeatedly assigned or released due to fast or frequent recycling.

Step 8 Click **OK**.

You can view the VPC and subnet on the namespace details page.

----End

Deleting a Namespace

NOTICE

Deleting a namespace will delete all resources (such as workloads, Services, pods, ConfigMaps, and secrets) related to the namespace.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Namespaces**.

Step 3 On the **Namespaces** page, locate the namespace you want to delete and click **Delete** in the **Operation** column. In the **Delete Namespace** dialog box, enter **DELETE** and click **OK**.

NOTE

To delete a VPC or subnet, go to the **VPC console**.

----End

3.2 Creating a Namespace in a Shared VPC

A shared VPC is a VPC that is shared among accounts through the Resource Access Manager (RAM) service. For example, you can share your VPC and subnets with another account so that this account can view the VPC and subnets and create resources for example, a CCI 2.0 namespace, in a shared subnet. For details, see **VPC Sharing Overview**.

Scenario

With VPC sharing, you can organize accounts in an orderly and centralized manner based on the organization structure or service form so that you can manage resources centrally and share them with other members to avoid repeated configurations. This helps you avoid repeated configurations and unify security and O&M for easier configurations of security policies.

Suppose that an enterprise IT account, the resource owner, creates a VPC and subnets and shares multiple subnets with other accounts.

- An enterprise service account (account A) creates resources in a shared subnet (subnet 1).
- Another enterprise service account (account B) creates resources in another shared subnet (subnet 2).

Constraints

- For clusters created in a shared VPC, load balancers cannot be shared.
- If a CCI 2.0 namespace has been created in a shared VPC, the owner of the shared VPC cannot delete the resource share, or the namespace will work abnormally.

Procedure

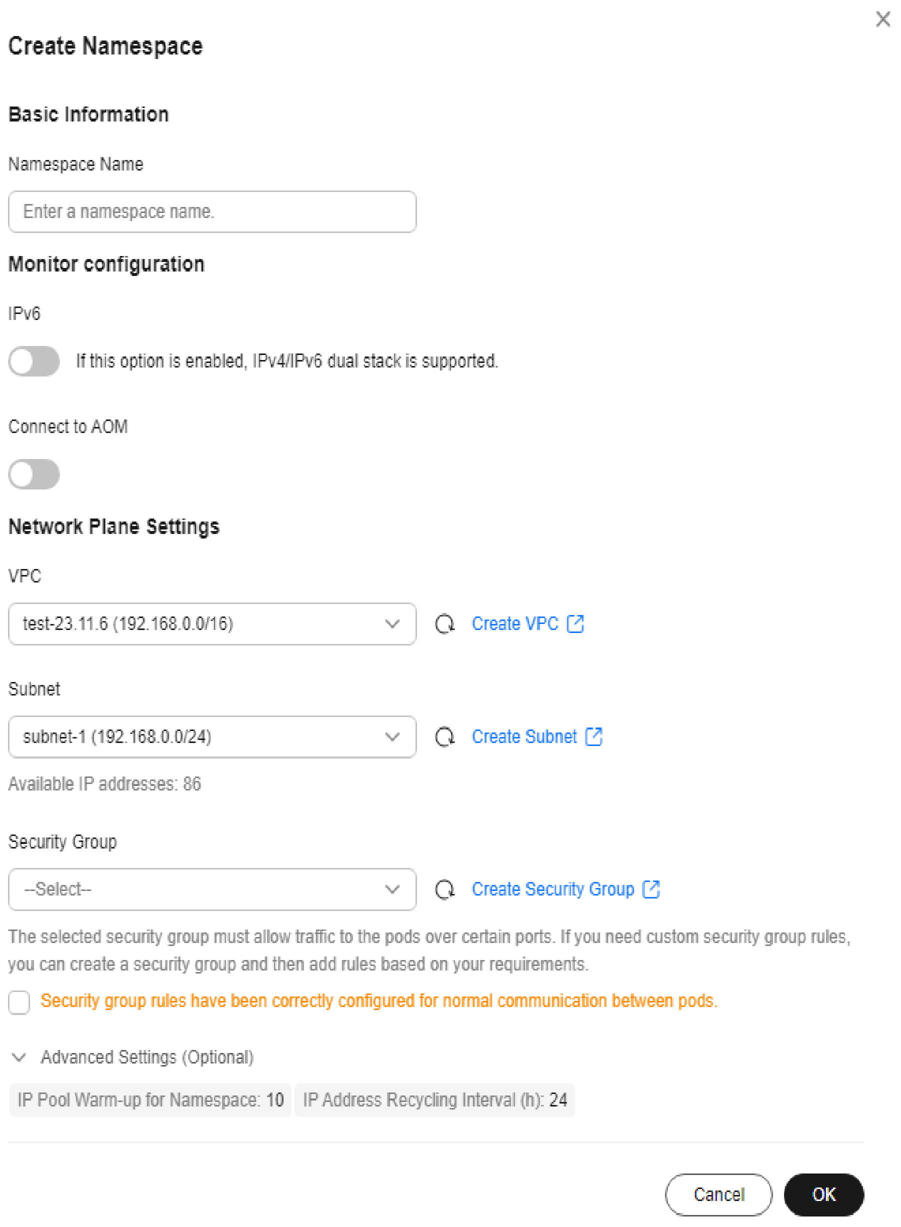
You share a VPC with another account (account B here as an example) to allow this account to select the shared VPC and subnets when creating a namespace.

1. You create a VPC share through RAM and specify account B as the resource principal. For details, see **Creating a Resource Share**.

   After the resource share is created, RAM sends an invitation to account B. Account B can access and use the shared VPC only after accepting the invitation.
2. Account B logs in to the **CCI 2.0 console** and create a namespace.

   Account B selects the VPC subnet you shared when configuring the network for the namespace. For details about other configurations, see **Creating a Namespace**.

Figure 3-2 Selecting a shared VPC subnet


Create Namespace

Basic Information

Namespace Name

Enter a namespace name.

Monitor configuration

IPv6

If this option is enabled, IPv4/IPv6 dual stack is supported.

Connect to AOM

Network Plane Settings

VPC

test-23.11.6 (192.168.0.0/16)

Subnet

subnet-1 (192.168.0.0/24)

Available IP addresses: 86

Security Group

--Select--

The selected security group must allow traffic to the pods over certain ports. If you need custom security group rules, you can create a security group and then add rules based on your requirements.

Security group rules have been correctly configured for normal communication between pods.

Advanced Settings (Optional)

IP Pool Warm-up for Namespace: 10 IP Address Recycling Interval (h): 24

4 Using CCI Through the Console

4.1 Workload Management

4.1.1 Deployments

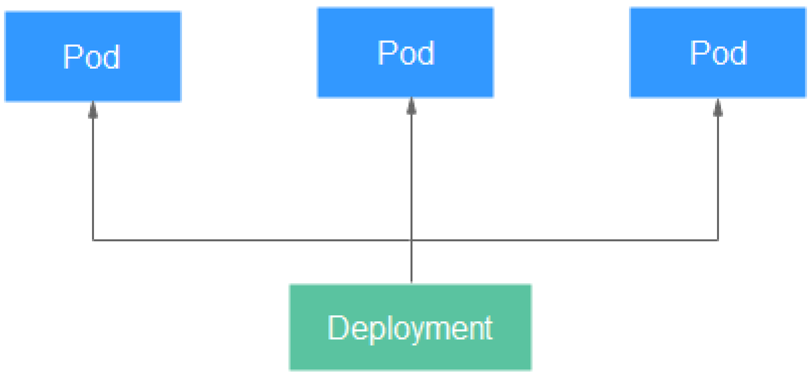
A Deployment is a service-oriented encapsulation of pods. A Deployment may manage one or more pods. These pods have the same role, and requests are distributed across the pods. All pods in a Deployment share the same volume.

Pods are the smallest deployable units. CCI provides controllers to manage pods. Controllers can create and manage pods, and provide replica management, rolling upgrade, and self-healing capabilities. The most commonly used controller is Deployment.

A Deployment can contain one or more pod replicas. These pod replicas have the same role, and requests are routed across the pod replicas.

A Deployment integrates a lot of functions, including rollout deployment, rolling upgrade, replica creation, and restoration of online jobs. To some extent, you can use Deployments to realize unattended rollout, which greatly reduces operation risks and improves rollout efficiency.

Figure 4-1 Deployment



Creating a Deployment on the Console

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create Deployment**.

Step 3 Add basic configuration.

Parameter	Description
Workload Name	Enter 1 to 63 characters starting and ending with a lowercase letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Do not enter consecutive periods or place a hyphen before or after a period. The workload name cannot be changed after the workload is created. If you need to change the name, create another workload.
Namespace	Select a namespace. If no namespaces are available, create one by following the steps provided in **Creating a Namespace**.
CPU Architecture	x86 or Kunpeng
Pod Type	**General** or **General-computing-lite** **NOTE** Only x86-based general-computing-lite pods are supported.
vCPUs	Select a value from **0.25** to **64**.
Memory	Select the memory based on the selected vCPUs.
Data Storage (Optional)	Only emptyDir volumes, ConfigMaps, and secrets are supported. Add a volume to the pod and then mount the volume to the specified container. Click **Add Data Store**, select a volume type, and enter a volume name. ● **emptyDir volume**: By default, CCI provides 30 GiB of free storage space, which is shared by emptyDir volumes and the system disk. ● **ConfigMap**: Select a ConfigMap. If no ConfigMaps are available, create one first. For details, see **Creating a ConfigMap**. ● **Secret**: Select a secret. If no secrets are available, create one first. For details, see **Creating a Secret**.
Pods	Specify the number of pods. A workload can have one or more pods. A pod can have one or more containers with the same flavor. Configure multiple pods for a workload if you want higher reliability. If one pod becomes faulty, the workload can still run normally.

Step 4 Add containers. A pod generally contains one or more containers created from different images. If your application needs to run on multiple containers in a pod,

click **Add Container** and then select an image each time when you need another container.

1. Click **Add Container**.
2. Specify basic information.

Table 4-1 Basic parameters

Parameter	Description
Container Name	Enter 1 to 63 characters starting and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
Image	Select an image. **NOTICE** If different containers in a pod listen to the same port, there will be port conflicts, and the pod may fail to start. For example, if a container created from an Nginx image (which listens to port 80) has been added to a pod, there will be a port conflict when another container created from an HTTP image in the pod tries to listen to port 80. – **My Images**: images you have pushed to SWR. **NOTE** ■ If you are an IAM user, you must obtain permissions before you can use the private images in the account. For details on how to obtain permissions, see **Uploading Images**. ■ Currently, CCI does not support third-party image repositories. ■ A single layer of a decompressed image cannot exceed 20 GiB. – **Shared Images**: images shared by others through SWR. – **Open Source Images**: public images in the image center.
Image Tag	Select a tag.
vCPUs	The value cannot be less than 0 or greater than the remaining quota.
Memory	The value cannot be less than 0 or greater than the remaining quota.

3. (Optional) Specify advanced settings.

- **Lifecycle**: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. For details, see **Lifecycle**.
- **Health Check**: Container health can be checked regularly when the container is running. For details about how to configure health checks, see **Health Check**.
- **Environment Variables**: You can manually set environment variables or add variable references. Environment variables provide flexibility in configuring workloads. The environment variables for which you have assigned values

during container creation will take effect upon container startup. This saves you the trouble of rebuilding the container image.

To manually set variables, enter the variable name and value.

To reference variables, set the variable name, reference type, and referenced value for each variable. The following variables can be referenced: **PodIP** (pod IP address), **PodName** (pod name), and **Secret**. For details about how to reference a secret, see **Secrets**.

- **Data Storage**: Volumes can be mounted to containers for data access in the containers.
- **Security Settings**: Users who run containers can be configured.

Step 5 Select an upgrade policy.

- **Upgrade Policy**: **Rolling upgrade** and **Replace** are available.
  - **Rolling upgrade**: New pods gradually replace old ones, and service traffic is evenly distributed to both old and new pods to maintain service continuity.

    Max Unavailable Pods: Maximum number of unavailable pods allowed in a rolling upgrade. If the number is equal to the total number of pods, services may be interrupted. (Minimum number of alive pods = The total number of pods – Maximum number of unavailable pods)
  - **Replace**: Old pods are deleted, and then new pods are created. Services are interrupted during the upgrade.

Step 6 Click **Create Deployment**.

In the workload list, if the workload status changes to **Running**, the workload is created. You can click the workload name to view the details and refresh the page to view the real-time workload status.

----End

Creating a Deployment Using YAML

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, click **Create from YAML**.

Step 3 Import or add a YAML file and click **OK** to create a Deployment.

The following is an example file:

- Resource description in the **deployment.yaml** file

```
apiVersion: cci/v2
kind: Deployment
metadata:
  annotations:
    description: ''
  labels: {}
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        vm.cci.io/pod-size-specs: 2.00_4.0
        resource.cci.io/pod-size-specs: 2.00_4.0
        metrics.alpha.kubernetes.io/custom-endpoints: '[{api:'''',path:'''',port:'''',names:''''}]'
        log.stdoutcollection.kubernetes.io: '{"collectionContainers": ["container-0"]}'
      labels:
        app: nginx
    spec:
      containers:
        - image: library/nginx:stable-alpine-perl
          name: container-0
          resources:
            limits:
              cpu: 2000m
              memory: 4096Mi
            requests:
              cpu: 2000m
              memory: 4096Mi
          command: []
          lifecycle: {}
      dnsPolicy: ''
      imagePullSecrets:
        - name: imagepull-secret
      dnsConfig: {}
  minReadySeconds: 0
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 0
      maxUnavailable: 1
```

- Resource description in the **deployment.json** file

```
{
    "apiVersion": "cci/v2",
    "kind": "Deployment",
    "metadata": {
        "annotations": {
            "description": ""
        },
        "labels": {},
        "name": "nginx"
    },
    "spec": {
        "replicas": 2,
        "selector": {
            "matchLabels": {
                "app": "nginx"
            }
        },
        "template": {
            "metadata": {
                "annotations": {
                    "vm.cci.io/pod-size-specs": "2.00_4.0",
                    "resource.cci.io/pod-size-specs": "2.00_4.0",
                    "metrics.alpha.kubernetes.io/custom-endpoints": "[{api:'',path:'',port:'',names:''}]",
                    "log.stdoutcollection.kubernetes.io": "{\"collectionContainers\": [\"container-0\"]}"
                },
                "labels": {
                    "app": "nginx"
                }
            },
            "spec": {
                "containers": [
                    {
                        "image": "library/nginx:stable-alpine-perl",
                        "name": "container-0",
                        "resources": {
                            "limits": {
                                "cpu": "2000m",
                                "memory": "4096Mi"
                            },
                            "requests": {
                                "cpu": "2000m",
                                "memory": "4096Mi"
                            }
                        },
                        "command": [],
                        "lifecycle": {}
                    }
                ],
                "dnsPolicy": "",
                "imagePullSecrets": [
                    {
                        "name": "imagepull-secret"
                    }
                ],
                "dnsConfig": {}
            }
        },
        "minReadySeconds": 0,
        "strategy": {
            "type": "RollingUpdate",
            "rollingUpdate": {
                "maxSurge": 0,
                "maxUnavailable": 1
            }
        }
    }
}
```

----End

Creating a Deployment Using ccictl

CCI 2.0 offers ccictl, a command-line tool that simplifies workload creation with a user experience closer to kubectl compared to the console.

Constraints

ccictl must be used to connect to CCI2.0.

Procedure

Step 1 Run **ccictl create namespace cci-test** to create a namespace.

Step 2 Run **ccictl create -f network.yaml** to create a network. The following is an example YAML file:

```
apiVersion: yangtse/v2
kind: Network
metadata:
  annotations:
    yangtse.io/domain-id: ${domain-id}
    yangtse.io/project-id: ${project-id}
  name: cci-network
  namespace: cci-test
spec:
  networkType: underlay_neutron
  securityGroups:
    - ${security-group-id}
  subnets:
    - subnetID: ${subnet-id}
```

Step 3 Run **ccictl apply -f deployment.yaml** to create a workload. The following is an example YAML file:

```
kind: Deployment
apiVersion: cci/v2
```

```
metadata:
  name: nginx
  namespace: cci-test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
          - containerPort: 80
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
      dnsPolicy: Default
```

Step 4 Run **ccictl apply -f service.yaml** to create a Service. The following is an example YAML file:

```
kind: Service
apiVersion: cci/v2
metadata:
  name: service-nginx
  namespace: cci-test
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: '${elb_id}'
spec:
  ports:
    - name: service-nginx-port
      protocol: TCP
      port: 80
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

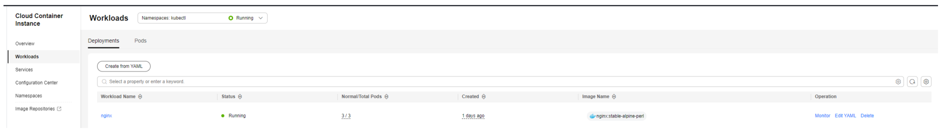
Step 5 Use the EIP and listening port of the load balancer to access the Nginx service.

----End

Updating a Deployment

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the deployment you want to update and click **Edit YAML** in the **Operation** column.



Workload Name	Status	Normal/Total Pods	Created	Image Name	Operation
nginx	Running	3 / 3	1 days ago	nginx:stable-alpine-perl	Monitor Edit YAML Delete

Step 3 Edit the YAML file and click **OK** to update the Deployment.

----End

Deleting a Pod

You can manually delete pods. Because pods are controlled by a controller, a new pod will be created immediately after you delete a pod. Manual pod deletion is useful when an upgrade fails halfway or when service processes need to be restarted.

Deleting a Deployment

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the target Deployment and click **Delete** in the **Operation** column.

----End

4.1.2 Pods

4.1.2.1 Overview

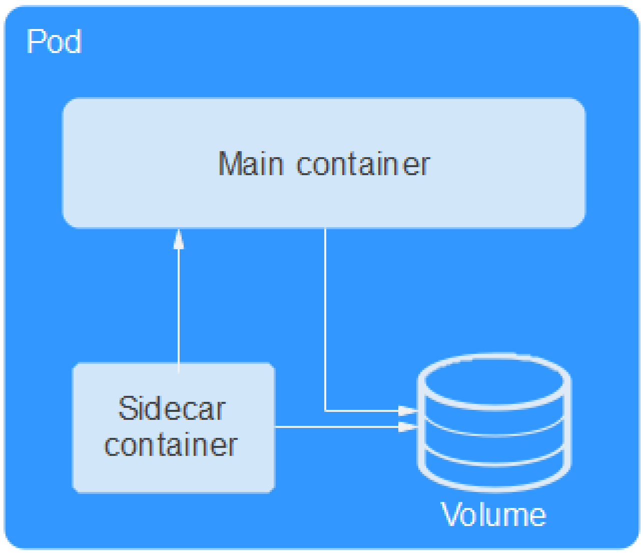
What Is a Pod?

Pods are the smallest deployable units of computing that you can create and manage. A pod is a group of one or more containers, with shared storage, a unique IP address, and a specification for how to run the containers.

Pods can be used in either of the following ways:

- A pod runs a single container. This is the most common use case. You can consider a pod as a wrapper around a single container. Pods can be managed directly rather than containers.
- A pod runs multiple containers that need to work together. In this scenario, a pod can encapsulate an application that is running in a main container and several sidecar containers. As shown in **Figure 4-2**, the main container serves as a web server that provides file services from a fixed directory, and the sidecar container periodically downloads files to the directory.

Figure 4-2 Pod



4.1.2.2 Creating a Pod

Scenario

You can use CCI 2.0 to quickly create pods for running workloads. On the CCI 2.0 console, you can view details about all pods, such as basic information, container list, storage volumes, and events. In addition, you can use remote terminals to access pods. You can also delete pods if you no longer need them.

Prerequisites

- A namespace has been created. If there are no namespaces, create one by referring to **Creating a Namespace**.
- The image has been uploaded. For details, see **Pushing an Image Through a Container Engine Client**.

Constraints

There are system components that help run the pods. These system components occupy some underlying resources, such as vCPU and memory. As a result, the resource usages for the pods may not reach the expected limits. To avoid this, refer to Reserved System Overhead.

Creating a Pod

Step 1 Log in to the **CCI 2.0 console**.

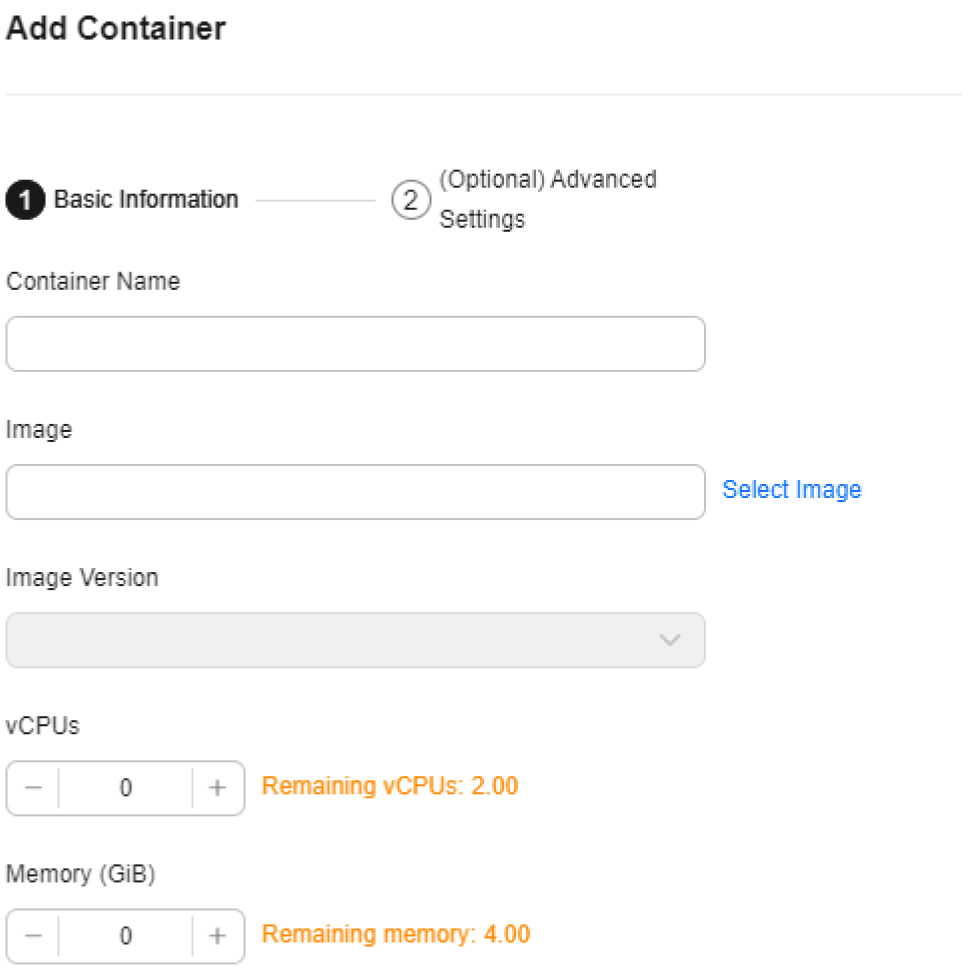
Step 2 In the navigation pane, choose **Workloads**. On the **Pods** tab, click **Create Pod**.

Step 3 On the **Create Pod** page, enter the basic information.

Parameter	Description
Pod Name	● Enter a name for the pod. The name must be unique in the same namespace. ● Enter 1 to 204 characters. Start and end with a lowercase letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. The total length of the pod name and namespace name cannot exceed 217 characters.
Namespace	Namespace that the pod belongs to.
Description (Optional)	Enter a description, which cannot exceed 250 characters.
CPU Architecture	x86 or Kunpeng
Pod Type	**General** or **General-computing-lite** **NOTE** Only x86-based general-computing-lite pods are supported.
vCPUs	Select a value from 0.25 to 64.
Memory	Select the memory based on the selected vCPUs.
Data Storage (Optional)	Only emptyDir volumes, ConfigMaps, and secrets are supported. Add a volume to the pod and then mount the volume to the specified container. Click **Add Data Store**, select a volume type, and enter a volume name. ● **emptyDir volume**: By default, CCI provides 30 GiB of free storage space, which is shared by emptyDir volumes and the system disk. ● **ConfigMap**: Select a ConfigMap. If no ConfigMaps are available, create one first. For details, see **Creating a ConfigMap**. ● **Secret**: Select a secret. If no secrets are available, create one first. For details, see **Creating a Secret**.

Step 4 Specify container settings.

1. Add basic container information. The total resources of a container cannot exceed the pod flavor.


Add Container

1 Basic Information — 2 (Optional) Advanced Settings

Container Name

Image

Select Image

Image Version

vCPUs

0

Remaining vCPUs: 2.00

Memory (GiB)

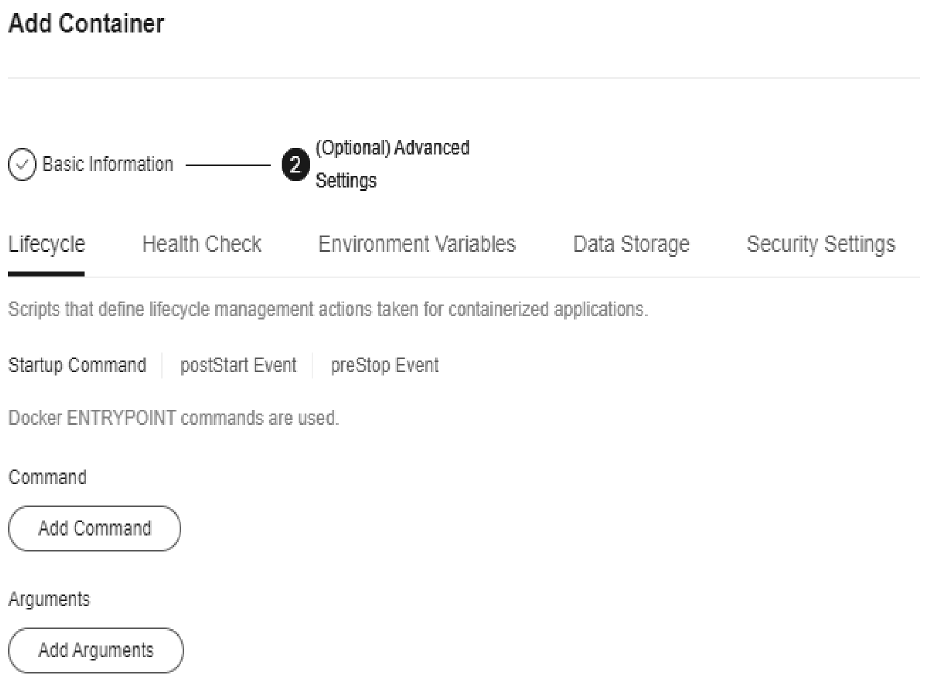
0

Remaining memory: 4.00

Table 4-2 Basic container information

Parameter	Description
Container Name	– The container name must be unique. – Enter 1 to 63 characters starting and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
Image	Select a container image. **CAUTION** Custom domain name images of the SWR Enterprise Edition cannot be used to create workloads.
Image Version	Select a container image tag.
vCPUs	Specify the vCPUs. The value cannot exceed that in the pod flavor.
Memory	Specify the memory. The value cannot exceed that in the pod flavor.

2. (Optional) Specify advanced container settings.


Add Container

Basic Information — 2 (Optional) Advanced Settings

Lifecycle | Health Check | Environment Variables | Data Storage | Security Settings

Scripts that define lifecycle management actions taken for containerized applications.

Startup Command | postStart Event | preStop Event

Docker ENTRYPOINT commands are used.

Command

Add Command

Arguments

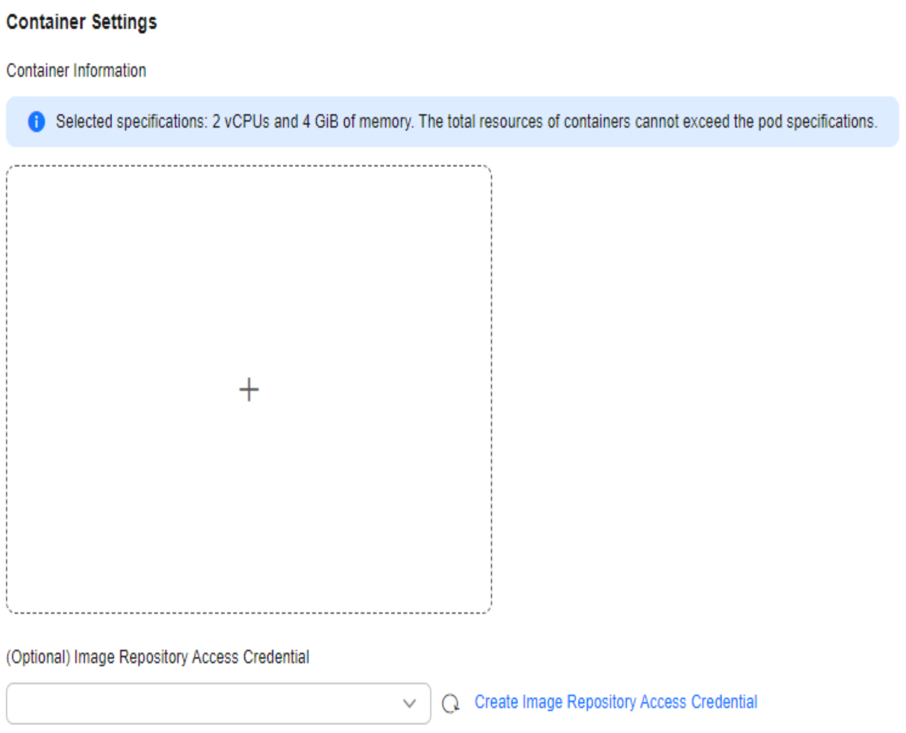
Add Arguments

Table 4-3 Advanced container settings

Parameter	Description
Lifecycle	CCI 2.0 provides containers with **lifecycle hooks**, which enable containers to run code triggered by events during their lifecycle. For example, if you want a container to perform a certain operation before it is stopped, you can register a hook. For details, see **Lifecycle**. CCI provides the following lifecycle hooks: – Startup command: Docker ENTRYPOINT commands are used. – PostStart: This hook is triggered after an application is started. – PreStop: This hook is triggered before an application is stopped.
Health Check	Container health can be checked regularly when the container is running. For details, see **Health Check**. CCI supports the following types of probes: – Liveness probe: checks whether a container is normal and a restart is required. – Readiness probe: checks whether a container is ready to respond to requests. – Startup probe: checks whether an application has already started.

Parameter	Description
Environment Variables	Environment variables affect the way a running container will behave. You can update them after deploying the workload.
Data Storage	Volumes can be mounted to containers to read data from files or store data files persistently. To mount a volume to a container, add the volume to the pod first.
Security Settings	Specify a user ID for all the containers to run with. For example, enter **0** to run as **root**.

Step 5 (Optional) Select an image repository access credential.


Container Settings

Container Information

Selected specifications: 2 vCPUs and 4 GiB of memory. The total resources of containers cannot exceed the pod specifications.

(Optional) Image Repository Access Credential

Step 6 Click **Create Now**.

----End

4.1.2.3 Pod Flavor

4.1.2.3.1 Upgrading Pod Flavors

For pods created on the CCI 2.0 console or using APIs, their flavors will be automatically upgraded based on the vCPUs and memory of containers to make full use of resources. This section describes how pod flavors are upgraded.

Table 4-4 Supported pod flavors

Container vCPUs	Container Memory (GiB)
0.25	0.5, 1, and 2
0.5	0.5, 1, 2, 3, and 4
1	1 to 8 (increment: 1 GiB)
2	2 to 16 (increment: 1 GiB)
4	4 to 32 (increment: 1 GiB)
8	8 to 64 (increment: 4 GiB)
16	16 to 128 (increment: 8 GiB)
32	32, 64, 128, and 256
48	96, 192, and 384
64	128, 256, and 512

There are two ways to specify pods flavors.

Method 1: Specifying the Memory and vCPUs for the Pod

- Use the pod annotation **resource.cci.io/pod-size-specs** to specify the pod flavor. The specified pod flavor must be the same as one listed in **Table 4-4**, or the API for creating a pod returns an error message and refuses to create the pod. The value of **resource.cci.io/pod-size-specs** can be in the format of *${CPU}_${MEM}*, for example, **2.00_4.0**.
- The total vCPUs and memory of containers in a pod cannot exceed the pod flavor specified by **resource.cci.io/pod-size-specs**, or the API for creating a pod returns an error message and refuses to create the pod.

Method 2: Specifying vCPUs and Memory of Containers

- If the pod flavor is not specified, the memory and vCPUs required by the pod are calculated based on the total vCPUs and memory of containers in the pod. If the calculated result matches a flavor listed in **Table 4-4**, this flavor is used for the pod. If the calculated result does not match any listed flavor, it will be automatically upgraded to the next available flavor. Assume that the calculated result is 6 vCPUs and 15-GiB memory. The pod flavor will be automatically upgraded to 8 vCPUs and 16-GiB memory.
- The pod flavor is calculated as follows:

  Pod flavor = max(max(sum(spec.Containers.Request), container.Limit), max(spec.initContainer))

  sum(spec.Containers.Request): total vCPU requests and memory requests of all containers

  max(sum(spec.Containers.Request), container.Limit): the larger values between the total requests and the limits of each container max(spec.initContainer): the larger values between vCPU requests and limits and the larger values between memory requests and limits

max(max(sum(spec.Containers.Request), container.Limit), max(spec.initContainer resource value)): the larger values between container resource specifications and init container resource specifications

- The rules for upgrading pod flavors as follows:

  The pod flavor is upgraded to the next available flavor, and the vCPUs and memory must not exceed those allowed in the pod flavor. If the pod flavor cannot be upgraded, the pod fails to be created.

  The upgraded pod flavor is included in the pod annotation: **resource.cci.io/size=***${cpuCeil}_${memoryCeil}*.

  If the number of vCPUs is greater than 32, the pod flavor will not be upgraded. If the vCPUs of containers in the pod do not match any pod flavor in **Table 4-4**, the pod fails to be created. To upgrade the pod flavor to one with more than 32 vCPUs, set **resource.cci.io/resizing-large-size-instance-greater-than-32-cores** to **true** in the pod annotation.

CAUTION

- If **resource.cci.io/pod-size-specs** is not set for the pod and requests and limits are not specified for all containers in the pod, an error message is returned, and the pod fails to be created.
- The vCPUs and memory of each container must meet the Kubernetes validation requirements. If both requests and limits are specified, both requests and limits must be greater than or equal to 0, and the requests must be greater than or equal to the limits.
- A maximum of 10 containers can be created for each pod. If you have special requirements, you can apply for removing this restriction.

Examples of Automatic Pod Flavor Upgrades

Table 4-5 Automatic upgrades

Scenario	Container Settings	Resource Requests or Limits	Upgraded Pod Flavor	Description

Single container, with only requests specified	
	```
	containers:
	  - resources:
	      requests:
	        cpu: '1.5'
	        memory: 2Gi
	```
		1.5 vCPUs and 2 GiB of memory	2 vCPUs and 4 GiB of memory	The pod flavors are upgraded based on the requests.

Scenario	Container Settings	Resource Requests or Limits	Upgraded Pod Flavor	Description
Single container, with both requests and limits specified	`containers: - resources: limits: cpu: '3.5' requests: cpu: '1.5' memory: 4.5Gi`	3.5 vCPUs and 4.5 GiB of memory	4 vCPUs and 5 GiB of memory	The requests and limits are compared, and the larger values are used.
Multiple containers, with only requests specified	`containers: - resources: requests: cpu: '1.5' memory: 2Gi - resources: requests: cpu: '1.5' memory: 2Gi`	3 vCPUs and 4 GiB of memory	4 vCPUs and 4 GiB of memory	The sum of requests is used.
Multiple containers, with only limits specified	`containers: - resources: limits: cpu: '1.5' memory: 2Gi - resources: limits: cpu: '1.5' memory: 2Gi`	1.5 vCPUs and 2 GiB of memory	2 vCPUs and 4 GiB of memory	The maximum limit is used.
Multiple containers, with both requests and limits specified	`containers: - resources: limits: cpu: '5' memory: 8Gi requests: cpu: '1.5' memory: 2Gi - resources: requests: cpu: '1.5' memory: 2Gi`	5 vCPUs and 8 GiB of memory	8 vCPUs and 8 GiB of memory	The requests and limits are compared, and the larger values are used.

Scenario	Container Settings	Resource Requests or Limits	Upgraded Pod Flavor	Description

Containers + init containers, with a large number of container resources	`initContainers:`
	`  - resources:`
	`      requests:`
	`        cpu: '1.5'`
	`        memory: 2Gi`
	`containers:`
	`  - resources:`
	`      requests:`
	`        cpu: '3'`
	`        memory: 4Gi`
		3 vCPUs and 4 GiB of memory	4 vCPUs and 4 GiB of memory	The requests and limits of containers are compared with those of init containers, and the larger values are used.

Containers + init containers, with a large number of init container resources	`initContainers:`
	`  - resources:`
	`      requests:`
	`        cpu: '5'`
	`        memory: 8Gi`
	`  - resources:`
	`      requests:`
	`        cpu: '6'`
	`        memory: 9Gi`
	`containers:`
	`  - resources:`
	`      requests:`
	`        cpu: '3'`
	`        memory: 4Gi`
		6 vCPUs and 9 GiB of memory	8 vCPUs and 12 GiB of memory	The requests and limits of containers are compared with those of init containers, and the larger values are used.

Resource requests, same as those in the flavor	`containers:`
	`  - resources:`
	`      requests:`
	`        cpu: '48'`
	`        memory: 96Gi`
		48 vCPUs and 96 GiB of memory	48 vCPUs and 96 GiB of memory	There is no need to upgrade to the flavor.

Upgrading to 32 vCPUs	`containers:`
	`  - resources:`
	`      requests:`
	`        cpu: '31'`
	`        memory: 32Gi`
		31 vCPUs and 32 GiB of memory	32 vCPUs and 32 GiB of memory	Fewer than 32 vCPUs after the upgrade

More than 32 vCPUs after the upgrade	`containers:`
	`  - resources:`
	`      requests:`
	`        cpu: '33'`
	`        memory: 96Gi`
		33 vCPUs and 96 GiB of memory	No matched flavor	If the vCPU request is more than 32 vCPUs, no upgrade is performed.

Scenario	Container Settings	Resource Requests or Limits	Upgraded Pod Flavor	Description
More than 32 vCPUs, and upgrading to a large flavor allowed	`annotations: "resource.cci.io/resizing-large-size-instance-greater-than-32-cores": "true" containers: - resources: requests: cpu: '33' memory: 96Gi`	33 vCPUs and 96 GiB of memory	48 vCPUs and 96 GiB of memory	Upgrading to a large flavor is allowed.

CAUTION

When a workload pod that has multiple containers is scheduled from CCE to CCI 2.0 and only limits are set for the containers, the pod creation complies with the default Kubernetes resource configuration rule as follows:

If only limits are specified, requests that have the same values as limits will be automatically specified.

Take a Deployment created on the CCE console as an example. The **spec.template** file of the Deployment contains two containers, and only limits are set for each container. After a pod is created, the requests of each container are specified and equal to the limits.

The pod flavor may be greater than expected. You can manually specify the requests for the containers to avoid this issue.

4.1.2.3.2 Reserved System Overhead

Some necessary system components required by the pods occupy system resources. There is a difference between the memory in the pod flavor and the allocatable pod memory. CCI calculates the pod memory that can be allocated as follows:

- Memory in the pod flavor ≤ 2 GiB

  Allocatable pod memory = **Memory in the pod flavor**
- Memory in the pod flavor > 2 GiB

  Allocatable pod memory = **Memory in the pod flavor** – **Memory reserved for CCI** – **Memory reserved for the OS**

CAUTION

Memory in the pod flavor is the value of **${memoryCeil}** displayed in the pod annotation in the format of *resource.cci.io/size=${cpuCeil}_${memoryCeil}*.

Rules for Reserving Pod Memory on CCI

The total reserved pod memory is equal to the sum of that reserved for the OS and that reserved for CCI to manage the pod.

The memory reserved for the OS includes basic memory and floating memory that varies with the memory in the pod flavor. The memory reserved for CCI to manage the pod is 250 MiB.

Table 4-6 Rules for reserving pod memory

Scenario	Reserved for	Basic/ Floating	Reservation	Used by
Memory in the pod flavor ≤ 2 GiB	None	\	\	\
Memory in the pod flavor > 2 GiB	OS	Basic	Fixed at 150 MiB	OS services
		Floating reservation (varying with the memory in the pod flavor)	20 MiB per GiB	OS kernel
	CCI	Basic	Fixed at 250 MiB	CCI components

Example

Table 4-7 Reservation example

Memory in the Pod Flavor	Available Pod Memory	Calculation	Remarks
0.5 GiB	0.5 GiB	/	Memory in the pod flavor ≤ 2 GiB
1 GiB	1 GiB	/	
2 GiB	2 GiB	/	
3 GiB	2.55 GiB	3 GiB – 250 MiB – (150 MiB + 3 × 20 MiB) = 2.55 GiB	Memory in the pod flavor > 2 GiB
4 GiB	3.53 GiB	4 GiB – 250 MiB – (150 MiB + 4 × 20 MiB) = 3.53 GiB	

Memory in the Pod Flavor	Available Pod Memory	Calculation	Remarks
8 GiB	7.45 GiB	8 GiB – 250 MiB – (150 MiB + 8 × 20 MiB) = 7.45 GiB	
16 GiB	15.29 GiB	16 GiB – 250 MiB – (150 MiB + 16 × 20 MiB) = 15.29 GiB	
32 GiB	30.98 GiB	32 GiB – 250 MiB – (150 MiB + 32 × 20 MiB) = 30.98 GiB	
64 GiB	62.35 GiB	64 GiB – 250 MiB – (150 MiB + 64 × 20 MiB) = 62.35 GiB	
128 GiB	125.1 GiB	128 GiB – 250 MiB – (150 MiB + 128 × 20 MiB) = 125.1 GiB	
256 GiB	250.6 GiB	256 GiB – 250 MiB – (150 MiB + 256 × 20 MiB) = 250.6 GiB	

4.1.2.3.3 Increasing Reserved System Overhead

For pods running on CCI 2.0, the OS and CCI occupy some underlying resources. In some extreme scenarios, the actual memory usage of a pod may not reach the memory in the pod flavor. To solve this issue, two annotations are provided to reserve the overhead for system components.

The **resource.cci.io/memory-reservation** annotation specifies whether to enable system overhead reservation. The default value is **false**, and the value **true** indicates that system overhead reservation is enabled. After this option is enabled, CCI 2.0 reserves 1 GiB of memory for the system components. If the memory exceeds that in the current pod flavor, the pod flavor is automatically rounded up to a flavor that is lightly larger than the current flavor. You will be billed for the rounded-up pod flavor.

The **resource.cci.io/memory-burst-size** annotation specifies whether to increase the container memory limits to the memory in the pod flavor. The default value is **false**, and the value **true** indicates that this option is enabled. After this option is enabled, the memory limits of all containers in the pod are adjusted to the memory in the pod flavor to increase the upper limit of the cgroup memory in the host environment. After system overhead reservation is enabled, the memory limited by cgroup can be released for application containers after the pod flavor is rounded up.

Table 4-8 Example

Scenario	Pod Configuration	Resource Requests	Rounded-up Pod Flavor	Final Pod Flavor on CCI	Description
The pod has only one container, and only resource requests are configured. System overhead reservation and increasing memory limits to the memory in the pod flavor are enabled.	annotations: "resource.cci.io/memory-reservation": "true" "resource.cci.io/memory-burst-size": "true" ... containers: - resources: requests: cpu: '1.5' memory: 2Gi	1.5 vCPUs and 3 GiB of memory	2 vCPUs and 4 GiB of memory	containers: - resources: limits: memory: 4Gi requests: cpu: '1.5' memory: 2Gi	How to calculate the required resources: 1. The pod flavor is rounded up based on the resource requests (1.5c2Gi). 2. System overhead reservation is enabled, and 1 GiB of memory (1.5c3Gi) is added. 3. The pod flavor is rounded up (2c4Gi) based on the rules described in **Upgrading Pod Flavors**. 4. The container memory limits are increased to the memory (4 GiB) in the pod flavor.

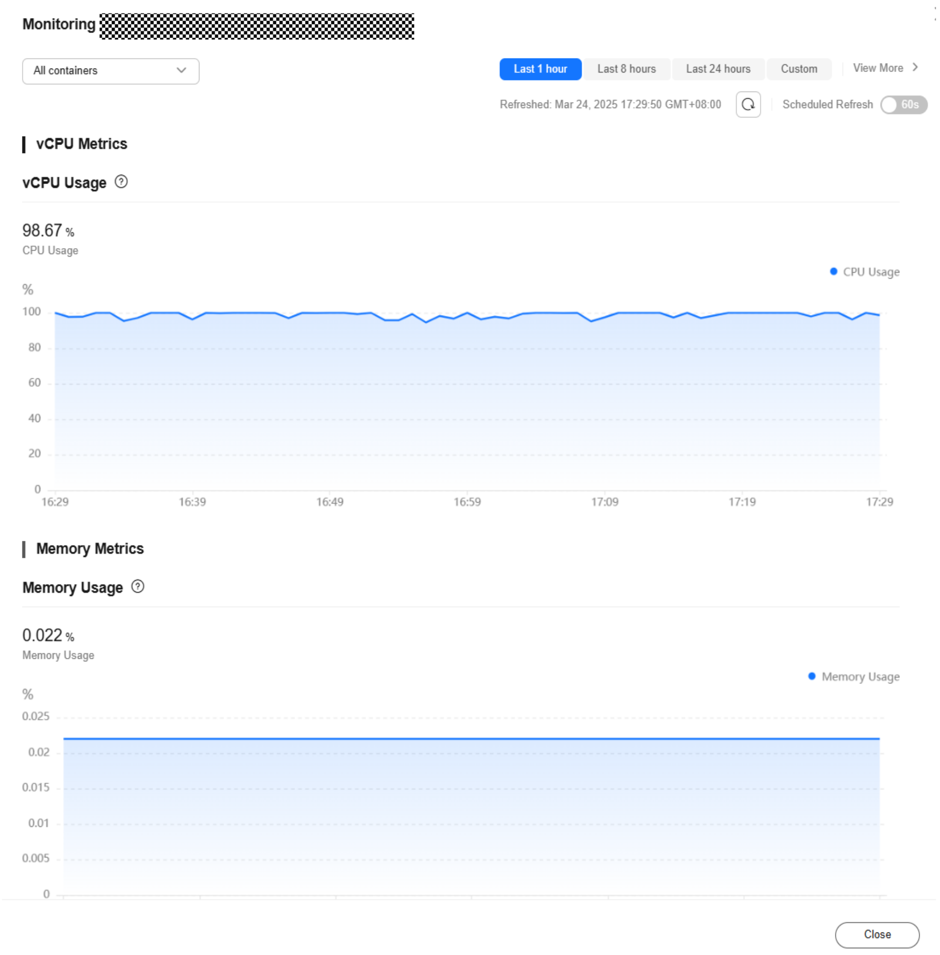
4.1.3 Viewing Resource Usages

If you enable AOM when creating a namespace, you can log in to the CCI 2.0 console to view the resource usages of each pod after a workload is created.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 Locate the Deployment or pod and click **Monitor** in the **Operation** column to view the resource usages.

Figure 4-3 Viewing monitoring data



Monitoring [redacted]

All containers

Refreshed: Mar 24, 2025 17:29:50 GMT+08:00

Scheduled Refresh 60s

vCPU Metrics

vCPU Usage

98.67 %
CPU Usage

Memory Metrics

Memory Usage

0.022 %
Memory Usage

----End

4.1.4 Lifecycle

4.1.4.1 Startup Command

Starting the container is to start the main process. However, some preparations must be made before the main process is started. For example, you configure or initialize MySQL databases before running MySQL servers. You can set **ENTRYPOINT** or **CMD** in the Dockerfile when you create an image. As shown in the following, the **ENTRYPOINT ["top", "-b"]** command is set in the Dockerfile. This command will be executed during container startup.

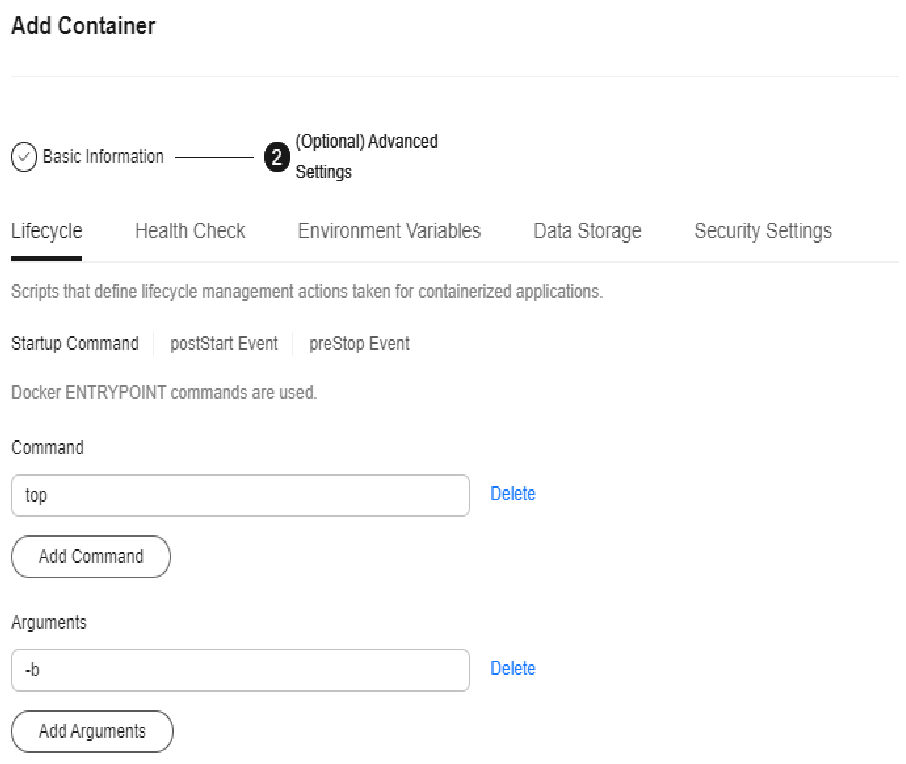
```
FROM ubuntu
ENTRYPOINT ["top", "-b"]
```

NOTICE

The startup command must be supported by the container image. Otherwise, the container startup will fail.

You can also set the container startup command. For example, to add the preceding command in the Dockerfile, you can click **Add** and enter the **top** command, and then click **Add** again and enter **-b** in the **Advanced Settings** area when you create a workload.

Figure 4-4 Startup command


Add Container

Basic Information — 2 (Optional) Advanced Settings

Lifecycle | Health Check | Environment Variables | Data Storage | Security Settings

Scripts that define lifecycle management actions taken for containerized applications.

Startup Command | postStart Event | preStop Event

Docker ENTRYPOINT commands are used.

Command

top — Delete

Add Command

Arguments

-b — Delete

Add Arguments

When the container engine runs, only one **ENTRYPOINT** command is supported. The startup command that you set in CCI 2.0 will overwrite the **ENTRYPOINT** and **CMD** commands that you set in the Dockerfile when you created the image. The following table lists the rules.

Image Entrypoint	Image CMD	Command for Running a Container	Parameter for Running 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]

4.1.4.2 PostStart/PreStop Processing

Setting Container Lifecycle Parameters

CCI 2.0 provides containers with **lifecycle hooks**, which enable containers to run code triggered by events during their lifecycle. For example, if you want a container to perform a certain operation before it is stopped, you can register a hook. The following lifecycle hooks are provided:

- PostStart: triggered immediately after the container is started
- PreStop: triggered immediately before the container is stopped

NOTE

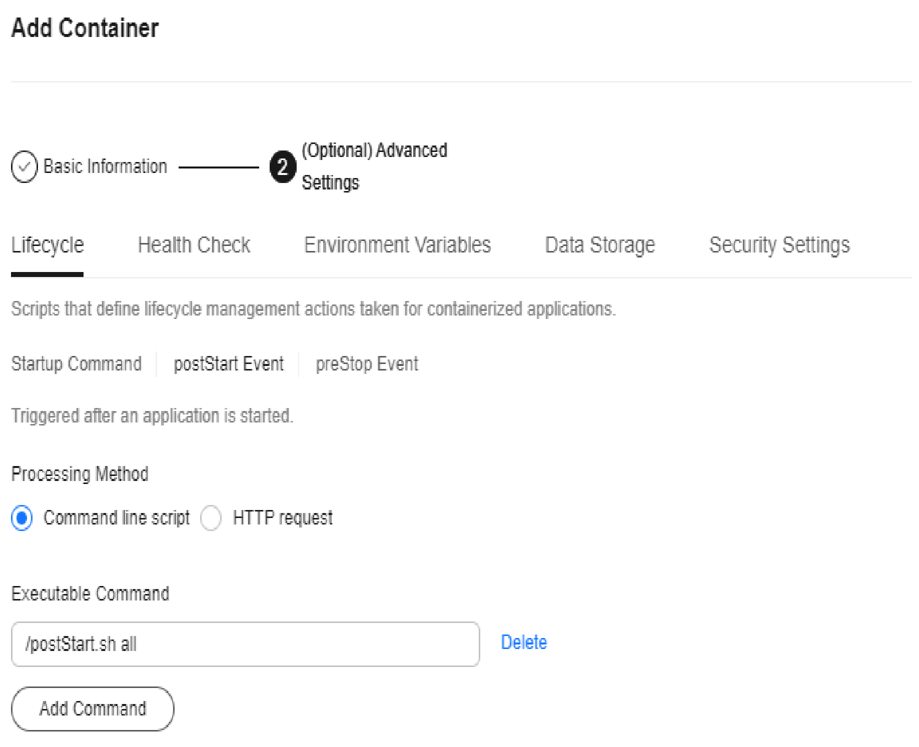
CCI 2.0 supports only hook handlers of the Exec type, which execute a specific command.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 On the **Lifecycle** tab, configure PostStart or PreStop.

For example, if you want to run the **/PostStart.sh all** command in the container, configure data on the page shown in the following figure. The first row indicates the script name and the second row indicates a parameter setting.

Figure 4-5 Command settings



Scripts that define lifecycle management actions taken for containerized applications.

Triggered after an application is started.

Processing Method

[✓ Command line script] HTTP request

Executable Command

/postStart.sh all

----End

4.1.5 Health Check

The health of a running container can be checked regularly.

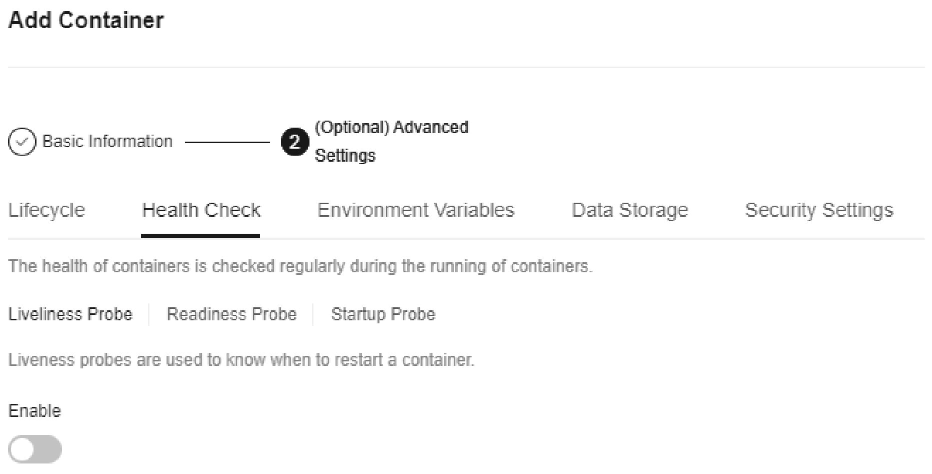
CCI 2.0 supports the following types of probes:

- Liveness probe: checks whether a containerized application is alive. The liveness probe is similar to the **ps** command for checking whether a process is running. If the application fails the check, the container will be restarted. If the application passes the check, no operation will be performed.
- Readiness probe: checks whether a containerized application is ready to handle requests. An application may take a long time to start up and provide services due to some reasons, for example, it needs to load disk data or wait for the startup of an external module. In this case, application processes are running, but the application is not ready to provide services. This is where the readiness probe is useful.
- Startup probe: checks whether a container is started successfully. It checks when the container is started to ensure that the application can be started and run normally.

Health Check Methods

Step 1 Log in to the **CCI 2.0 console**.

Step 2 On the **Health Check** tab, configure the liveness probe, readiness probe, or startup probe.



Add Container

Basic Information — 2 (Optional) Advanced Settings

Lifecycle | Health Check | Environment Variables | Data Storage | Security Settings

The health of containers is checked regularly during the running of containers.

Liveliness Probe | Readiness Probe | Startup Probe

Liveness probes are used to know when to restart a container.

Enable

- **HTTP request**

  The probe sends an HTTP GET request to the container. If the probe receives a 2xx or 3xx status code, the container is healthy.

- **TCP connection**

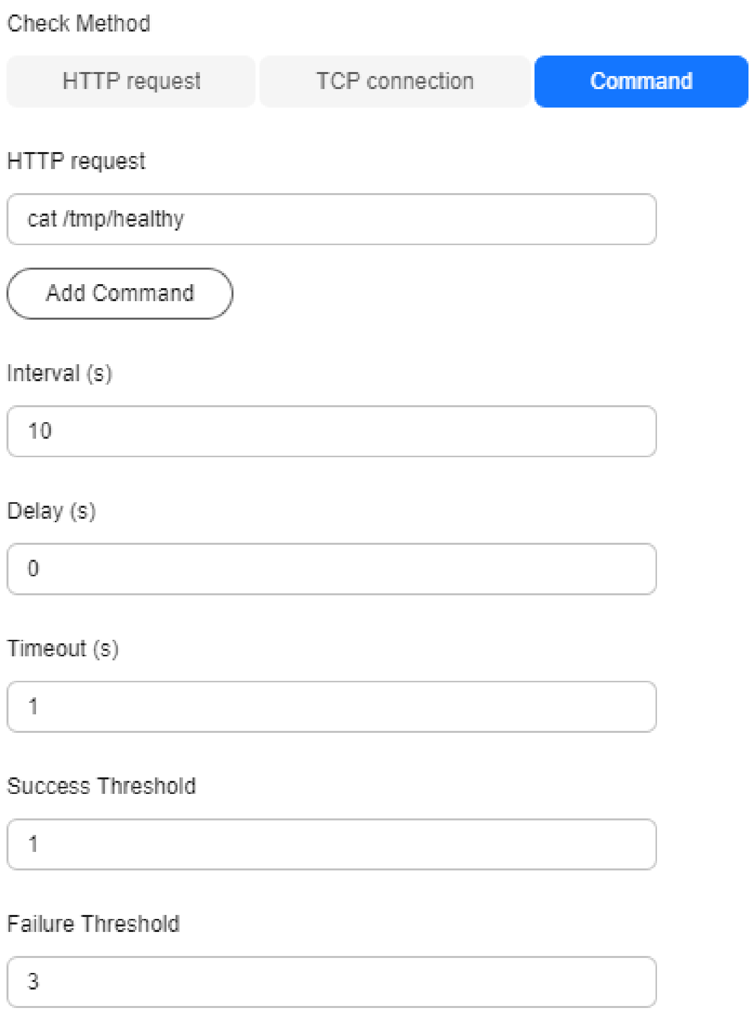
  The probe sends a TCP request to the container. If the probe receives a 2xx or 3xx status code, the container is healthy.

- **Command**

  The probe runs a command in the container and checks the exit status code. If the exit status code is 0, the container is healthy.

  For example, if you want to run the **cat /tmp/healthy** command to check whether the **/tmp/healthy** directory exists, configure data as shown in the following figure.

Figure 4-6 Command setting



Check Method

HTTP request: cat /tmp/healthy

Interval (s): 10

Delay (s): 0

Timeout (s): 1

Success Threshold: 1

Failure Threshold: 3

----End

Common Parameters

Table 4-9 Health check parameters

Parameter	Description
Interval (s)	Specifies the interval for performing a health check, in seconds. For example, if this parameter is set to **10**, the health check is performed every 10 seconds.
Delay (s)	Specifies the time required for the probe to be initiated after the container has started, in seconds. For example, if you set this parameter to **10**, the probe is initiated within 10 seconds after the container is started.

Parameter	Description
Timeout (s)	Specifies the amount of time after which the probe times out, in seconds. For example, if you set this parameter to **10**, the container must return a response within 10 seconds. Otherwise, the probe is counted as failed. If you set this parameter to **0** or do not specify any value, the default value (1 second) is used. **NOTE** If the executed command generates a subprocess in the image, **timeoutSeconds** may not take effect.
Success Threshold	Specifies the number of consecutive successful health checks for a container to be considered healthy. For example, if this parameter is set to **1**, the health check is successful if the probe succeeds for one time after a probe failure.
Failure Threshold	Specifies the number of consecutive failed health checks for a container to be considered unhealthy. For example, if this parameter is set to **3**, the health check fails after the probe fails for three consecutive times. In this case, the container is considered unhealthy and will be restarted.

4.1.6 Web Terminal

A web terminal allows you to connect to the containers for debugging.

Constraints

- By default, sh is used for login so containers must support sh.
- You can only log in to running containers using the terminal.
- To exit the terminal, you need to enter **exit**, or the sh process will not be terminated.

Connecting to the Container Using the Terminal

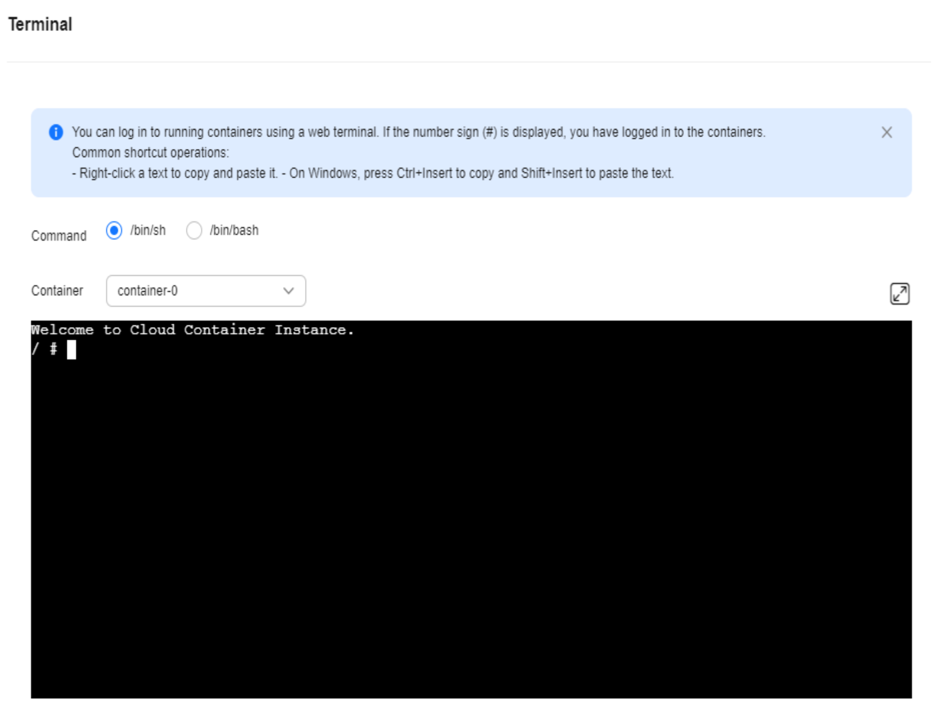
Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. Then click the **Pods** tab.

Step 3 Locate the target pod and click **View Terminal** in the **Operation** column.

If **#** is displayed, the login is successful.

Figure 4-7 Terminal



Terminal

You can log in to running containers using a web terminal. If the number sign (#) is displayed, you have logged in to the containers.
Common shortcut operations:
- Right-click a text to copy and paste it. - On Windows, press Ctrl+Insert to copy and Shift+Insert to paste the text.

Command (● /bin/sh) ○ /bin/bash

Container: container-0

```
Welcome to Cloud Container Instance.
/ #
```

----End

4.1.7 Workload Upgrade

You can upgrade a workload after you create it. There are two ways to upgrade a workload.

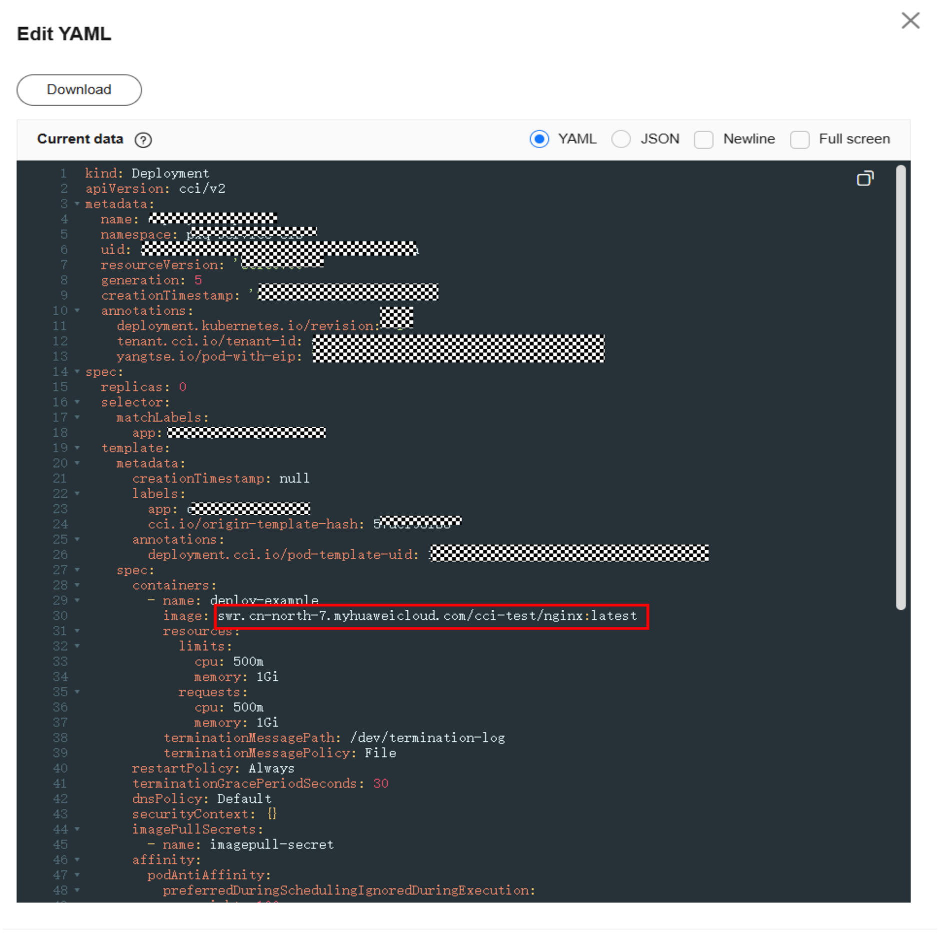
- **Rolling upgrade**: gradually replaces old pods with new pods. During the upgrade, service traffic is evenly distributed to the old and new pods to ensure service continuity.
- **In-place upgrade**: deletes an old pod and then creates a new one. Services will be interrupted during the upgrade.

Upgrading a Workload

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the target workload and click **Edit YAML** in the **Operation** column.

Step 3 Modify the fields in the YAML file to upgrade the workload. For example, to update an image, modify the fields highlighted in the following figure:



```yaml
kind: Deployment
apiVersion: cci/v2
metadata:
  name: [redacted]
  namespace: [redacted]
  uid: [redacted]
  resourceVersion: '[redacted]
  generation: 5
  creationTimestamp: '[redacted]
  annotations:
    deployment.kubernetes.io/revision:[redacted]
    tenant.cci.io/tenant-id: [redacted]
    yangtse.io/pod-with-eip: [redacted]
spec:
  replicas: 0
  selector:
    matchLabels:
      app: [redacted]
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: [redacted]
        cci.io/origin-template-hash: 5[redacted]
      annotations:
        deployment.cci.io/pod-template-uid: [redacted]
    spec:
      containers:
        - name: deploy-example
          image: swr.cn-north-7.myhuaweicloud.com/cci-test/nginx:latest
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: Default
      securityContext: {}
      imagePullSecrets:
        - name: imagepull-secret
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
```

Step 4 Click **OK**.

----End

4.1.8 Workload Scaling

There are two workload scaling methods: auto scaling and manual scaling.

- Auto scaling: triggered by metrics. After the configuration is complete, pods can be automatically added or deleted as the resources change over time.
- Manual scaling: After the configuration is complete, pods can be added or deleted immediately.

Constraints

- Auto scaling is only supported for Deployments.
- CCI 2.0 supports auto scaling only in the TR-Istanbul, AF-Johannesburg, AP-Singapore, and ME-Riyadh regions.

Auto Scaling

NOTE

- If **spec.metrics.resource.target.type** is set to **Utilization**, you need to specify the resource requests value when creating a workload.
- When **spec.metrics.resource.target.type** is set to **Utilization**, the resource usage is calculated as follows: Resource usage = Used resource/Available pod flavor. You can determine the actual flavor of a pod by referring to **Pod Flavor**.

A properly configured auto scaling policy includes the metric, threshold, and step. It eliminates the need to manually adjust resources in response to service changes and traffic bursts, thus helping you reduce workforce and resource consumption.

CCI 2.0 supports **auto scaling based on metrics**. The workloads are scaled out or in based on the vCPU or memory usage. You can specify a vCPU or memory usage threshold. If the usage is higher or lower than the threshold, pods are automatically added or deleted.

- Configure a metric-based auto scaling policy.
  a. Log in to the **CCI 2.0 console**.
  b. In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the target Deployment and click its name.
  c. On the **Auto Scaling** tab, click **Create from YAML** to configure an auto scaling policy.

     The following describes the auto scaling policy file format:

     - Resource description in the **hpa.yaml** file

```
kind: HorizontalPodAutoscaler
apiVersion: cci/v2
metadata:
  name: nginx             # Name of the HorizontalPodAutoscaler
  namespace: test         # Namespace of the HorizontalPodAutoscaler
spec:
  scaleTargetRef:         # Reference the resource to be automatically scaled.
    kind: Deployment      # Type of the target resource, for example, Deployment
    name: nginx           # Name of the target resource
    apiVersion: cci/v2    # Version of the target resource
  minReplicas: 1          # Minimum number of replicas for HPA scaling
  maxReplicas: 5          # Maximum number of replicas for HPA scaling
  metrics:
    - type: Resource                  # Resource metrics are used.
      resource:
        name: memory                  # Resource name, for example, cpu or memory
        target:
          type: Utilization         # Metric type. The value can be Utilization (percentage) or
AverageValue (absolute value).
          averageUtilization: 50      # Resource usage. For example, when the CPU usage
reaches 50%, scale-out is triggered.
  behavior:
    scaleUp:
      stabilizationWindowSeconds: 30  # Scale-out stabilization duration, in seconds
      policies:
      - type: Pods          # Number of pods to be scaled
        value: 1
        periodSeconds: 30  # The check is performed once every 30 seconds.
    scaleDown:
      stabilizationWindowSeconds: 30  # Scale-in stabilization duration, in seconds
      policies:
      - type: Percent      # The resource is scaleed in or out based on the percentage of
existing pods.
        value: 50
        periodSeconds: 30  # The check is performed once every 30 seconds.
```

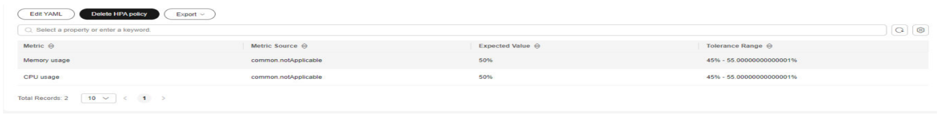
- Resource description in the **hpa.json** file

```
{
    "kind": "HorizontalPodAutoscaler",
    "apiVersion": "cci/v2",
    "metadata": {
        "name": "nginx",             # Name of the the HorizontalPodAutoscaler
                "namespace": "test"          # Namespace of the HorizontalPodAutoscaler
    },
    "spec": {
                "scaleTargetRef": {         # Reference the resource to be automatically scaled.
            "kind": "Deployment",        # Type of the target resource, for example, Deployment
            "name": "nginx",             # Name of the target resource
            "apiVersion": "cci/v2"       # Version of the target resource
        },
        "minReplicas": 1,             # Minimum number of replicas for HPA scaling
        "maxReplicas": 5,             # Maximum number of replicas for HPA scaling
        "metrics": [
            {
                "type": "Resource",             # Resource metrics are used.
                "resource": {
                    "name": "memory",       # Resource name, for example, cpu or memory
                    "target": {
                        "type": "Utilization",           # Metric type. The value can be Utilization
(percentage) or AverageValue (absolute value).
                        "averageUtilization": 50         # Resource usage. For example, when the
CPU usage reaches 50%, scale-out is triggered.
                    }
                }
            }
        ],
        "behavior": {
            "scaleUp": {
                "stabilizationWindowSeconds": 30,
                "policies": [
                    {
                        "type": "Pods",
                        "value": 1,
                        "periodSeconds": 30
                    }
                ]
            },
            "scaleDown": {
                "stabilizationWindowSeconds": 30,
                "policies": [
                    {
                        "type": "Percent",
                        "value": 50,
                        "periodSeconds": 30
                    }
                ]
            }
        }
    }
}
```

d. Click **OK**.

   You can view the auto scaling policy on the **Auto Scaling** tab.

   Figure 4-8 Auto scaling policy



Metric	Metric Source	Expected Value	Tolerance Range
Memory usage	common.notApplicable	50%	45% - 55.00000000000001%
CPU usage	common.notApplicable	50%	45% - 55.00000000000001%

Total Records: 2 10 < 1 >

   When the trigger condition is met, the auto scaling policy will be executed.

Manual Scaling

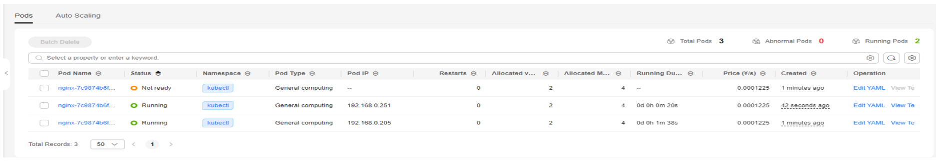
Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the target Deployment and click **Edit YAML**.

Step 3 Change the value of **spec.replicas**, for example, to **3**, and click **OK**.

Step 4 On the **Pods** tab, view the new pods being created. When the statuses of all added pods change to **Running**, the scaling is complete.

Figure 4-9 Pod list after a manual scaling



Pod Name	Status	Namespace	Pod Type	Pod IP	Restarts	Allocated v...	Allocated M...	Running Du...	Price (¥/s)	Created	Operation
nginx-7c9874b6f...	Not ready	kubectl	General computing	--	0	2	4	--	0.0001225	1 minutes ago	Edit YAML View Te
nginx-7c9874b6f...	Running	kubectl	General computing	192.168.0.251	0	2	4	0d 0h 0m 20s	0.0001225	42 seconds ago	Edit YAML View Te
nginx-7c9874b6f...	Running	kubectl	General computing	192.168.0.205	0	2	4	0d 0h 1m 38s	0.0001225	1 minutes ago	Edit YAML View Te

Total Records: 3 50 < 1 >

----End

4.2 Network Management

4.2.1 Services

4.2.1.1 Service Overview

A Service is a resource used to define a pod network access interface. It allows a group of pods to be accessed in a stable manner so you do not need to consider the location and quantity of the pods. CCI 2.0 supports Services of the LoadBalancer type.

Workloads can be accessed over a private or public network, and they can also access the public network.

- **Private network access**: Create a Service of the LoadBalancer type, configure a private network load balancer, and use the private IP address of the private network load balancer to access the workload. You can also configure a trustlist and blocklist for access control. This method can be used in the following scenarios: mutual access between workloads in the same namespace, mutual access between other cloud resources (such as ECSs) and CCI 2.0 workloads in the same VPC, and mutual access between workloads in different namespaces of the same VPC. Services are provided over the private network through the private IP address and port of the load balancer in the format of *<private-IP-address>:<port>*.
- **Public network access**: Create a Service of the LoadBalancer type and configure a public network load balancer. You can use the public IP address and port of the load balancer to access the workload from the public network. You can also configure a trustlist or blocklist for access control.

Constraints on Services

- For Services of the LoadBalancer type, only dedicated load balancers are available. If a Service of the LoadBalancer type is used, the pods can have IPv4 IP addresses.
- If the CCE Cloud Bursting Engine for CCI add-on is used to schedule the workloads to CCI 2.0, dedicated load balancers can be configured for ingresses and Services of the LoadBalancer type. The CCE Cloud Bursting Engine for CCI add-on does not support Services of the LoadBalancer type if its version is earlier than 1.5.5.

4.2.1.2 Private Network Access

Overview

If you create a Service of the LoadBalancer type and configure a private network load balancer for the Service, you can use the private IP address and port of the load balancer to access the workload. This method can be used in the following scenarios: mutual access between workloads in the same namespace, mutual access between other cloud resources (such as ECSs) and CCI 2.0 workloads in the same VPC, and mutual access between workloads in different namespaces of the same VPC. Services are provided over the private network through the private IP address and port of the load balancer in the format of *<private-IP-address>:<port>*.

Workloads run in pods. Accessing a workload is to access the pods for that workload.

Constraints

- The load balancer must be in the same VPC as the workload.
- Only dedicated load balancers are supported.

Creating a Service for an Existing Workload

You can create a Service for a workload after it is created. Creating a Service has no impact on the workload. Once created, the Service can be used by the workload for network access immediately.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the right of the page, click **Create from YAML**.

Step 3 Import or add a YAML file.

The following is an example YAML file:

- Resource description in the **service.yaml** file

```
apiVersion: cci/v2
kind: Service
metadata:
  name: kubectl-test
  namespace: kubectl
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: 1234567890 # Load balancer ID. Only dedicated load balancers are supported.
```

```
spec:
  selector:
    app: kubectl-test # Label of the associated workload
  ports:
    - name: service-0
      targetPort: 80   # Container port
      port: 12222      # Access port (load balancer's port for accessing the workload)
      protocol: TCP     # Protocol used to access the workload
  type: LoadBalancer
```

- Resource description in the **service.json** file

```
{
    "apiVersion": "cci/v2",
    "kind": "Service",
    "metadata": {
        "name": "kubectl-test",
        "namespace": "kubectl",
        "annotations": {
                        "kubernetes.io/elb.class": "elb",
            "kubernetes.io/elb.id": "1234567890"  # Load balancer ID. Only dedicated load balancers are
supported.
        }
    },
    "spec": {
        "selector": {
            "app": "kubectl-test" # Label of the associated workload
        },
        "ports": [
            {
                "name": "service-0",
                "targetPort": 80,     # Container port
                "port": 12222,        # Access port (load balancer's port for accessing the workload)
                "protocol": "TCP",    # Protocol used to access the workload
                "type": "LoadBalancer"
            }
        ]
    }
}
```

Step 4 Click **OK**. Access the workload through the load balancer's private IP address and port in the format of *<private-IP-address>:<port>*.

----End

Updating a Service

After you add a Service, you can update the access port of the Service.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the **Services** page, select the target namespace, locate the Service and click **Edit YAML** in the **Operation** column.

Step 3 Only the access port can be modified.

spec.ports[i].port indicates the access port. The port number ranges from **1** to **65535**.

Step 4 Click **OK**. The Service will be updated for the workload.

----End

4.2.1.3 Public Network Access

Overview

Workloads can be accessed from the public network. For this to work, you need to create a Service of the LoadBalancer type and create a public network load balancer in the same VPC as the workload.

Constraints

- The load balancer must be in the same VPC as the workload.
- You must familiarize yourself with the constraints on EIPs. For details, see **EIP Notes and Constraints**.
- Only dedicated load balancers are supported, and each load balancer must have an EIP bound.

Creating a Service for an Existing Workload

You can create a Service for a workload after it is created. Creating a Service has no impact on the workload. Once created, the Service can be used by the workload for network access immediately.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the right of the page, click **Create from YAML**.

Step 3 Import or add a YAML file.

The following is an example YAML file.

- Resource description in the **service.yaml** file

```
apiVersion: cci/v2
kind: Service
metadata:
  name: kubectl-test
  namespace: kubectl
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: 1234567890 # Load balancer ID. Only dedicated load balancers are supported.
spec:
  selector:
    app: kubectl-test # Label of the associated workload
  ports:
    - name: service-0
      targetPort: 80   # Container port
      port: 12222      # Access port (load balancer's port for accessing the workload)
      protocol: TCP     # Protocol used to access the workload
  type: LoadBalancer
```

- Resource description in the **service.json** file

```
{
    "apiVersion": "cci/v2",
    "kind": "Service",
    "metadata": {
        "name": "kubectl-test",
        "namespace": "kubectl",
        "annotations": {
                        "kubernetes.io/elb.class": "elb",
            "kubernetes.io/elb.id": "1234567890"  # Load balancer ID. Only dedicated load balancers are
supported.
        }
    },
    "spec": {
        "selector": {
            "app": "kubectl-test" # Label of the associated workload
        },
        "ports": [
            {
                "name": "service-0",
                "targetPort": 80,     # Container port
                "port": 12222,        # Access port (load balancer's port for accessing the workload)
                "protocol": "TCP",    # Protocol used to access the workload
                "type": "LoadBalancer"
            }
        ]
    }
}
```

Step 4 Click **OK**. Access the workload through the load balancer's EIP and port in the format of *<EIP-of-the-load-balancer>:<port>*.

----End

What If a Workload Cannot Be Accessed from the Public Network?

- A workload can only be accessed from the public network when it is in the running state. If your workload is abnormal or not ready, it cannot be accessed from the public network.
- It may take one to three minutes from the time when the workload was created to the time for it to be ready for access from the public network. During this time period, the network routes have not yet been configured. As a result, the workload cannot be accessed from the public network.
- If a workload is inaccessible 3 minutes after it is created, and there is no alarm event, a possible cause is that the container port is not being listened to. You need to use the image to check whether the container port is being listened to. If the container port is being listened to, the access failure may be caused by the load balancer. In this case, you need to check the load balancer.

Updating a Service

After you add a Service, you can update the access port of the Service.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the **Services** page, select the target namespace, locate the Service and click **Edit YAML** in the **Operation** column.

Step 3 Only the access port can be modified.

spec.ports[i].port: indicates the access port. The port number ranges from **1** to **65535**.

Step 4 Click **OK**. The Service will be updated for the workload.

----End

4.2.1.4 Configuring HTTP/HTTPS for a LoadBalancer Service

4.2.1.4.1 Configuring HTTP for a LoadBalancer Service

Configuring an HTTP Service

You can create a Service for a workload after it is created. Creating a Service has no impact on the workload. Once created, the Service can be used by the workload for network access immediately.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the right of the page, click **Create from YAML**.

Step 3 Import or add a YAML file.

The following is an example YAML file.

- Resource description in the **service.yaml** file:

```
apiVersion: cci/v2
kind: Service
metadata:
  name: kubectl-test
  namespace: kubectl
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: 1234567890 # Load balancer ID. Only dedicated load balancers are supported.
    kubernetes.io/elb.protocol-port: "http:80"  # HTTP and port number, which must be the same as
the port number in spec.ports
spec:
  selector:
    app: kubectl-test # Label of the associated workload
  ports:
    - name: service-0
      targetPort: 80   # Container port
      port: 80        # Access port (load balancer's port for accessing the workload)
      protocol: TCP    # Set the protocol to TCP.
  type: LoadBalancer
```

- Resource description in the **service.json** file:

```
{
    "apiVersion": "cci/v2",
    "kind": "Service",
    "metadata": {
        "name": "kubectl-test",
        "namespace": "kubectl",
        "annotations": {
                        "kubernetes.io/elb.class": "elb",
            "kubernetes.io/elb.id": "1234567890"  # Load balancer ID. Only dedicated load balancers are
supported.
                        "kubernetes.io/elb.protocol-port": "http:80"  # HTTP and port number, which must be
the same as the port number in spec.ports.
        }
    },
    "spec": {
        "selector": {
            "app": "kubectl-test" # Label of the associated workload
        },
        "ports": [
            {
                "name": "service-0",
                "targetPort": 80,     # Container port
                "port": 80,           # Access port of the Service
                "protocol": "TCP",    # Set the protocol to TCP.
                "type": "LoadBalancer"
            }
        ]
```

```
  }
}
```

Step 4 Click **OK**. Access the workload through the load balancer's IP address and port in the format of *<IP-address>:<port>*.

----End

Updating a Service

After you add a Service, you can update the access port of the Service.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the **Services** page, select the target namespace, locate the Service, and click **Edit YAML** in the **Operation** column.

Step 3 Only the access port can be modified.

spec.ports[i].port: indicates the access port. The port number ranges from **1** to **65535**. You need to change the value of **kubernetes.io/elb.protocol-port** accordingly.

Step 4 Click **OK**. The Service will be updated for the workload.

----End

Parameters for an HTTP Service

You can refer to the following table to add the annotations to configure a listener for an HTTP Service.

Parameter	Description
kubernetes.io/elb.http2-enable	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Value options: ● **true**: enabled ● **false**: disabled (default value) **CAUTION** HTTP/2 can be enabled or disabled only for HTTPS listeners. This option is invalid when the listeners use HTTP. The default value is **false**.
kubernetes.io/elb.tls-certificate-ids	IDs of SNI certificates used in ELB, separated by commas (,). Each SNI certificate must contain domain names. To obtain the value, log in to the ELB console and choose **Elastic Load Balance** > **Certificates**.

Parameter	Description
kubernetes.io/elb.security-pool-protocol	If the listener uses HTTPS, you can set the backend protocol to HTTPS. **The backend protocol of an existing listener cannot be changed. If you want to change the backend protocol, you need to add a new listener to the load balancer.** ● **true**: enabled ● **false**: disabled
kubernetes.io/elb.tls-ciphers-policy	Security policy used by a listener. Value options include **tls-1-0-inherit**, **tls-1-0**, **tls-1-1**, **tls-1-2**, **tls-1-2-strict**, **tls-1-2-fs**, **tls-1-0-with-1-3**, **tls-1-2-fs-with-1-3**, and **hybrid-policy-1-0**. The default value is **tls-1-0**. This option is available for HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.x-forwarded-port	If this option is enabled, the listening port of the load balancer can be transferred to backend servers through the HTTP header of the packet. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.x-forwarded-for-port	If this option is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.x-forwarded-host	If this option is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the request and transferred to backend servers. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.

Parameter	Description
kubernetes.io/elb.gzip-enabled	Data compression. ● **true**: Data compression is enabled, and specific file types will be compressed. ● **false**: Data compression is disabled, and no files will be compressed. By default, data compression is disabled. The files in the following format can be compressed: ● Brotli can compress all file formats. ● GZIP can compress the files of the following types: text/xml, text/plain, text/css, application/javascript, application/x-javascript, application/rss+xml, application/atom+xml, application/xml, and application/json. This option is available for HTTP/HTTPS listeners of dedicated load balancers.

4.2.1.4.2 Configuring HTTPS for a LoadBalancer Service

Configuring an HTTPS Service

You can create a Service for a workload after it is created. Creating a Service has no impact on the workload. Once created, the Service can be used by the workload for network access immediately.

Step 1 Log in to the ELB console and click **Create Certificate**.

Step 2 Log in to the **CCI 2.0 console**.

Step 3 In the navigation pane, choose **Services**. On the right of the page, click **Create from YAML**.

Step 4 Import or add a YAML file.

The following is an example YAML file.

- Resource description in the **service.yaml** file:

```
apiVersion: cci/v2
kind: Service
metadata:
  name: kubectl-test
  namespace: kubectl
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: 1234567890 # Load balancer ID. Only dedicated load balancers are supported.
    kubernetes.io/elb.protocol-port: "http:443"  # HTTP and port number, which must be the same as
that in spec.ports.
    kubernetes.io/elb.cert-id: "17e3b4f4bc40471c86741dc3aa211379"  # Certificate ID of the
LoadBalancer Service
spec:
  selector:
    app: kubectl-test # Label of the associated workload
  ports:
```

```
  - name: service-0
    targetPort: 80   # Container port
    port: 443      # Access port (load balancer's port for accessing the workload)
    protocol: TCP    # Set the protocol to TCP.
 type: LoadBalancer
```

- Resource description in the **service.json** file:

```
{
    "apiVersion": "cci/v2",
    "kind": "Service",
    "metadata": {
        "name": "kubectl-test",
        "namespace": "kubectl",
        "annotations": {
                        "kubernetes.io/elb.class": "elb",
            "kubernetes.io/elb.id": "1234567890"  # Load balancer ID. Only dedicated load balancers are
supported.
                        "kubernetes.io/elb.protocol-port": "https:443"  # HTTPS and port number, which must
be the same as that in spec.ports.
                        "kubernetes.io/elb.cert-id": "17e3b4f4bc40471c86741dc3aa211379" # Certificate ID
of the LoadBalancer Service
        }
    },
    "spec": {
        "selector": {
            "app": "kubectl-test" # Label of the associated workload
        },
        "ports": [
            {
                "name": "service-0",
                "targetPort": 80,     # Container port
                "port": 443,          # Access port of the Service
                "protocol": "TCP",    # Set the protocol to TCP.
                "type": "LoadBalancer"
            }
        ]
    }
}
```

Step 5 Click **OK**. Access the workload through the load balancer's IP address and port in the format of *<IP-address>:<port>*.

----End

Updating a Service

After you add a Service, you can update the access port of the Service.

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the **Services** page, select the target namespace, locate the Service, and click **Edit YAML** in the **Operation** column.

Step 3 Only the access port can be modified.

spec.ports[i].port: indicates the access port. The port number ranges from **1** to **65535**. You need to change the value of **kubernetes.io/elb.protocol-port** accordingly.

Step 4 Click **OK**. The Service will be updated for the workload.

----End

Parameters for an HTTPS Service

You can refer to the following table to add the annotations to configure a listener for an HTTPS Service.

Parameter	Description
kubernetes.io/elb.http2-enable	Whether HTTP/2 is enabled. Request forwarding using HTTP/2 improves the access performance between your application and the load balancer. However, the load balancer still uses HTTP/1.x to forward requests to the backend server. Value options: ● **true**: enabled ● **false**: disabled (default value) **CAUTION** HTTP/2 can be enabled or disabled only for HTTPS listeners. This option is invalid when the listeners use HTTP. The default value is **false**.
kubernetes.io/elb.tls-certificate-ids	IDs of SNI certificates used in ELB, separated by commas (,). Each SNI certificate must contain domain names. To obtain the value, log in to the ELB console and choose **Elastic Load Balance** > **Certificates**.
kubernetes.io/elb.security-pool-protocol	If the listener uses HTTPS, you can set the backend protocol to HTTPS. **The backend protocol of an existing listener cannot be changed. If you want to change the backend protocol, you need to add a new listener to the load balancer.** ● **true**: enabled ● **false**: disabled
kubernetes.io/elb.tls-ciphers-policy	Security policy used by a listener. Value options include **tls-1-0-inherit**, **tls-1-0**, **tls-1-1**, **tls-1-2**, **tls-1-2-strict**, **tls-1-2-fs**, **tls-1-0-with-1-3**, **tls-1-2-fs-with-1-3**, and **hybrid-policy-1-0**. The default value is **tls-1-0**. This option is available for HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.x-forwarded-port	If this option is enabled, the listening port of the load balancer can be transferred to backend servers through the HTTP header of the packet. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.

Parameter	Description
kubernetes.io/elb.x-forwarded-for-port	If this option is enabled, the source port of the client can be transferred to backend servers through the HTTP header of the packet. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.x-forwarded-host	If this option is enabled, **X-Forwarded-Host** will be rewritten using the **Host** field in the request and transferred to backend servers. ● **true**: enabled ● **false**: disabled This option is available for HTTP/HTTPS listeners of dedicated load balancers.
kubernetes.io/elb.gzip-enabled	Data compression. ● **true**: Data compression is enabled, and specific file types will be compressed. ● **false**: Data compression is disabled, and no files will be compressed. By default, data compression is disabled. The files in the following format can be compressed: ● Brotli can compress all file formats. ● GZIP can compress the files of the following types: text/xml, text/plain, text/css, application/javascript, application/x-javascript, application/rss+xml, application/atom+xml, application/xml, and application/json. This option is available for HTTP/HTTPS listeners of dedicated load balancers.

4.2.1.5 Configuring an Access Policy for a Service

Overview

You can add IP addresses to a trustlist or blocklist to control access to a load balancer associated with the Service.

- Trustlist: Only the IP addresses in the list can access the load balancer.
- Blocklist: IP addresses in the list are not allowed to access the load balancer.

Prerequisites

IP address groups have been created on the ELB console. For details, see **Creating an IP Address Group**.

Configuring an Access policy

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Services**. On the right of the page, click **Create from YAML**.

Step 3 Import or add the YAML file of the Service. For details about the parameters, see **Table 4-10**.

The following is an example YAML file:

- Resource description in the **service.yaml** file

```
apiVersion: cci/v2
kind: Service
metadata:
  name: kubectl-test
  namespace: kubectl
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: 1234567890 # Load balancer ID. Only dedicated load balancers are supported.
    kubernetes.io/elb.acl-id: <your_acl_id>              # ID of an IP address group for accessing the load
balancer
    kubernetes.io/elb.acl-type: 'white'                  # Trustlist for access control
spec:
  selector:
    app: kubectl-test
  ports:
    - name: service-0
      targetPort: 80   # Container port
      port: 12222      # Access port (load balancer's port for accessing the workload)
      protocol: TCP    # Protocol used to access the workload
  type: LoadBalancer
```

- Resource description in the **service.json** file

```
{
    "apiVersion": "cci/v2",
    "kind": "Service",
    "metadata": {
        "name": "kubectl-test",
        "namespace": "kubectl",
        "annotations": {
                        "kubernetes.io/elb.class": "elb"
            "kubernetes.io/elb.id": "1234567890"  # Load balancer ID. Only dedicated load balancers are
supported.
                        kubernetes.io/elb.acl-id: <your_acl_id>              # ID of an IP address group for
accessing the load balancer
                        "kubernetes.io/elb.acl-type": "white"                # Trustlist for access control
        }
    },
    "spec": {
        "selector": {
            "app": "kubectl-test"
        },
        "ports": [
            {
                "name": "service-0",
                "targetPort": 80,     # Container port
                "port": 12222,        # Access port (load balancer's port for accessing the workload)
                "protocol": "TCP",    #Protocol used to access the workload
                "type": "LoadBalancer"
            }
```

```
      ]
    }
}
```

Step 4 Click **OK**. Access the workload through the load balancer's IP address and port in the format of *<IP-address>:<port>*.

- If a trustlist is used for access control, only the IP addresses in the trustlist can access the load balancer.
- If a blocklist is used for access control, the IP addresses in the blocklist cannot access the load balancer.

----End

Table 4-10 Annotations for ELB access control

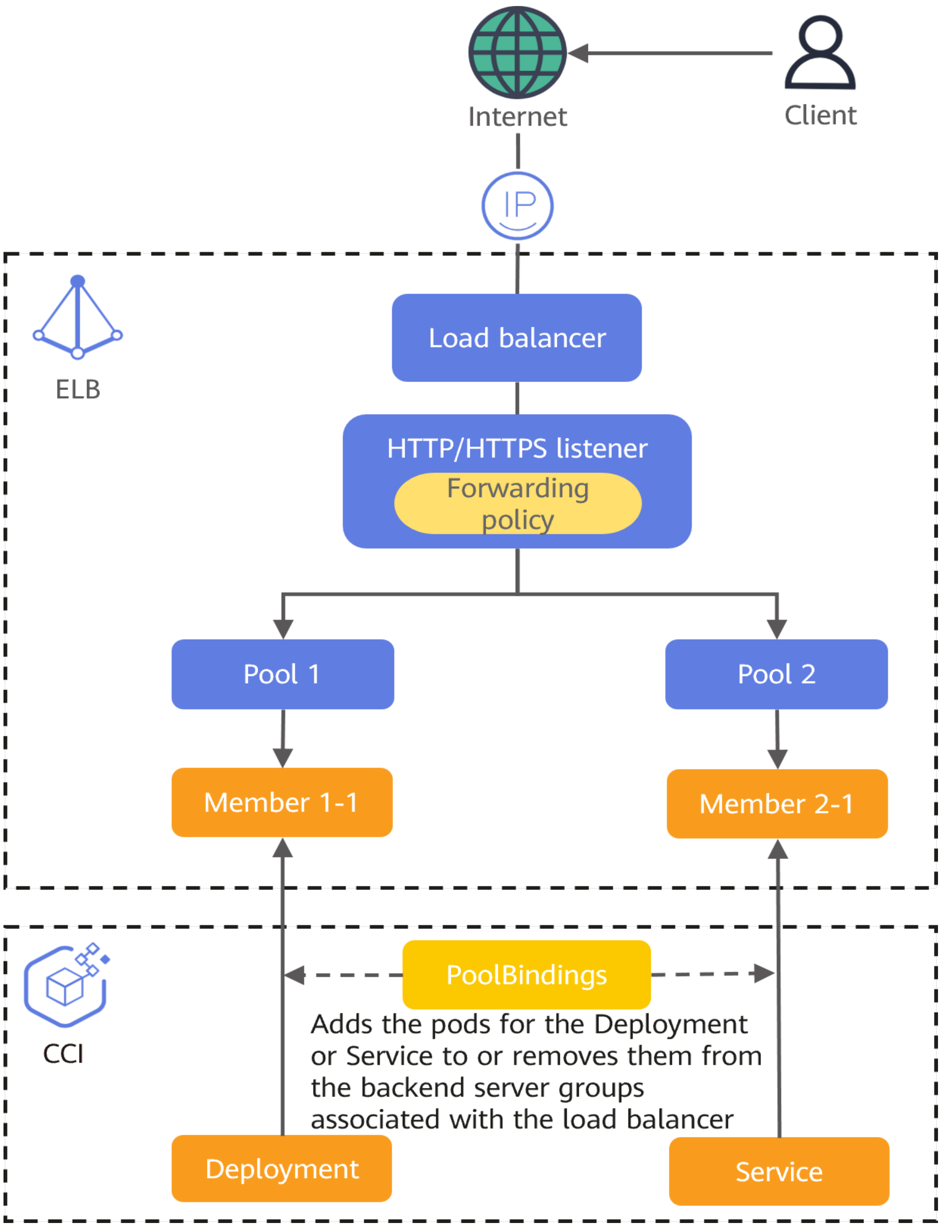
Parameter	Type	Description
kubernetes.io/elb.acl-id	String	● If this parameter is not specified, CCI does not modify access control on ELB. ● If this parameter is set to the IP address group ID of the load balancer, access control is enabled, and you need to configure an IP address blocklist or trustlist for the load balancer. ● You can enter a maximum of five IP address group IDs separated by commas (,). ● To obtain an IP address group ID, take the following steps: Log in to the console. In the **Service List**, choose **Networking** > **Elastic Load Balance**. On the Network Console, choose **Elastic Load Balance** > **IP Address Groups** and copy the **ID** of the target IP address group. For details, see IP Address Group.
kubernetes.io/elb.acl-type	String	This parameter is mandatory when you configure an IP address blocklist or trustlist for a load balancer. ● **black**: The selected IP address group cannot access the load balancer. ● **white**: Only the selected IP address group can access the load balancer. If **kubernetes.io/elb.acl-id** is specified but **kubernetes.io/elb.acl-type** is not, the trustlist is used by default.

4.2.2 Gateways

4.2.2.1 PoolBindings

4.2.2.1.1 PoolBinding Overview

After deploying a workload on CCI 2.0, you can use ELB to enable access to the workload over the public network. CCI 2.0 provides PoolBindings to handle a large number of HTTP/HTTPS requests. A PoolBinding can automatically add the pods of a Deployment or Service to or remove them from a backend server group (pool) associated with the load balancer. You can specify the load balancer and listener configurations as needed and configure advanced forwarding policies on the ELB console for load balancing at Layer 7.



Constraints

- PoolBindings can only be associated with hybrid backend server groups whose forwarding mode is load balancing. Currently, CCI 2.0 supports backend pods with IPv4 addresses not backend pods with IPv6 addresses.
- A backend server group can only be associated with one PoolBinding, and backend server groups can be associated with dedicated load balancers but

not shared load balancers. The VPC where the load balancer and backend server group are running must be the same as that of the namespace where the PoolBinding is created.

- PoolBindings support two types of backend objects: Deployments and Services.
- Creating PoolBindings on the CCI 2.0 console is not supported. You can create PoolBindings using the **curl** command or ccictl. For details about how to configure ccictl, see **ccictl Configuration Guide**. PoolBindings can be created, deleted, and queried, but cannot be updated.
- For details about the constraints on the load balancer configuration, see **Notes and Constraints**.

4.2.2.1.2 Associating a Deployment with a PoolBinding

A PoolBinding can be associated with a Deployment. After a PoolBinding is created, the endpoints of the pods for the Deployment are automatically added to or removed from the associated backend server group.

Prerequisites

- You have created a backend server group on the ELB console.
- You have created a Deployment on the CCI 2.0 console.

Method 1: Managing PoolBindings Using ccictl

- poolbinding.yaml example

```
apiVersion: loadbalancer.networking.openvessel.io/v1
kind: PoolBinding
metadata:
  name: test-c1       //Name of the PoolBinding to be created
  namespace: test-ns  //Namespace of the PoolBinding. It must be in the same namespace as the
associated Deployment.
spec:
  poolRef:
    id: e08*****-****-****-****-********29c1 //ID of the created backend server group
  targetRef:
    kind: Deployment  //Type of the backend object. In this example, the backend object is a
Deployment.
    group: cci/v2     //Group of the backend object
    name: test-name   //Name of the backend object, which is the name of the created Deployment in
this example
    port: 80          //Target port number of the backend object. If the backend object is a Deployment,
the port is the container port.
```

- Common ccictl commands

```
// Create a PoolBinding.
ccictl create -f poolbinding.yaml
// Query details about all PoolBindings in a namespace.
ccictl get plb -n $namespace -oyaml
// Query a PoolBinding in a namespace.
ccictl get plb -n $namespace $name -oyaml
// Delete a PoolBinding.
ccictl delete plb -n $namespace $name
```

Table 4-11 Key parameter descriptions

Parameter	Description
$namespace-name	Namespace of the PoolBinding. It must be in the same namespace as the associated Deployment.
$name	PoolBinding name

Method 2: Managing PoolBindings Using curl Commands

- poolbinding.json example

```
{
    "apiVersion": "loadbalancer.networking.openvessel.io/v1",
    "kind": "PoolBinding",
    "metadata": {
        "name": "test-c1",       //Name of the PoolBinding to be created
        "namespace": "test-ns"   //Namespace of the PoolBinding. It must be in the same namespace as
the associated Deployment.
    },
    "spec": {
        "poolRef": {
            "id": "e08*****-****-****-****-********29c1"  //ID of the created backend server group
        },
        "targetRef": {
            "kind": "Deployment",    //Type of the backend object. In this example, the backend object is
a Deployment.
            "group": "cci/v2",       //Group of the backend object
            "name": "test-name",     //Name of the backend object, which is the name of the created
Deployment in this example
            "port": 80               //Target port number of the backend object. If the backend object is a
Deployment, the port is the container port.
        }
    }
}
```

- Common curl commands

```
// Create a PoolBinding.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings" -X POST -
d@poolbinding.json
// Query details about all PoolBindings in a namespace.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings"
// Query a PoolBinding in a namespace.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings/$name"
// Delete a PoolBinding.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings/$name" -X
DELETE
```

Table 4-12 Key parameters in the commands

Parameter	Description
$token	Token used to authenticate requests.

Parameter	Description
$endpoint	CCI 2.0 endpoint. An endpoint is the **request address** for calling an API. Endpoints vary with services and regions. An endpoint can be obtained from **Regions and Endpoints**.
$namespace-name	Namespace of the PoolBinding. It must be in the same namespace as the associated Deployment.
$name	PoolBinding name

4.2.2.1.3 Associating a Service with a PoolBinding

A PoolBinding can be associated with a Service. After a PoolBinding is created, the endpoints of the pods for the Service are automatically added to or removed from the associated backend server group.

Constraints

The backend server group associated with a PoolBinding cannot be the one associated with the load balancer of the Service.

Prerequisites

- You have created a backend server group on the ELB console.
- You have created a Deployment on the CCI 2.0 console.
- You have created a Service on the CCI 2.0 console.

Method 1: Managing PoolBindings Using ccictl

- poolbinding.yaml example
```
apiVersion: loadbalancer.networking.openvessel.io/v1
kind: PoolBinding
metadata:
  name: test-c1       //Name of the PoolBinding to be created
  namespace: test-ns  //Namespace of the PoolBinding. It must be in the same namespace as the
associated Service.
spec:
  poolRef:
    id: e08*****-****-****-****-********29c1 //ID of the created backend server group
  targetRef:
    kind: Service    //Type of the backend object. In this example, the backend object is a Service.
    group: cci/v2    //Group of the backend object
    name: test-name   //Name of the backend object, which is the name of the created Service in this
example
    port: 123        //Target port number of the associated backend object. If the backend object is a
Service, the value is the access port of the Service.
```
- Common ccictl commands
```
// Create a PoolBinding.
ccictl create -f poolbinding.yaml
// Query details about all PoolBindings in a namespace.
ccictl get poolbindings -n $namespace -oyaml
// Query a PoolBinding in a namespace.
ccictl get poolbindings -n $namespace $name -oyaml
// Delete a PoolBinding.
ccictl delete poolbindings -n $namespace $name
```

Table 4-13 Key parameter descriptions

Parameter	Description
$namespace-name	Namespace of the PoolBinding. It must be in the same namespace as the associated Deployment.
$name	PoolBinding name

Method 2: Managing PoolBindings Using curl Commands

- poolbinding.json example

```
{
    "apiVersion": "loadbalancer.networking.openvessel.io/v1",
    "kind": "PoolBinding",
    "metadata": {
        "name": "test-c1",       //Name of the PoolBinding to be created
        "namespace": "test-ns"  //Namespace of the PoolBinding. It must be in the same namespace as
the associated Service.
    },
    "spec": {
        "poolRef": {
            "id": "e08*****-****-****-****-********29c1"  //ID of the created backend server group
        },
        "targetRef": {
             "kind": "Service",   //Type of the backend object. In this example, the backend object is a
Service.
            "group": "cci/v2",    //Group of the backend object
            "name": "test-name",  //Name of the backend object, which is the name of the created
Service in this example
            "port": 123           //Target port number of the associated backend object. If the backend
object is a Service, the value is the access port of the Service.
        }
    }
}
```

- Common curl commands

```
// Create a PoolBinding.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings" -X POST -
d@poolbinding.json
// Query details about all PoolBindings in a namespace.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings"
// Query a PoolBinding in a namespace.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings/$name"
// Delete a PoolBinding.
curl -k -H "Content-Type: application/json" -H "X-Auth-Token: $token" "https://$endpoint/apis/
loadbalancer.networking.openvessel.io/v1/namespaces/$namespace-name/poolbindings/$name" -X
DELETE
```

Table 4-14 Key parameters in the commands

Parameter	Description
$token	Token used to authenticate requests.

Parameter	Description
$endpoint	CCI 2.0 endpoint. An endpoint is the **request address** for calling an API. Endpoints vary with services and regions. An endpoint can be obtained from **Regions and Endpoints**.
$namespace-name	Namespace of the PoolBinding. It must be in the same namespace as the associated Deployment.
$name	PoolBinding name

4.2.3 Specifying a Subnet for a Pod

Scenario

If the pod network contains multiple subnets, you can use the **yangtse.io/subnets** annotation to specify the subnets after a pod is created.

Constraints

- A maximum of 20 subnets can be specified for a pod.
- The subnets specified for a pod must be included in the pod network configuration.
- Multiple subnets are separated using commas (,).
- There must be idle IP addresses in each subnet, or the pod fail to be created due to insufficient IP addresses.

Using ccictl

You can add annotations to a workload to specify the subnets for each pod.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  annotations:
    description: ''
  labels: {}
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        vm.cci.io/pod-size-specs: 2.00_4.0
        resource.cci.io/pod-size-specs: 2.00_4.0
        metrics.alpha.kubernetes.io/custom-endpoints: '[{api:'''',path:'''',port:'''',names:''''}]'
        yangtse.io/subnets: ${subnetID1},${subnetID2}    # Subnets specified for the pod
        log.stdoutcollection.kubernetes.io: '{"collectionContainers": ["container-0"]}'
      labels:
        app: nginx
    spec:
      containers:
```

```
      - image: library/nginx:stable-alpine-perl
        name: container-0
        resources:
          limits:
            cpu: 2000m
            memory: 4096Mi
          requests:
            cpu: 2000m
            memory: 4096Mi
        command: []
        lifecycle: {}
    dnsPolicy: ''
    imagePullSecrets:
      - name: imagepull-secret
    dnsConfig: {}
minReadySeconds: 0
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 0
    maxUnavailable: 1
```

NOTE

yangtse.io/subnets: IDs of the subnets to be specified for a pod

4.3 Auto Scaling

4.3.1 Horizontal Scaling

A HorizontalPodAutoscaler (HPA) is a resource object that can dynamically scale the number of pods running a Deployment based on certain metrics so that services running on the Deployment can adapt to metric changes. HPA automatically increases or decreases the number of pods based on the configured policy to meet your requirements.

Constraints

There are system components that help run the pods. These system components occupy some underlying resources, such as vCPU and memory. As a result, the resource usages for the pods may not reach the expected limits, and the threshold specified in the HPA policy may not be reached. To avoid this, refer to Reserved System Overhead.

Auto Scaling

NOTE

- If **spec.metrics.resource.target.type** is set to **Utilization**, you need to specify the resource requests value when creating a workload.
- When **spec.metrics.resource.target.type** is set to **Utilization**, the resource usage is calculated as follows: Resource usage = Used resource/Available pod flavor. You can determine the actual flavor of a pod by referring to **Pod Flavor**.

A properly configured auto scaling policy includes the metric, threshold, and step. It eliminates the need to manually adjust resources in response to service changes and traffic bursts, thus helping you reduce workforce and resource consumption.

CCI 2.0 supports **auto scaling based on metrics**. The workloads are scaled out or in based on the vCPU or memory usage. You can specify a vCPU or memory usage threshold. If the usage is higher or lower than the threshold, pods are automatically added or deleted.

- Configure a metric-based auto scaling policy.
  a. Log in to the **CCI 2.0 console**.
  b. In the navigation pane, choose **Workloads**. On the **Deployments** tab, locate the target Deployment and click its name.
  c. On the **Auto Scaling** tab, click **Create from YAML** to configure an auto scaling policy.

     The following describes the auto scaling policy file format:

     - Resource description in the **hpa.yaml** file

```
kind: HorizontalPodAutoscaler
apiVersion: cci/v2
metadata:
  name: nginx            # Name of the HorizontalPodAutoscaler
  namespace: test         # Namespace of the HorizontalPodAutoscaler
spec:
  scaleTargetRef:        # Reference the resource to be automatically scaled.
    kind: Deployment      # Type of the target resource, for example, Deployment
    name: nginx          # Name of the target resource
    apiVersion: cci/v2    # Version of the target resource
  minReplicas: 1         # Minimum number of replicas for HPA scaling
  maxReplicas: 5         # Maximum number of replicas for HPA scaling
  metrics:
    - type: Resource                  # Resource metrics are used.
      resource:
        name: memory                  # Resource name, for example, cpu or memory
        target:
          type: Utilization          # Metric type. The value can be Utilization (percentage) or
AverageValue (absolute value).
          averageUtilization: 50     # Resource usage. For example, when the CPU usage
reaches 50%, scale-out is triggered.
  behavior:
    scaleUp:
      stabilizationWindowSeconds: 30  # Scale-out stabilization duration, in seconds
      policies:
      - type: Pods           # Number of pods to be scaled
        value: 1
        periodSeconds: 30  # The check is performed once every 30 seconds.
    scaleDown:
      stabilizationWindowSeconds: 30  # Scale-in stabilization duration, in seconds
      policies:
      - type: Percent      # The resource is scaleed in or out based on the percentage of
existing pods.
        value: 50
        periodSeconds: 30  # The check is performed once every 30 seconds.
```

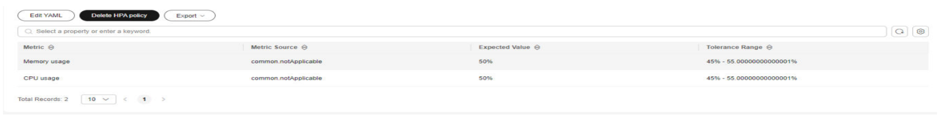
     - Resource description in the **hpa.json** file

```
{
    "kind": "HorizontalPodAutoscaler",
    "apiVersion": "cci/v2",
    "metadata": {
        "name": "nginx",             # Name of the the HorizontalPodAutoscaler
                "namespace": "test"          # Namespace of the HorizontalPodAutoscaler
    },
    "spec": {
                "scaleTargetRef": {         # Reference the resource to be automatically scaled.
            "kind": "Deployment",        # Type of the target resource, for example, Deployment
            "name": "nginx",           # Name of the target resource
            "apiVersion": "cci/v2"       # Version of the target resource
        },
        "minReplicas": 1,            # Minimum number of replicas for HPA scaling
        "maxReplicas": 5,              # Maximum number of replicas for HPA scaling
        "metrics": [
            {
                "type": "Resource",              # Resource metrics are used.
                "resource": {
                    "name": "memory",       # Resource name, for example, cpu or memory
                    "target": {
                        "type": "Utilization",           # Metric type. The value can be Utilization
(percentage) or AverageValue (absolute value).
                        "averageUtilization": 50         # Resource usage. For example, when the
CPU usage reaches 50%, scale-out is triggered.
                    }
                }
            }
        ],
        "behavior": {
            "scaleUp": {
                "stabilizationWindowSeconds": 30,
                "policies": [
                    {
                        "type": "Pods",
                        "value": 1,
                        "periodSeconds": 30
                    }
                ]
            },
            "scaleDown": {
                "stabilizationWindowSeconds": 30,
                "policies": [
                    {
                        "type": "Percent",
                        "value": 50,
                        "periodSeconds": 30
                    }
                ]
            }
        }
    }
}
```

d. Click **OK**.

   You can view the auto scaling policy on the **Auto Scaling** tab.

   Figure 4-10 Auto scaling policy

Metric	Metric Source	Expected Value	Tolerance Range
Memory usage	common.notApplicable	50%	45% - 55.00000000000001%
CPU usage	common.notApplicable	50%	45% - 55.00000000000001%

Total Records: 2 10 1

   When the trigger condition is met, the auto scaling policy will be executed.

4.4 Storage Management

4.4.1 Overview

CCI 2.0 supports both persistent storage and ephemeral storage to meet your requirements. You can use the following types of storage volumes when creating a workload.

Ephemeral Storage

By default, CCI 2.0 allocates 30 GiB of free ephemeral storage space to a pod. The storage space is unavailable when the pod is not in use. If you need to use more ephemeral storage space, you can expand the capacity. For details, see **Ephemeral Storage**.

SFS Turbo Volumes

CCI 2.0 allows you to create SFS Turbo volumes and mount them to specific container paths. SFS Turbo volumes are fast, on-demand, and scalable. They are suitable for DevOps, containerized microservices, and enterprise office applications. For details, see **SFS Turbo Volumes**.

OBS Parallel File System Volumes

CCI 2.0 allows you to create volumes from OBS parallel file systems. Parallel file systems are optimized object-based file systems. They are designed for big data scenarios where OBS is used as the unified data lake storage. It features access latency in milliseconds, TB/s-level bandwidth, millions of IOPS, and high compatibility, performance, scalability, and reliability. For details, see **OBS Parallel File System Volumes**.

4.4.2 Ephemeral Storage

Scenarios

By default, CCI 2.0 allocates 30 GiB of free ephemeral storage space to a pod. The storage space is unavailable when the pod is not in use. If the pod needs to write a large amount of data to rootfs or emptyDir or if the image size is greater than 30 GiB, you need to expand the ephemeral storage capacity.

Constraints

- The maximum ephemeral storage capacity is 994 GiB.
- When the ephemeral storage space is expanded, workload creation is slower than before because the capacity expansion takes time.

Using YAML

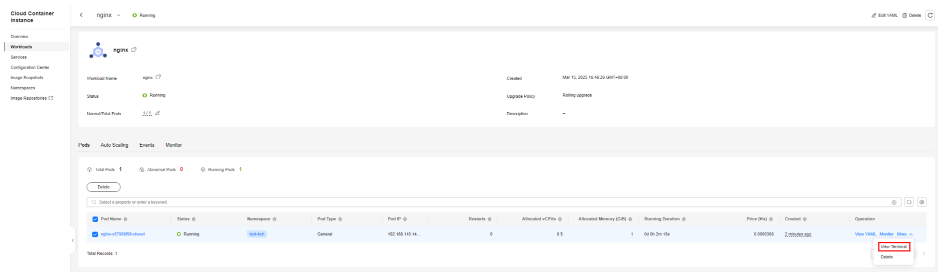
- When creating a pod, you need to add 10 GiB of ephemeral storage space. The YAML file is defined as follows:
  ```
  apiVersion: cci/v2
  kind: Pod
  metadata:
    name: nginx
    namespace: ns
  spec:
    extraEphemeralStorage:
      sizeInGiB: 10
    containers:
    - name: container-1
      image: nginx:latest
  ```
- When creating a Deployment, you need to add 10 GiB of ephemeral storage space. The YAML file is defined as follows:

```
apiVersion: cci/v2
kind: Deployment
metadata:
  name: nginx
  namespace: ns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      extraEphemeralStorage:
        sizeInGiB: 10
      containers:
      - image: nginx:latest
        name: container-0
```

Verifying the Capacity Expansion

Step 1 Log in to the CCI 2.0 console.

Step 2 In the navigation pane, choose **Workloads**. Locate the target workload and click its name to go to the details page. On the **Pods** tab, locate the target pod and click **View Terminal**.



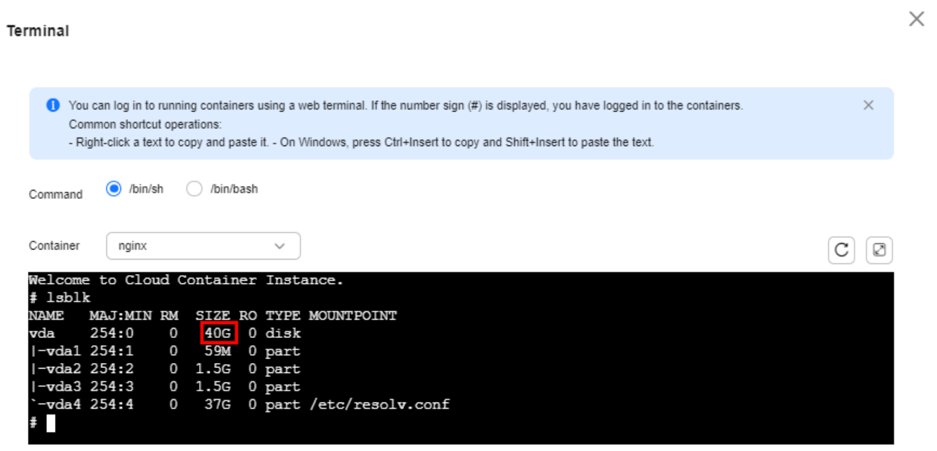
Workload Name: nginx
Status: Running
Normal/Total Pods: 1/1
Created: Mar 15, 2025 16:46:26 GMT+08:00
Upgrade Policy: Rolling upgrade
Description: --

Total Pods 1 Abnormal Pods 0 Running Pods 1

Pod Name	Status	Namespace	Pod Type	Pod IP	Restarts	Allocated vCPUs	Allocated Memory (GiB)	Running Duration	Price (¥/s)	Created	Operation
nginx-c67986f98-cbwwl	Running	test-hch	General	192.168.110.14...	0	0.5	1	0d 0h 2m 18s	0.0000306	2 minutes ago	View YAML Monitor More

Total Records: 1

Step 3 Enter **lsblk** and press **Enter** to check the system disk size after the expansion. (The ephemeral storage capacity is 30 GiB by default.)



You can log in to running containers using a web terminal. If the number sign (#) is displayed, you have logged in to the containers.
Common shortcut operations:
- Right-click a text to copy and paste it. - On Windows, press Ctrl+Insert to copy and Shift+Insert to paste the text.

Command: [✓ /bin/sh] /bin/bash
Container: nginx

```
Welcome to Cloud Container Instance.
# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda    254:0    0   40G  0 disk
|-vda1 254:1    0   59M  0 part
|-vda2 254:2    0  1.5G  0 part
|-vda3 254:3    0  1.5G  0 part
`-vda4 254:4    0   37G  0 part /etc/resolv.conf
#
```

----End

4.4.3 SFS Turbo Volumes

CCI 2.0 allows you to mount **SFS** Turbo volumes to container paths. SFS Turbo volumes are fast, on-demand, and scalable. They are suitable for DevOps, containerized microservices, and enterprise office applications.

Constraints

- SFS Turbo file systems are billed on a pay-per-use basis. For more information, see **SFS Turbo Pricing**.
- The SFS Turbo file system must be in the same VPC as the workload. If they are in different VPCs, the workload cannot use the SFS Turbo file system for persistent storage.
- If an SFS Turbo file system is in use, the VPC where the file system is deployed cannot be changed. Once the VPC is changed, the containers in CCI 2.0 will not be able to access the file system.
- If an SFS Turbo file system is deleted, containers in CCI 2.0 will become unavailable.
- SFS Turbo file systems do not involve AZs, so PVs of the SFS Turbo type do not support AZ affinity.
- CCI 2.0 supports SFS Turbo only in the TR-Istanbul and AF-Johannesburg regions.

Importing SFS Turbo File Systems

Currently, SFS Turbo file systems can only be used by CCI 2.0 containers through static PVC binding.

Step 1 Create an SFS Turbo file system. For details, see **Creating an SFS Turbo File System**.

Step 2 Create a PV.

1. Obtain the IAM token.

```
 curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -X POST -d
'{"auth": {"identity":{"methods": ["password"],"password":{"user": {"name": "$username","password":
"$password","domain": {"name": "$domain"}}}},"scope": {"project":{"name": "$project"}}}}' https://
iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens | grep "X-Subject-Token"|awk -F ": " '{print $2}'|
sed "s/\r//"
```

```
[r[redacted] ~]# curl -i -k -H 'Accept:application/json' -H 'Content-Type:application/json;charset=utf8' -X POST -d '{"auth": {"identity":{"methods": ["password"],"password":{"user": {"name": "[redacted]","password": "[redacted]","domain": {"name": "[redacted]"}}}},"scope": {"project":{"name": "[redacted]"}}}}' https://iam.[redacted].myhuaweicloud.com/v3/auth/tokens | grep "X-Subject-Token"|awk -F ": " '{print $2}'|sed "s/\r//"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  246k  100  246k  100   216  2791k   2447 --:--:-- --:--:-- --:--:-- 2801k
[redacted]
```

2. Call the API for creating a PV.

```
 curl -H "X-Auth-Token:$token" -H 'Content-Type: application/json' -X POST -d @pv.json  -k -v https://
cci.cn-north-4.myhuaweicloud.com/apis/cci/v2/persistentvolumes
```

In this command, **$token** is the IAM token obtained in **Step 2.1**, and **pv.json** is the information about the PV to be created. The following is an example:

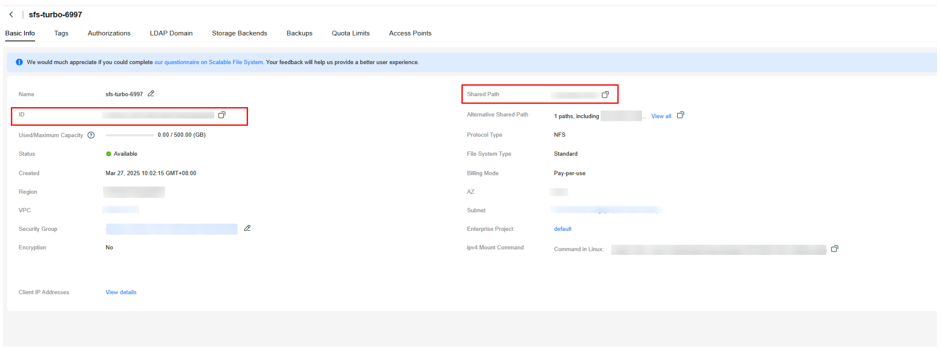
```
{
        "apiVersion": "cci/v2",
        "kind": "PersistentVolume",
    "metadata": {
        "annotations": {
        },
        "name": "pv-sfs-test"
    },
    "spec": {
        "accessModes": [
            "ReadWriteMany"
        ],
        "capacity": {
            "storage": "500Gi"
        },
        "csi": {
            "driver": "sfsturbo.csi.everest.io",
            "fsType": "nfs",
            "volumeHandle": "**************",
            "volumeAttributes": {
                "everest.io/share-export-location": "*******"
            }
        },
        "persistentVolumeReclaimPolicy": "Retain",
        "storageClassName": "csi-sfsturbo",
        "mountOptions": [
        ]
    }
}
```

Table 4-15 Key parameters

Parameter	Mandatory	Type	Description
accessModes	Yes	List	Description: Storage access mode. Constraint: The value must be **ReadWriteMany** for SFS Turbo volumes.
driver	Yes	String	Description: Storage driver that the volume depends on. Constraint: The value must be **sfsturbo.csi.everest.io**.
fsType	Yes	String	Description: Storage instance type. Constraint: The value must be **nfs**, which indicates file system volumes.
volumeHandle	Yes	String	Description: ID of the SFS Turbo file system. Constraint: The ID must be that of an existing SFS Turbo file system.

Parameter	Mandatory	Type	Description
persistentVolumeReclaimPolicy	Yes	String	Description: PV reclaim policy. Constraint: Only the **Retain** policy is supported. **Retain**: When a PVC is deleted, both the PV and underlying storage are retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storage	Yes	String	Description: Storage capacity, in Gi. Constraint: Set it to the size of the SFS Turbo file system.
storageClassName	Yes	String	Description: Storage class name of the SFS Turbo volume. Constraint: The storage class name of SFS Turbo volumes is **csi-sfsturbo**.

volumeHandle is the ID of the SFS Turbo file system created in **Step 1**, and **everest.io/share-export-location** is the shared path of the SFS Turbo file system.

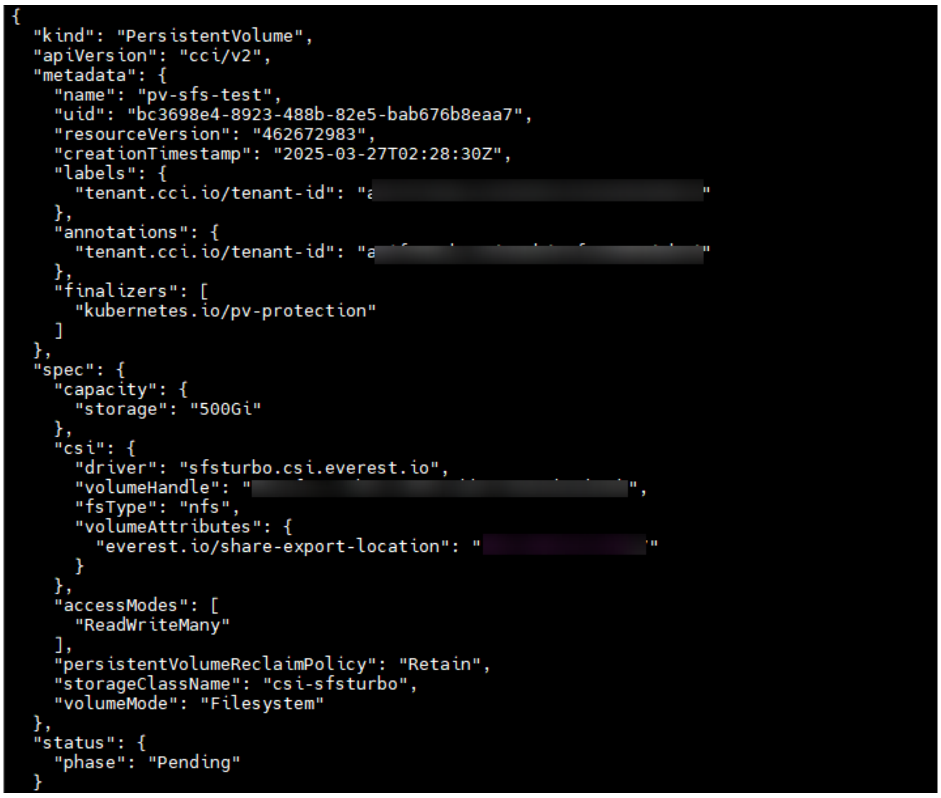

sfs-turbo-6997

Basic Info | Tags | Authorizations | LDAP Domain | Storage Backends | Backups | Quota Limits | Access Points

We would much appreciate if you could complete our questionnaire on Scalable File System. Your feedback will help us provide a better user experience.

Name	sfs-turbo-6997
ID	[redacted]
Used/Maximum Capacity	0.00 / 500.00 (GB)
Status	Available
Created	Mar 27, 2025 10:02:15 GMT+08:00
Region	[redacted]
VPC	[redacted]
Security Group	[redacted]
Encryption	No
Client IP Addresses	View details
Shared Path	[redacted]
Alternative Shared Path	1 paths, including [redacted].. View all
Protocol Type	NFS
File System Type	Standard
Billing Mode	Pay-per-use
AZ	[redacted]
Subnet	[redacted]
Enterprise Project	default
ipv4 Mount Command	Command in Linux: [redacted]

The following figure shows the response to the request for creating a PV.



```
{
  "kind": "PersistentVolume",
  "apiVersion": "cci/v2",
  "metadata": {
    "name": "pv-sfs-test",
    "uid": "bc3698e4-8923-488b-82e5-bab676b8eaa7",
    "resourceVersion": "462672983",
    "creationTimestamp": "2025-03-27T02:28:30Z",
    "labels": {
      "tenant.cci.io/tenant-id": "[redacted]"
    },
    "annotations": {
      "tenant.cci.io/tenant-id": "[redacted]"
    },
    "finalizers": [
      "kubernetes.io/pv-protection"
    ]
  },
  "spec": {
    "capacity": {
      "storage": "500Gi"
    },
    "csi": {
      "driver": "sfsturbo.csi.everest.io",
      "volumeHandle": "[redacted]",
      "fsType": "nfs",
      "volumeAttributes": {
        "everest.io/share-export-location": "[redacted]"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "storageClassName": "csi-sfsturbo",
    "volumeMode": "Filesystem"
  },
  "status": {
    "phase": "Pending"
  }
```

Step 3 Create a PVC.

1. Obtain the IAM token. For details, see **Step 2.1**.
2. Call the API for creating a PVC and bind the created PV.

```
curl -H "X-Auth-Token:$token" -H 'Content-Type: application/json' X POST -d @pvc.json  -k -v https://
cci.cn-north-4.myhuaweicloud.com/apis/cci/v2/namespaces/${namespace}/persistentvolumeclaims
```

pvc.json indicates the information about the PVC to be created. The following is an example:

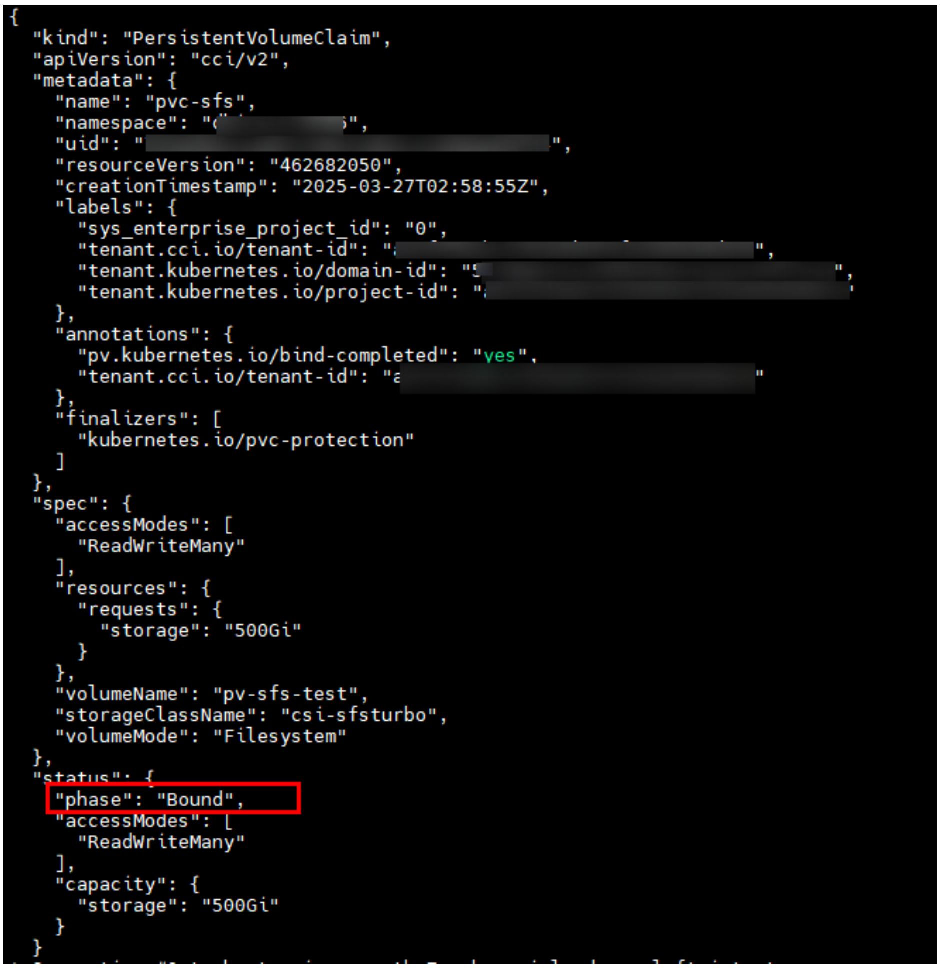
```
{
      "apiVersion": "cci/v2",
      "kind": "PersistentVolumeClaim",
      "metadata": {
         "name": "pvc-sfs",
         "annotations": {
         }
      },
      "spec": {
         "accessModes": [
            "ReadWriteMany"
         ],
         "resources": {
            "requests": {
               "storage": "500Gi"
            }
         },
         "storageClassName": "csi-sfsturbo",
         "volumeName": "pv-sfs-test"
      }
   }
```

Table 4-16 Key parameters

Parameter	Mandatory	Type	Description
storage	Yes	String	Description: PVC capacity, in Gi. Constraints – Set it to the size of the SFS Turbo file system. – The value is the same as the capacity set for the PV in **Table 4-15**.
storageClassName	Yes	String	Description: Storage class name. Constraints: The value must be the same as the storage class of the PV in **Table 4-15**. The storage class name of SFS Turbo volumes is **csi-sfsturbo**.
volumeName	Yes	String	Description: PV name. Constraint: The value must be the same as the PV name in **Table 4-15**.

volumeName indicates the name of the PV created in **Step 3**.

The following figure shows the response to the request for creating a PVC, and the PVC has been bound to the PV.



```
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "cci/v2",
  "metadata": {
    "name": "pvc-sfs",
    "namespace": "[redacted]",
    "uid": "[redacted]",
    "resourceVersion": "462682050",
    "creationTimestamp": "2025-03-27T02:58:55Z",
    "labels": {
      "sys_enterprise_project_id": "0",
      "tenant.cci.io/tenant-id": "[redacted]",
      "tenant.kubernetes.io/domain-id": "[redacted]",
      "tenant.kubernetes.io/project-id": "[redacted]
    },
    "annotations": {
      "pv.kubernetes.io/bind-completed": "yes",
      "tenant.cci.io/tenant-id": "[redacted]"
    },
    "finalizers": [
      "kubernetes.io/pvc-protection"
    ]
  },
  "spec": {
    "accessModes": [
      "ReadWriteMany"
    ],
    "resources": {
      "requests": {
        "storage": "500Gi"
      }
    },
    "volumeName": "pv-sfs-test",
    "storageClassName": "csi-sfsturbo",
    "volumeMode": "Filesystem"
  },
  "status": {
    "phase": "Bound",
    "accessModes": [
      "ReadWriteMany"
    ],
    "capacity": {
      "storage": "500Gi"
    }
  }
```

----End

Using SFS Turbo Volumes

For details, see **Deployments**. Add the volume configuration to the workload YAML file.

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: nginx
  namespace: test-ns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      volumes:
        - name: my-storage
          persistentVolumeClaim:
            claimName: PVC name
      containers:
        - name: nginx
          image: nginx:latest
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
          volumeMounts:
            - name: my-storage
              mountPath: mount path of a container
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: Default
      securityContext: {}
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 25%
```

NOTE

- When an SFS Turbo file system is being created, an independent VM will also be created, and this will take a long time. Therefore, you are advised to select existing SFS Turbo volumes.
- **subPath** is a sub-directory in the root path of the SFS Turbo file system. If there is no sub-directory, a sub-directory is automatically created in the SFS Turbo file system. Note that **subPath** must be a relative path.
- If SFS Turbo volumes are used, workloads can be created only using YAML or ccictl.

4.4.4 OBS Parallel File System Volumes

4.4.4.1 Parallel File System Overview

What Is a Parallel File System?

Parallel File System, a sub-product of OBS, is a high-performance file system. It aims to provide solutions for big data scenarios where OBS is used as the unified data lake storage. It features access latency in milliseconds, TB/s-level bandwidth, millions of IOPS, and high compatibility, performance, scalability, and reliability.

Different from the bucket structure, each directory in the access path of a parallel file system is independent. For example, **/dir01/dir02/example.txt** is a file and **/dir01/** and **/dir01/dir02/** are directories. In a hierarchical directory structure, you can rename a single directory. You do not need to list and modify all files with a specific directory prefix. This hierarchical structure makes the data organization of a parallel file system basically the same as that of Hadoop Distributed File System (HDFS). The big data analysis framework that uses HDFS as the data access layer can access data in a parallel file system through the OBSFileSystem plug-in (OBSA-HDFS). For details, see **About PFS**.

Performance

Every time when a volume created from a parallel file system is mounted to a workload, there will be a resident process at the backend for the volume. When a workload uses too many parallel file system volumes or reads and writes a large number of parallel file systems, resident processes will consume a significant amount of memory. **Table 4-17** list the used memory in some scenarios. To ensure that the workload can run normally, the number of parallel file system volumes used depends on the requested memory. For example, if the workload requests 4 GiB of memory, the workload can have no more than 4 parallel file system volumes.

Table 4-17 Memory used by a resident process per parallel file system

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

Prerequisites

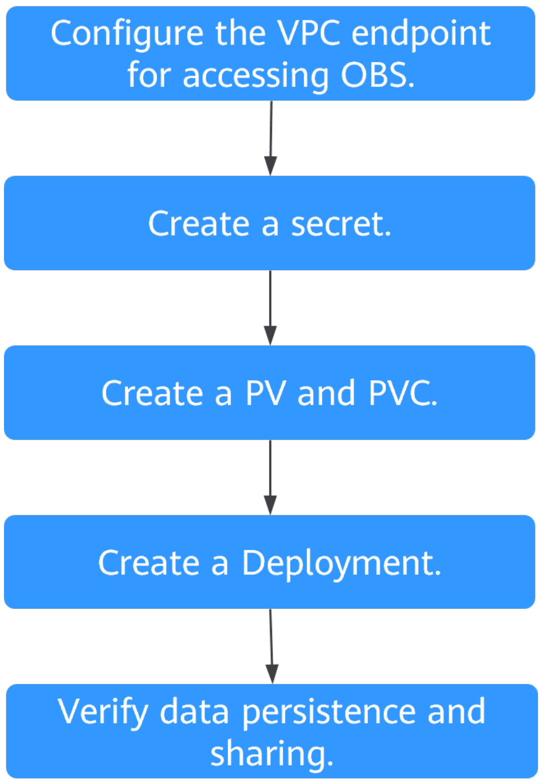
Before using a parallel file system for persistent data storage, you have configured a VPC endpoint for accessing OBS. Otherwise, the volume created from the parallel file system may fail to be mounted. You are advised to create all possible VPC endpoints for OBS at a time to avoid repeated operations when creating volumes from existing parallel file systems. For details about VPC Endpoint, see **What Is VPC Endpoint?** For details about how to create a VPC endpoint, see **Purchasing VPC Endpoints**. You can submit a service ticket or contact OBS O&M personnel to obtain the name of each VPC endpoint for OBS.

Scenarios

Only existing parallel file systems can be mounted to workloads as volumes. You need to use an existing parallel file system to create a PV and then mount the PV to a workload through a PVC. For details, see **Creating a Volume from an Existing Parallel File System**.

Process Flowchart

Figure 4-11 Process of using a parallel file system to create a volume


Configure the VPC endpoint for accessing OBS.
Create a secret.
Create a PV and PVC.
Create a Deployment.
Verify data persistence and sharing.

Billing

For details about the billing, see **OBS Billing**.

4.4.4.2 Setting Access Keys (AK/SK) for Mounting a Parallel File System Volume

Scenario

Before mounting a parallel file system volume to a pod, you need to set access keys (AK/SK). IAM users can use their own access keys to mount parallel file system volumes and control access to OBS. For details, see **Differences Between OBS Permissions Control Methods**.

Prerequisites

If you need to create access keys by running commands, you need to use ccictl to connect to CCI 2.0. For details, see **ccictl Configuration Guide**.

Constraints

When the access keys (AK/SK) are used by a parallel file system volume, the AK/SK cannot be deleted or disabled. Otherwise, the service containers cannot access the mounted parallel file system.

Obtaining Access Keys

Step 1 Log in to the management console.

Step 2 Hover the cursor over the username in the upper right corner and choose **My Credentials** from the drop-down list.

Step 3 In the navigation pane, choose **Access Keys**.

Step 4 Click **Create Access Key**. The **Create Access Key** dialog box is displayed.

Step 5 Click **OK** to download access keys.

----End

Creating a Secret Using Access Keys

Step 1 Obtain access keys.

Step 2 Encode the access keys using Base64. (Assume that the AK is *xxx* and SK is *yyy*.) Run the following commands on Linux:

```
echo -n xxx|base64
echo -n yyy|base64
```

Record the encoded AK and SK.

Step 3 Create a YAML file for the secret, for example, **secret-obs.yaml**.

```
apiVersion: cci/v2
data:
  access.key: WE5WWVhVNU*****
  secret.key: Nnk4emJyZ0*****
kind: Secret
metadata:
  name: secret-obs
  namespace: test-obs-v1
  labels:
    secret.kubernetes.io/used-by: csi
type: cci/secure-opaque
```

The parameters are described as follows:

Parameter	Mandatory	Type	Description
access.key	Yes	String	Description: AK after Base64 encoding.
secret.key	Yes	String	Description: SK after Base64 encoding.
name	Yes	String	Description: Secret name.
namespace	Yes	String	Description: Namespace of the secret.

Parameter	Mandatory	Type	Description
secret.kubernetes.io/used-by	Yes	String	Description: Secret label used by CSI storage. Constraint: The value must be **csi**.
type	Yes	String	Description: Key type. Constraint: The value must be **cci/secure-opaque**. If this value is used, the data you enter will be automatically encrypted.

Step 4 Create a secret.

```
ccictl create -f secret-obs.yaml
```

----End

Follow-up Operations

After setting the access keys (AK/SK), you can **use an existing parallel file system to create a storage volume**.

4.4.4.3 Creating a Volume from an Existing Parallel File System

This section describes how to use an existing parallel file system to create a PV and PVC for data persistence and sharing in the workload.

Prerequisites

- You have set the access keys (**Setting Access Keys (AK/SK) for Mounting a Parallel File System Volume**).
- If you need to create access keys by running commands, you need to use ccictl to connect to CCI 2.0. For details, see **ccictl Configuration Guide**.

Constraints

- When parallel file systems are used, the group and permission of the mount point cannot be modified.
- Every time when a volume created from a parallel file system is mounted to a workload through the PVC, there will be a resident process at the backend for the volume. When a workload uses too many parallel file system volumes or reads and writes a large number of parallel file systems, resident processes will consume a significant amount of memory. To ensure that the workload can run normally, the number of parallel file system volumes used depends on the requested memory. For example, if the workload requests 4 GiB of memory, the workload can have no more than 4 parallel file system volumes.
- If parallel file systems are used, read-only mounting is not supported. For details about how to configure permissions for parallel file systems, see **Permissions Configuration**.
- Multiple PVs can use the same parallel file system if the following requirements are met:

- Multiple PVCs or PVs that use the same parallel file system cannot be mounted to a single pod. Doing so will cause pod startup failures, as not all PVCs can be mounted due to identical **volumeHandle** value.
- The **persistentVolumeReclaimPolicy** parameter in the PVs should be set to **Retain**. If any another value is used, when a PV is deleted, the associated parallel file system may be deleted. In this case, other PVs associated with the parallel file system will malfunction.
- If the parallel file system is repeatedly used, you must maintain data consistency. Enable application-layer isolation and protection for ReadWriteMany to prevent multiple clients from writing data to the same file, thereby avoiding data overwriting and loss.

- CCI 2.0 supports parallel file system volumes in CN-Hong Kong.

Using ccictl

Step 1 Use ccictl to connect to CCI 2.0.

Step 2 Create a PV.

1. Create the **pv-obs.yaml** file.

```
apiVersion: cci/v2
kind: PersistentVolume
metadata:
  name: pv-obs                              # PV name.
spec:
  accessModes:
  - ReadWriteMany                              # The access mode must be ReadWriteMany for parallel file
system volumes.
  capacity:
    storage: 1Gi                            # Storage capacity. This parameter is only for verification. Its
value cannot be empty or 0, and any value you set does not take effect for parallel file systems.
  csi:
    driver: obs.csi.everest.io                  # Storage driver that the volume depends on.
    fsType: obsfs                           # Instance type.
    volumeHandle: <your_file_system_name>   # Name of the parallel file system.
    nodePublishSecretRef:                      # Secret for the parallel file system volume.
      name: <your_secret_name>                 # Secret name.
      namespace: <your_namespace>              # Name of the secret.
  persistentVolumeReclaimPolicy: Retain         # Reclaim policy.
  storageClassName: csi-obs                     # Storage class name.
  mountOptions: []                              # Mount options.
```

Table 4-18 Key parameters

Parameter	Mandatory	Type	Description
accessModes	Yes	List	Description: Storage access mode. Constraint: The value must be **ReadWriteMany** for parallel file system volumes.

Parameter	Mandatory	Type	Description
driver	Yes	String	Description: Storage driver that the volume depends on. Constraint: The value must be **obs.csi.everest.io** for parallel file system volumes.
fsType	Yes	String	Description: Storage instance type. Constraint: The value must be **obsfs**, which indicates parallel file systems.
volumeHandle	Yes	String	Description: Name of the parallel file system. Constraint: The value must be the name of an existing parallel file system.

nodePublishSecretRef	Yes	Object	Description: Access keys (AK/SK) that can be used to mount the parallel file system volume. You can create a secret using the AK/SK and use this secret for the PV. For details, see **Setting Access Keys (AK/SK) for Mounting a Parallel File System Volume**.
			The following is an example:
			`nodePublishSecretRef:`
			`  name: secret-demo`
			`  namespace: default`

mountOptions	No	List	Description: Mount options. For details, see **Setting Mount Options for a Parallel File System Volume**.
persistentVolumeReclaimPolicy	Yes	String	Description: PV reclaim policy. Constraints: Only the **Retain** policy is supported. **Retain**: When a PVC is deleted, both the PV and underlying storage are retained. You need to manually delete these resources. After the PVC is deleted, the PV is in the **Released** state and cannot be bound to a PVC again.
storage	Yes	String	Description: Storage capacity, in Gi. Constraint: For parallel file systems, this parameter is only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for parallel file systems.

Parameter	Mandatory	Type	Description
storageClassName	Yes	String	Description: Storage class name of the parallel file system volume. Constraint: The value is **csi-obs** for parallel file system volumes.

2. Create a PV.
```
ccictl apply -f pv-obs.yaml
```

Step 3 Create a PVC.

1. Create the **pvc-obs.yaml** file.
```
apiVersion: cci/v2
kind: PersistentVolumeClaim
metadata:
  name: pvc-obs
  namespace: test-obs-v1
  annotations:
    csi.storage.k8s.io/fstype: obsfs
    csi.storage.k8s.io/node-publish-secret-name: <your_secret_name>          # Secret name.
    csi.storage.k8s.io/node-publish-secret-namespace: <your_namespace>       # Namespace of the
secret.
spec:
  accessModes:
  - ReadWriteMany                                                 # The access mode of parallel file system
volumes must be ReadWriteMany.
  resources:
    requests:
      storage: 1Gi
 storageClassName: csi-obs                                        # Storage class name, which must be
the same as that of the PV.
  volumeName: pv-obs                                              # Name of the PV.
```

Table 4-19 Key parameters

Parameter	Mandatory	Type	Description
fsType	Yes	String	Description: Storage instance type. Constraint: The value must be **obsfs**, which indicates parallel file systems.
csi.storage.k8s.io/node-publish-secret-name	Yes	String	Description: Name of the secret specified for the PV.
csi.storage.k8s.io/node-publish-secret-namespace	Yes	String	Description: Namespace of the secret specified for the PV.

Parameter	Mandatory	Type	Description
storage	Yes	String	Description: PVC capacity, in Gi. Constraints – For parallel file systems, this parameter is only for verification (cannot be empty or **0**). Its value is fixed at **1**, and any value you set does not take effect for parallel file systems. – The value is the same as the capacity set for the PV in **Table 4-18**.
storageClassName	Yes	String	Description: Storage class name. Constraint: The value must be the same as the storage class of the PV in **Table 4-18**. The storage class name of parallel file system volumes is **csi-obs**.
volumeName	Yes	String	Description: PV name. Constraint: The value must be the same as the PV name in **Table 4-18**.

2. Create a PVC.
```
ccictl apply -f pvc-obs.yaml
```

Step 4 Create an application.

1. Create a file named **web-demo.yaml**. In this example, the parallel file system volume is mounted to the **/data** path.
```
apiVersion: cci/v2
kind: Deployment
metadata:
  name: web-demo
  namespace: test-obs-v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-demo
  template:
    metadata:
      labels:
        app: web-demo
    spec:
      containers:
      - name: container-1
        image: nginx:latest
        resources:
          limits:
            cpu: 500m
            memory: 1Gi
          requests:
            cpu: 500m
            memory: 1Gi
        volumeMounts:
        - name: pvc-obs-volume    # Volume name, which must be the same as the volume name in the
```

```
volumes field.
          mountPath: /data  # Location where the storage volume is mounted.
      imagePullSecrets:
        - name: imagepull-secret
      volumes:
        - name: pvc-obs-volume    # Volume name, which can be changed as needed.
          persistentVolumeClaim:
            claimName: pvc-obs    # PVC name.
```

2. Create a workload that the parallel file system volume is mounted to.
   ```
   ccictl apply -f web-demo.yaml
   ```
   After the workload is created, you can try to verify data persistence and sharing. For details, see **Verifying Data Persistence and Sharing**.

----End

Verifying Data Persistence and Sharing

Step 1 View the deployed application and files.

1. Run the following command to view the created pods:
   ```
   ccictl get pod -n test-obs-v1| grep web-demo
   ```
   The expected output is as follows:
   ```
   web-demo-7864446874-6d4lp   1/1     Running   0            52s
   web-demo-7864446874-xx6qh   1/1     Running   0            52s
   ```
2. Run the following commands in sequence to check the files in the **/data** path of the pods:
   ```
   ccictl exec web-demo-7864446874-6d4lp -n test-obs-v1 -- ls /data
   ccictl exec web-demo-7864446874-xx6qh -n test-obs-v1 -- ls /data
   ```
   If no result is returned for both pods, no file exists in the **/data** path.

Step 2 Create a file named **static** in the **/data** path.

```
ccictl exec web-demo-7864446874-6d4lp -n test-obs-v1 --  touch /data/static
```

Step 3 View the files in the **/data** path.

```
ccictl exec web-demo-7864446874-6d4lp -n test-obs-v1 -- ls /data
```

The expected output is as follows:

```
static
```

Step 4 Verify data persistence.

1. Delete the pod named **web-demo-7864446874-6d4lp**.
   ```
   ccictl delete pod web-demo-7864446874-6d4lp -n test-obs-v1
   ```
   The expected output is as follows:
   ```
   pod "web-demo-7864446874-6d4lp" deleted
   ```
   After the deletion, the Deployment controller automatically creates a replica.
2. View the created pod.
   ```
   ccictl get pod -n test-obs-v1 | grep web-demo
   ```
   In the command output below, **web-demo-7864446874-84slz** is the created pod.
   ```
   web-demo-7864446874-84slz   1/1     Running   0            110s
   web-demo-7864446874-xx6qh   1/1     Running   0            8m47s
   ```
3. Check whether the file in the **/data** path of the new pod has been modified:
   ```
   ccictl exec web-demo-7864446874-84slz -n test-obs-v1 -- ls /data
   ```
   The expected output is as follows:
   ```
   static
   ```

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

Step 5 Verify data sharing.

1. View the created pods.
```
ccictl get pod -n test-obs-v1 | grep web-demo
```
The expected output is as follows:
```
web-demo-7864446874-84slz   1/1     Running   0           6m3s
web-demo-7864446874-xx6qh   1/1     Running   0            13m
```
2. Create a file named **share** in the **/data** path of either pod. In this example, select the pod named **web-demo-7864446874-84slz**.
```
ccictl exec web-demo-7864446874-84slz -n test-obs-v1 --  touch /data/share
```
Check the files in the **/data** path of the pod.
```
ccictl exec web-demo-7864446874-84slz -n test-obs-v1 -- ls /data
```
The expected output is as follows:
```
share
static
```
3. Check whether the **share** file exists in the **/data** path of another pod (**web-demo-7864446874-xx6qh**) as well to verify data sharing.
```
ccictl exec web-demo-7864446874-xx6qh -n test-obs-v1 -- ls /data
```
The expected output is as follows:
```
share
static
```
After you create a file in the **/data** path of a pod, if the file is also created in the **/data** path of the other pod, the two pods share the same volume.

----End

4.4.4.4 Updating the Access Keys (AK/SK) for a Parallel File System Volume

If your service containers in a self-managed cluster use an OBS parallel file system for data storage and access, you must manually restart them whenever the volume's access keys are changed to apply the new keys. This process may interrupt services.

CCI can automatically apply the updated access keys (AK/SK) of parallel file system volumes. After the access keys of a volume are updated, CCI 2.0 automatically detects the change and applies the new keys to all affected workloads. This eliminates the need to manually restart these workloads, ensuring seamless service continuity during the key update.

Update the Access Keys (AK/SK)

Step 1 Log in to the CCI 2.0 console.

Step 2 In the navigation pane, choose **Configuration Center**. On the **Secrets** tab, locate the secret to be updated and click **Edit YAML** in the **Operation** column.

Step 3 Modify the AK/SK and then click **OK**.

CAUTION

Ensure that the updated access keys are valid and have the permissions to access the corresponding parallel file system volume. Otherwise, the workload cannot access the mounted parallel file system.

----End

4.4.4.5 Setting Mount Options for a Parallel File System Volume

This section describes how to configure mount options for a parallel file system volume. You can set mount options for a PV and then bind the PV to a PVC.

Mount Options for a Parallel File System Volume

When you mount a parallel file system volume, CCI 2.0 sets the parameters in **Table 4-20** and **Table 4-21** by default. The parameters in **Table 4-20** cannot be canceled. You can configure other mount options if needed. For details, see **Mounting a Parallel File System**.

Table 4-20 Mandatory mount options configured by default

Parameter	Mandatory	Type	Description
use_ino	Yes	No parameter value is required.	Description: inode numbers allocated by obsfs. This option is enabled by default in read/write mode.
big_writes	Yes	No parameter value is required.	Description: If this parameter is configured, the maximum size of the write cache can be changed.
nonempty	Yes	No parameter value is required.	Description: The mount directory can contain files.

Parameter	Mandatory	Type	Description
allow_other	Yes	No parameter value is required.	Description: Other users can access the mount point.
no_check_certificate	Yes	No parameter value is required.	Description: The server certificate is not verified.

Table 4-21 Optional mount options configured by default

Parameter	Mandatory	Type	Description
max_write	No	Integer	Description: This parameter is only valid when **big_writes** is configured. The recommended value is 128 KB. Default value: **131072**
ssl_verify_hostname	No	Integer	Description: Do not verify the SSL certificate using the host name. Default value: **0**
max_background	No	Integer	Description: The maximum number of requests that can be waited in the backend. The value is automatically used by the parallel file system volume. Default value: **100**
umask	No	Integer	Description: File permission umask. For example, if the **umask** value is **022**, the directory permission (the maximum permission is **777**) is **755** (777 – 022 = 755, rwxr-xr-x). Default value: **0**

Configuring Mount Options in a PV

You can use the **mountOptions** field to configure mount options in a PV. The options you can configure in **mountOptions** are listed in **Mount Options for a Parallel File System Volume**.

Step 1 Use ccictl to connect to CCI 2.0.

Step 2 Configure mount options in a PV. The following is an example:

```
apiVersion: cci/v2
kind: PersistentVolume
metadata:
  name: pv-obs                              # PV name.
spec:
  accessModes:
  - ReadWriteMany                           # The access mode must be ReadWriteMany for parallel file
system volumes.
  capacity:
    storage: 1Gi                        # Storage capacity. This parameter is only for verification. Its value
cannot be empty or 0, and any value you set does not take effect for parallel file systems.
  csi:
    driver: obs.csi.everest.io                 # Storage driver that the volume depends on.
    fsType: obsfs                       # Instance type.
    volumeHandle: <your_file_system_name>  # Name of the parallel file system.
    nodePublishSecretRef:                      # Secret for the parallel file system volume.
      name: <your_secret_name>                 # Secret name.
      namespace: <your_namespace>                # Name of the secret.
  persistentVolumeReclaimPolicy: Retain        # Reclaim policy.
  storageClassName: csi-obs                 # Storage class name.
  mountOptions:                              # Mount options.
  - umask=027
```

Step 3 Create a PVC, bind the created PV to it, and mount the PV to the containers in the workload. For details, see **Creating a Volume from an Existing Parallel File System**.

Step 4 Check whether the mount options are effective.

In this example, the PVC is mounted to the workload that uses the **nginx:latest** image. You can log in to the node where the pod with the mounted parallel file system volume is located and run the following command to view the process details:

```
ps -ef | grep obsfs
```

Information similar to the following is displayed:

```
root      1470      1  0 Jul03 ?        00:01:23 /bin/obsfs {your_obs_name} /mnt/paas/kubernetes/kubelet/pods/
{pod_uid}/volumes/kubernetes.io~csi/{your_pv_name}/mount -o url=https://{endpoint}:443 -o
endpoint={region} -o passwd_file=/opt/everest-host-connector/obstmpcred/{pod_uid}/{your_obs_name} -o
allow_other -o nonempty -o big_writes -o use_ino -o no_check_certificate -o ssl_verify_hostname=0 -o
max_background=100 -o umask=027 -o max_write=131072
```

----End

4.5 Configuration Center

4.5.1 ConfigMaps

ConfigMaps are objects that you can use to store the configurations required by applications. After you create a ConfigMap, you can use it as a file in a containerized application.

Creating a ConfigMap

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Configuration Center**.

Step 3 Select a namespace and click the **ConfigMaps** tab.

Step 4 Click **Create from YAML** in the upper left corner and edit the YAML file. For details about the YAML file, see **YAML format**.

NOTE

CCI 2.0 supports both JSON and YAML, and the file size cannot exceed 1 MiB.

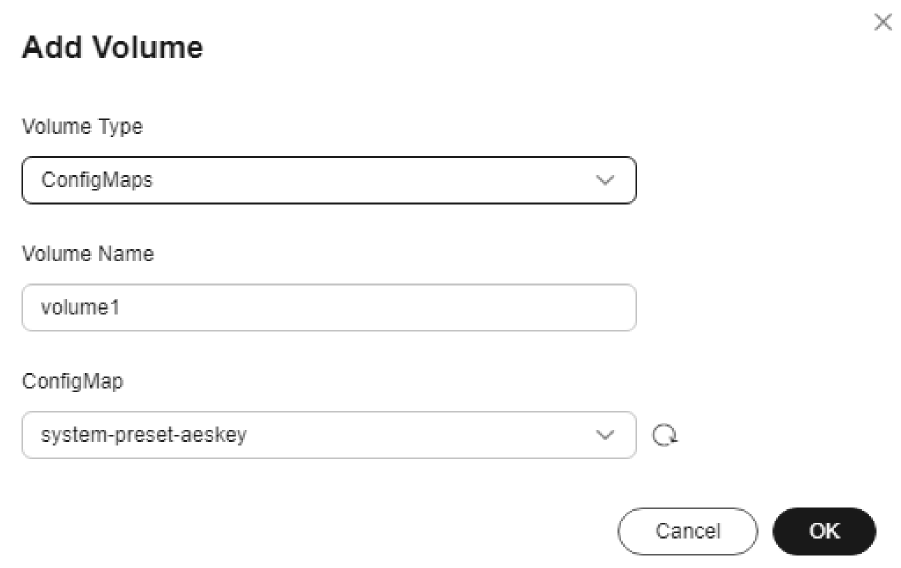
Step 5 Click **OK**.

----End

Using a ConfigMap

After a ConfigMap is created, you can mount it to a container as a storage volume during pod creation. For example, mount a ConfigMap named **system-preset-aeskey** to a container and set the storage volume name to **volume1**.

Figure 4-12 ConfigMaps


Add Volume

Volume Type: ConfigMaps

Volume Name: volume1

ConfigMap: system-preset-aeskey

ConfigMap File Format

A ConfigMap resource file must be in either JSON or YAML format, and the file size cannot exceed 1 MiB.

- YAML format

  Example file: **configmap.yaml**

```
apiVersion: cci/v2
kind: ConfigMap
```

```
metadata:
  name: configmap-example
data:
  key1: value1
  key2: value2
```

- JSON format

  Example file: **configmap.json**

```
{
    "apiVersion": "cci/v2",
    "kind": "ConfigMap",
    "metadata": {
        "name": "configmap-example"
    },
    "data": {
        "key1": "value1",
        "key2": "value2"
    }
}
```

4.5.2 Secrets

Secrets are objects that you can use to store sensitive data such as authentication information, certificates, and private keys. You can load a secret to a container as an environment variable when the container is started or mount a secret to a container as a file.

NOTE

It is recommended that you encrypt the uploaded secrets.

Creating a Secret

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Configuration Center**.

Step 3 Select a namespace and click the **Secrets** tab.

Step 4 Click **Create from YAML** in the upper left corner and edit the YAML file. For details about the YAML file, see **YAML format**.

NOTE

CCI 2.0 supports both JSON and YAML, and the file size cannot exceed 1 MiB.

Step 5 Click **OK**.

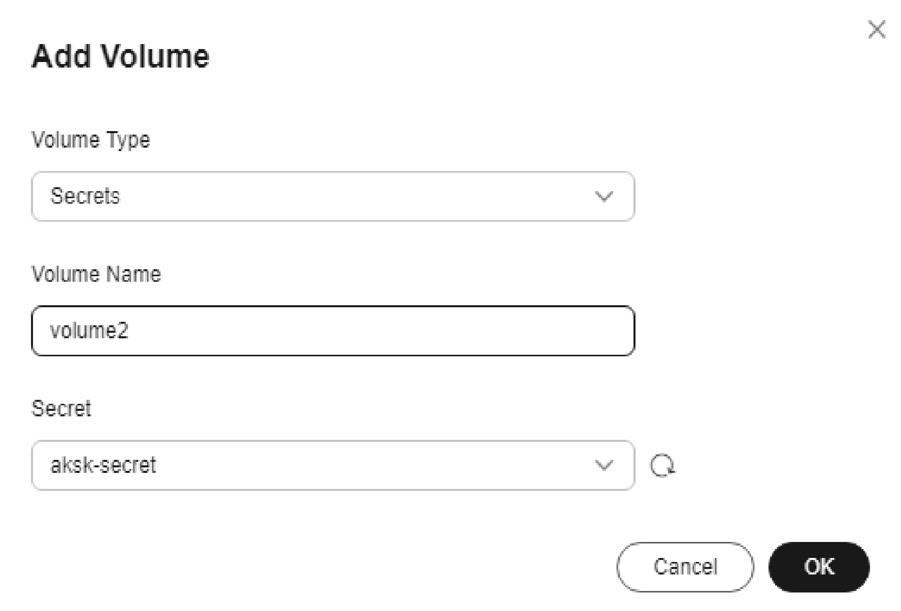
You can view the newly created secret in the secret list.

----End

Using a Secret

After a secret is created, you can mount it to a container as a storage volume during pod creation. For example, mount a secret named **aksk-secret** to a container and set the storage volume name to **volume2**.

Figure 4-13 Mounting a secret


Add Volume

Volume Type: Secrets

Volume Name: volume2

Secret: aksk-secret

Secret File Format

- **secret.yaml** resource description file

  For example, you can use a secret to obtain the following key-value pairs and encrypt them for an application:

  key1: value1

  key2: value2

  The **secret.yaml** file is defined as below. (Base64 encoding is required for the value of each key. For details about the Base64 encoding method, see **Base64 Encoding**.)

```
apiVersion: cci/v2
kind: Secret
metadata:
  name: mysecret           #Secret name
data:
  key1: dmFsdWUx    #Base64 encoding required
  key2: dmFsdWUy  #Base64 encoding required
type: Opaque         #The type must be Opaque.
```

- **secret.json** resource description file

  The content is as follows:

```
{
    "apiVersion": "cci/v2",
    "kind": "Secret",
    "metadata": {
        "name": "mysecret"
    },
    "data": {
        "key1": "dmFsdWUx",
        "key2": "dmFsdWUy"
    },
    "type": "Opaque"
}
```

Base64 Encoding

To perform Base64 encoding on a character string, run the **echo -n** *{Content to be encoded}* **| base64** command.

```
root@ubuntu:~# echo -n "3306" | base64
MzMwNg==
```

4.6 Images

4.6.1 Image Snapshots

4.6.1.1 Overview

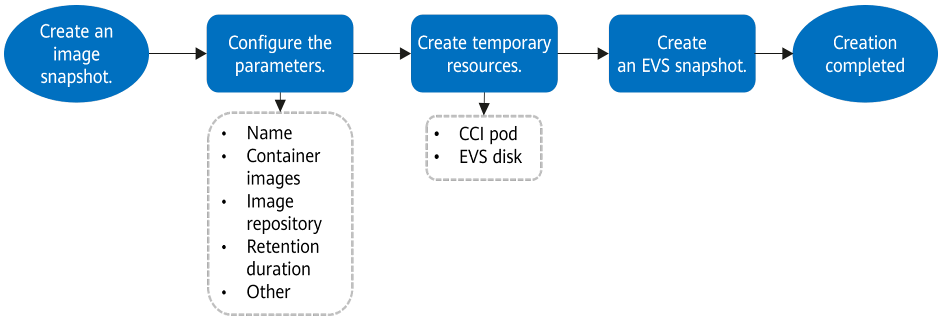
You can pull images from SWR, open-source image repositories, and self-managed image repositories to create image snapshots and then use them to create workloads without pulling the images, speeding up workload startup.

You can use an image snapshot to create a pod. There is no need to pull the image, and therefore pod startup time is reduced. This section describes the constraints on image snapshots and how you can use them.

Constraints

- An image snapshot can contain a maximum of 10 images.
- Private image repositories are supported. However, the access credentials of private image repositories must be provided, including the address and authentication information, such as the auth information in **~/.docker/config.json**.
- If the image needs to be pulled over the public network, you need to specify the public network access configuration in advance.
- If only some images in an image snapshot are consistent with those for the pod, inconsistent images still need to be pulled.
- If the image of a running pod changes and the new image is inconsistent with any image in the image snapshot, the new image needs to be pulled.
- By default, a CCI pod with 0.5 vCPUs and 1 GiB of memory is used to create an image snapshot. There are expenditures during the creation.

The following figure illustrates how to create an image snapshot.



How to Use

You can use an image snapshot to create a pod in either of the following ways:

- Automatic matching

The optimal image is selected from all available image snapshots you have created. The matching is performed in the following sequence:

a. Image matching degree: The images in the image snapshots are matched with those for the pod, and the mostly matched image snapshot is preferred.

b. Time when image snapshots were created: Image snapshots that were created later are preferred.

- Specific image snapshot

Specify the image snapshot to be used.

4.6.1.2 Creating an Image Snapshot

Scenario

This section describes how you can create an image snapshot. For details about how image snapshots work, see **Overview**.

Procedure

To run a container, you need to pull the specified container image first. However, due to factors such as the network and container image size, the pod startup speed slows down by image pull. You can create a snapshot using the image to be used. Then, you can use the snapshot to create pods without pulling the images, speeding up pod startup.

The following example creates an image snapshot named **my-imagesnapshot**.

```
apiVersion: cci/v2
kind: ImageSnapshot
metadata:
  name: 'my-imagesnapshot'
spec:
  buildingConfig:
    namespace: test-namespace
    eipID: xxxxxxxx
  imageSnapshotSize: 30
  ttlDaysAfterCreated: 7
  images:
    - image: 'nginx:stable-alpine-perl'
  registries:
    - imagePullSecret: imagepull-secret
      server: xxxxx.myhuaweicloud.com
      plainHTTP: true
```

Table 4-22 Parameter description

Field	Type	Mandatory	Example Value	Description
.metadata.name	String	Yes	my-imagesnapshot	Image snapshot name.
.spec.images.image	String	Yes	nginx:latest	Image used to create an image snapshot.
.spec.registries.server	String	Yes	serverA.com	Image repository address without the http:// or https:// prefix.
.spec.registries.imagePullSecret	String	No	imagepull-secret	Secret for accessing the image repository.
.spec.registries.plainHTTP	boolean	No	true	If the self-managed image repository address uses the HTTP protocol, set the value to **true**, or the image fails to be pulled due to different protocols. The default value is **false**.
.spec.registries.insecureSkipVerify	boolean	No	true	If the self-managed image repository address uses a self-issued certificate, set the value to **true** to skip certificate authentication, or the image fails to be pulled due to certificate authentication failure. The default value is **false**.
.spec.buildingConfig.namespace	String	Yes	my-namespace-a	User namespace. During image snapshot creation, you need to create a pod in the user namespace.
.spec.buildingConfig.eipID	String	No	3cxxxxe0-xxxx-xxxx-xxxx-8xxxxf3xxxx4	EIP used for pulling images from the public network.

Field	Type	Mandatory	Example Value	Description
.spec.buildingConfig.autoCreateEIP	boolean	No	true	Whether to automatically assign an EIP for the pod that will create the image snapshot. If **eipId** is specified, this parameter is ignored. If this parameter is set to **true**, you need to specify the EIP configuration using **autoCreateEIPAttribute**.
.spec.buildingConfig.autoCreateEIPAttribute.bandwidthChargeMode	String	No	bandwidth	Whether the billing is based on traffic or bandwidth. The value can be **traffic** or **bandwidth**. If this parameter is left blank or is an empty string, default value **bandwidth** is used. For IPv6 addresses, the default parameter value is **bandwidth** outside China and is **traffic** in China.
.spec.buildingConfig.autoCreateEIPAttribute.bandwidthSize	int	No	1000	Bandwidth size. The value ranges from 1 Mbit/s to 2,000 Mbit/s by default.
.spec.buildingConfig.autoCreateEIPAttribute.type	String	No	5_bgp	EIP type. The value can be **5_bgp** (dynamic BGP), **5_sbgp** (static BGP), or **5_youxuanbgp** (premium BGP).
.spec.buildingConfig.autoCreateEIPAttribute.ipVersion	int	No	4	EIP version. The value can be **4** or **6**. ● **4** indicates IPv4. If this parameter is left empty or is an empty string, an IPv4 address is assigned by default. ● **6** indicates IPv6. If the parameter is set to **6**, NAT64 is enabled.

Field	Type	Mandatory	Example Value	Description
.spec.buildingConfig.TimeoutMinutes	int	No	1440	Timeout interval for creating a snapshot, in minutes. The value is an integer from 30 to 10,080 (which is one week). The default value is **1440**, which is one day.
.spec.ttlDaysAfterCreated	integer	No	10	Retention period of the image snapshot, in days. Expired image snapshots will be deleted. The default value is **0**, indicating that the image snapshot never expires. When an image snapshot is used by a workload or pod, its expiration time is reset to the time when the image snapshot is used plus the image snapshot retention period. **NOTE** After an image snapshot expires, it still occupies the quota. You need to periodically review and delete expired image snapshots.
.spec.imageSnapshotSize	integer	No	20	Image snapshot size, in GiB. The default value is **20**.

4.6.1.3 Using an Image Snapshot

You can use an image snapshot to create a pod in either of the following ways:

- Automatic matching

  The optimal image is selected from all available image snapshots based on the following:

  a. Image matching degree: The images in the image snapshots are matched with those for the pod, and the mostly matched image snapshot is preferred.

  b. Time when image snapshots were created: Image snapshots that were created later are preferred.

- Specific image snapshot

  Specify the image snapshot to be used.

NOTE

If you create a pod using both the specified and automatically matched image snapshots and the two methods conflict, 400 will be returned.

Automatic Matching

When creating a pod, you can add the following annotation to enable automatic image snapshot matching.

Key	Example Value	Description
cci.io/image-snapshot-auto-match	"true"	Whether to enable automatic image snapshot matching.
cci.io/image-snapshot-usage-strategy	"size"	Automatic matching policy for image snapshots. The default value is **size**. ● **size**: selects the image snapshot with the largest sum of image sizes. ● **quantity**: selects the image snapshot that matches the largest number of images.

In the following example, a Deployment will be created.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  name: deployment-test
  namespace: ns-test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
      annotations:
        cci.io/image-snapshot-auto-match: "true"
    spec:
      containers:
        - image: redis
          name: container-0
          resources:
            limits:
              cpu: 500m
              memory: 1024Mi
            requests:
              cpu: 500m
              memory: 1024Mi
      imagePullSecrets:
        - name: imagepull-secret
```

Specifying an Image Snapshot

When creating a pod, you can add the following annotations to specify the image snapshot and interception policy.

Key	Example Value	Description
cci.io/image-snapshot-specified-name	"my-imagesnapshot"	Name of the specified image snapshot.
cci.io/image-snapshot-reject-if-not-available	"true"	If the specified image snapshot does not exist or is unavailable, the image is pulled from the image repository by default. To intercept pod creation, set the value to **"true"**.

In the following example, a Deployment will be created.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  name: deployment-test
  namespace: ns-test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
      annotations:
        cci.io/image-snapshot-specified-name: "my-imagesnapshot"
        cci.io/image-snapshot-reject-if-not-available: "true"
    spec:
      containers:
        - image: redis
          name: container-0
          resources:
            limits:
              cpu: 500m
              memory: 1024Mi
            requests:
              cpu: 500m
              memory: 1024Mi
      imagePullSecrets:
        - name: imagepull-secret
```

4.6.1.4 Managing Image Snapshots

Scenario

This section describes how to view and delete an image snapshot on the console. For details about how image snapshots work, see **Overview**.

Viewing an Image Snapshot

After an image snapshot is created, take the following steps to view the details:

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Image Snapshots**.

Step 3 View the name, status, remaining validity period, and size of the image snapshot. You can also view event details.

When the image snapshot status is **Available**, you can use the image snapshot.

If the image snapshot status is **Abnormal**, you can click **Events** to view the event details.

----End

Deleting Image Snapshots

If image snapshots are no longer used, take the following steps to delete them:

Step 1 Log in to the **CCI 2.0 console**.

Step 2 In the navigation pane, choose **Image Snapshots**.

Step 3 Select the image snapshots to be deleted and click **Delete** above the image snapshot list.

To delete an image snapshot, click **Delete** in the **Operation** column.

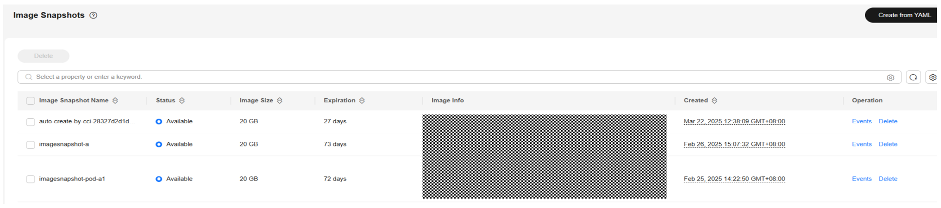


Image Snapshot Name	Status	Image Size	Expiration	Image Info	Created	Operation
auto-create-by-cci-28327d2d1d...	Available	20 GB	27 days	[redacted]	Mar 22, 2025 12:38:09 GMT+08:00	Events Delete
imagesnapshot-a	Available	20 GB	73 days	[redacted]	Feb 26, 2025 15:07:32 GMT+08:00	Events Delete
imagesnapshot-pod-a1	Available	20 GB	72 days	[redacted]	Feb 25, 2025 14:22:50 GMT+08:00	Events Delete

----End

4.6.2 Pulling an Image from a Self-Managed Image Repository

When an image is pulled from a self-managed image repository, the image may fail to be pulled due to different protocols or certificate authentication failures. In this section, HTTP and a self-issued certificate are used as examples to describe how to create a Deployment or pod by pulling an image from a self-managed image repository.

Configuration Description

Table 4-23 Configuration description

Annotation	Example Value	Configuration Description
cci.io/http-registries	"harbor.***.com,192.168.XX.XX:5000,100.95.XX.XX,http://harbor.***.com"	If you want to pull an image from a self-managed image repository using HTTP, you need to configure this annotation. The value can contain the https://prefix, port number, and relative path. Use commas (,) to separate multiple addresses, which can be private IP addresses, domain names, or public IP addresses. A maximum of 10 addresses are allowed.
cci.io/insecure-registries	"harbor.***.com,192.168.XX.XX:5000,100.95.XX.XX,https://harbor.***.com"	If you want to pull an image from a self-managed image repository using a self-issued certificate, you need to add this annotation to skip certificate authentication. The value can contain the https:// prefix, port number, and relative path. Use commas (,) to separate multiple addresses, which can be private IP addresses, domain names, or public IP addresses. A maximum of 10 addresses are allowed.

NOTE

- If the image repository address has a port number, the port number must be included. For example, if the image path is *192.168.XX.XX:5000/nginx:latest*, **cci.io/http-registries** can be set to *192.168.XX.XX:5000*.
- If HTTP is used, data transmission is not encrypted, and data is vulnerable to man-in-the-middle attacks and lacks identity authentication, which may cause data leak and service loss. HTTPS is recommended.

Example 1: Using HTTP for a Self-Managed Image Repository

- Creating a Deployment with 2 vCPUs and 4-GiB memory
```
apiVersion: cci/v2
kind: Deployment
metadata:
  labels:
    app: http
  name: http
spec:
  replicas: 1
  selector:
    matchLabels:
      app: http
  template:
    metadata:
      labels:
        app: http
      annotations:
        resource.cci.io/pod-size-specs: 2.00_4.0
        cci.io/http-registries: 192.168.XX.XX
    spec:
      containers:
        - image: 192.168.XX.XX/harbor/nginx:latest
          name: container-0
      imagePullSecrets:
        - name: harbor-secret-new
```
- Creating a pod with 2 vCPUs and 4-GiB memory
```
apiVersion: cci/v2
kind: Pod
metadata:
  annotations:
    resource.cci.io/pod-size-specs: 2.00_4.0
    cci.io/http-registries: 192.168.XX.XX
  name: http
spec:
  containers:
    - image: '192.168.XX.XX/harbor/nginx:latest'
      imagePullPolicy: IfNotPresent
      name: container-1
  imagePullSecrets:
    - name: harbor-secret
```

Example 2: Using a Self-Issued Certificate for a Self-Managed Image Repository

- Creating a Deployment with 2 vCPUs and 4-GiB memory
```
apiVersion: cci/v2
kind: Deployment
metadata:
  labels:
    app: insecure
  name: insecure
spec:
  replicas: 1
  selector:
    matchLabels:
      app: insecure
  template:
    metadata:
      labels:
        app: insecure
      annotations:
        resource.cci.io/pod-size-specs: 2.00_4.0
        cci.io/insecure-registries: 192.168.XX.XX
    spec:
      containers:
        - image: 192.168.XX.XX/harbor/nginx:latest
          name: container-0
      imagePullSecrets:
        - name: harbor-secret-new
```

- Creating a pod with 2 vCPUs and 4 GiB-memory

```
apiVersion: cci/v2
kind: Pod
metadata:
  annotations:
    resource.cci.io/pod-size-specs: 2.00_4.0
    cci.io/insecure-registries: 192.168.XX.XX
  name: insecure
spec:
  containers:
    - image: '192.168.XX.XX/harbor/nginx:latest'
      imagePullPolicy: IfNotPresent
      name: container-1
  imagePullSecrets:
    - name: harbor-secret
```

4.7 Monitoring

A Prometheus instance for cloud services can be used to monitor multiple metrics of CCI 2.0. For details, see **Prometheus Monitoring Overview**.

Constraints

Only one Prometheus instance for cloud services can be created in an enterprise project.

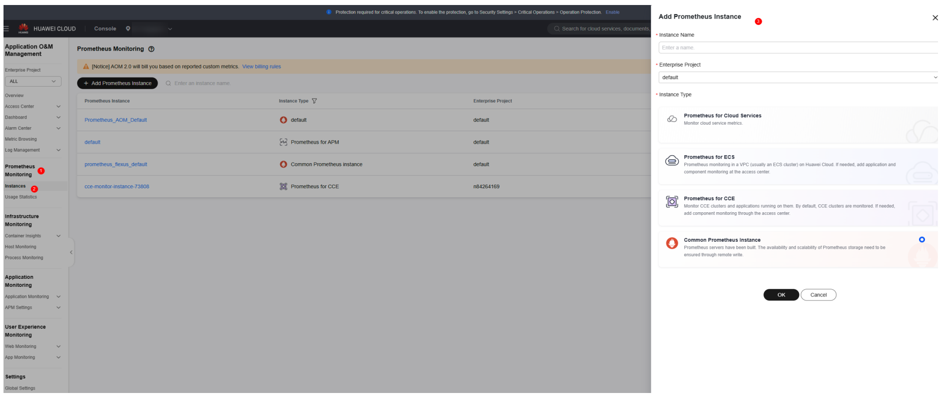
CCI 2.0 supports monitoring only in the TR-Istanbul, AF-Johannesburg, AP-Singapore, and ME-Riyadh regions.

Step 1: Create a Prometheus Instance

1. Log in to the AOM 2.0 console.
2. In the navigation pane on the left, choose **Metric Browsing** > **Prometheus Monitoring**. On the displayed page, click **Add Prometheus Instance**.
3. Set the instance name, enterprise project, and instance type.

   NOTE

   If the CCE Cloud Bursting Engine for CCI add-on is being used or required, the CCE cluster also needs this Prometheus instance, with the instance type set to **Prometheus for CCE**.

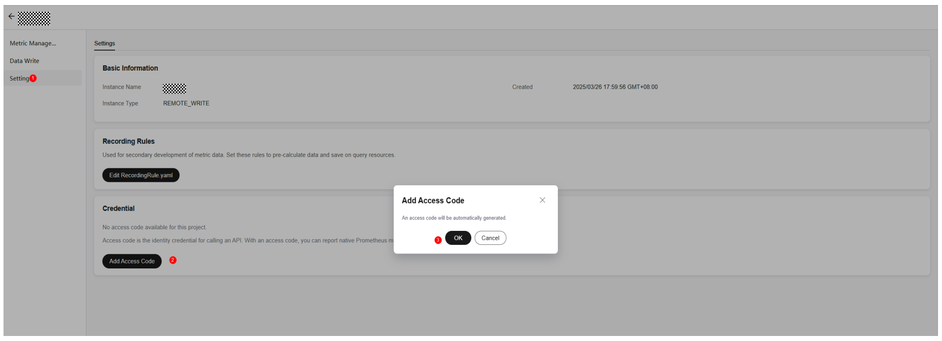

Protection required for critical operations. To enable the protection, go to Security Settings > Critical Operations > Operation Protection. Enable
HUAWEI CLOUD
Console
Search for cloud services, documents,
Application O&M Management
Enterprise Project
ALL
Overview
Access Center
Dashboard
Alarm Center
Metric Browsing
Log Management
Prometheus Monitoring
Instances
Usage Statistics
Infrastructure Monitoring
Container Insights
Host Monitoring
Process Monitoring
Application Monitoring
Application Monitoring
APM Settings
User Experience Monitoring
Web Monitoring
App Monitoring
Settings
Global Settings
Prometheus Monitoring
[Notice] AOM 2.0 will bill you based on reported custom metrics. View billing rules
Add Prometheus Instance
Enter an instance name.
1
2
3
Add Prometheus Instance
Instance Name
Enter a name.
Enterprise Project
default
Instance Type
Prometheus for Cloud Services
Monitor cloud service metrics.
Prometheus for ECS
Prometheus monitoring in a VPC (usually an ECS cluster) on Huawei Cloud. If needed, add application and component monitoring at the access center.
Prometheus for CCE
Monitor CCE clusters and applications running on them. By default, CCE clusters are monitored. If needed, add component monitoring through the access center.
Common Prometheus Instance
Prometheus servers have been built. The availability and scalability of Prometheus storage need to be ensured through remote write.
OK
Cancel

Prometheus Instance	Instance Type	Enterprise Project
Prometheus_AOM_Default	default	default
default	Prometheus for APM	default
prometheus_flexus_default	Common Prometheus instance	default
cce-monitor-instance-73808	Prometheus for CCE	n84264169

4. Click **OK**.

Step 2: Obtain the Access Code

1. Click the name of the created Prometheus instance.
2. In the navigation pane, choose **Settings**. In the **Credential** area, click **Add Access Code**.
3. Click **OK**.



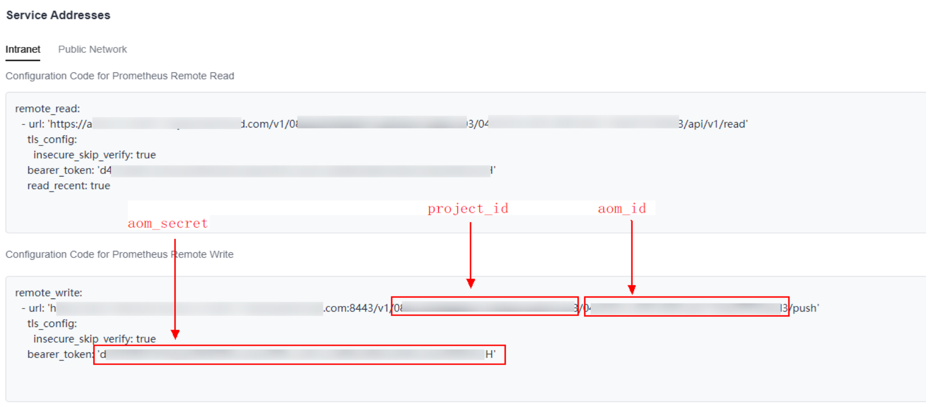
Basic Information
Instance Name [redacted]
Created 2025/03/26 17:59:56 GMT+08:00
Instance Type REMOTE_WRITE

Recording Rules
Used for secondary development of metric data. Set these rules to pre-calculate data and save on query resources.

Credential
No access code available for this project.
Access code is the identity credential for calling an API. With an access code, you can report native Prometheus m

Add Access Code
An access code will be automatically generated.

4. Obtain the values of *<project_id>*, *<aom_id>*, and *<aom_secret>*.



Configuration Code for Prometheus Remote Read

```
remote_read:
  - url: 'https://a[redacted]l.com/v1/0[redacted]3/04[redacted]3/api/v1/read'
    tls_config:
      insecure_skip_verify: true
    bearer_token: 'd4[redacted]I'
    read_recent: true
```

Configuration Code for Prometheus Remote Write

```
remote_write:
  - url: 'h[redacted].com:8443/v1/0[redacted]3/0[redacted]l3/push'
    tls_config:
      insecure_skip_verify: true
    bearer_token: 'd[redacted]H'
```

5. Combine the obtained values into a character string in the format of *<project_id>_<aom_id>: <aom_secret>*.

```
a81[redacted]24_879f[redacted]1: PRu[redacted]Iwg
```

CAUTION

There must be a space after the colon (:).

6. Encode the character string using Base64.

   YTgxZ[redacted]XU3SVdn

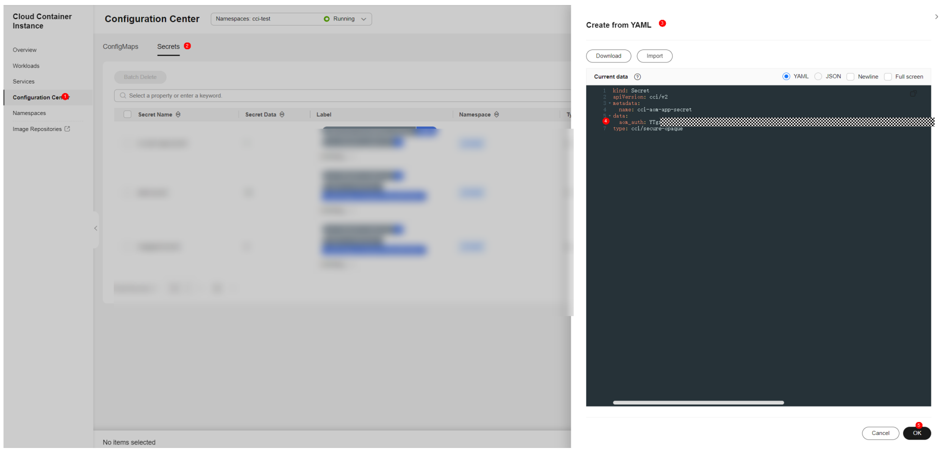
   You can run the following **shell** commands for encoding:

```
project_id=<project_id>
aom_id=<aom_id>
aom_secret=<aom_secret>
echo -n "${project_id}_${aom_id}: ${aom_secret}" |base64 -w 0
```

7. Fill the Base64-encoded character string in the following template to replace **{AOMAuthBase64}**, and then copy the code.

```
kind: Secret
apiVersion: cci/v2
metadata:
  name: cci-aom-app-secret
data:
  aom_auth: {AOMAuthBase64}
type: cci/secure-opaque
```

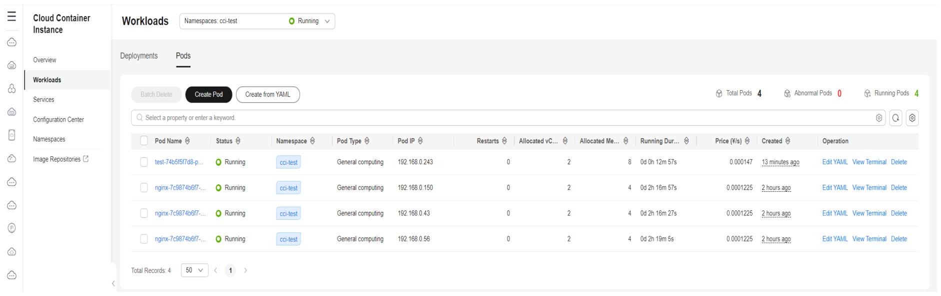
8. Log in to the **CCI 2.0 console**.
9. In the navigation pane, choose **Configuration Center**.
10. Click the **Secrets** tab.
11. Click **Create from YAML** and use the copied code to replace the code on the CCI 2.0 console to create an AOM app secret.



```
kind: Secret
apiVersion: cci/v2
metadata:
  name: cci-aom-app-secret
data:
  aom_auth: YTg[redacted]
type: cci/secure-opaque
```

Step 3 Create a Pod

1. Log in to the **CCI 2.0 console**.
2. In the navigation pane, choose **Workloads**. On the displayed page, click the **Pods** tab.
3. Click **Create Pod** and configure the parameters. For details about the parameters, see **Creating a Pod**.



Pod Name	Status	Namespace	Pod Type	Pod IP	Restarts	Allocated vC...	Allocated Me...	Running Dur...	Price (¥/s)	Created	Operation
test-74b5f5f7d8-p...	Running	cci-test	General computing	192.168.0.243	0	2	8	0d 0h 12m 57s	0.000147	13 minutes ago	Edit YAML View Terminal Delete
nginx-7c9874b6f7-...	Running	cci-test	General computing	192.168.0.150	0	2	4	0d 2h 16m 57s	0.0001225	2 hours ago	Edit YAML View Terminal Delete
nginx-7c9874b6f7-...	Running	cci-test	General computing	192.168.0.43	0	2	4	0d 2h 16m 27s	0.0001225	2 hours ago	Edit YAML View Terminal Delete
nginx-7c9874b6f7-...	Running	cci-test	General computing	192.168.0.56	0	2	4	0d 2h 19m 5s	0.0001225	2 hours ago	Edit YAML View Terminal Delete

Total Records: 4 50 1

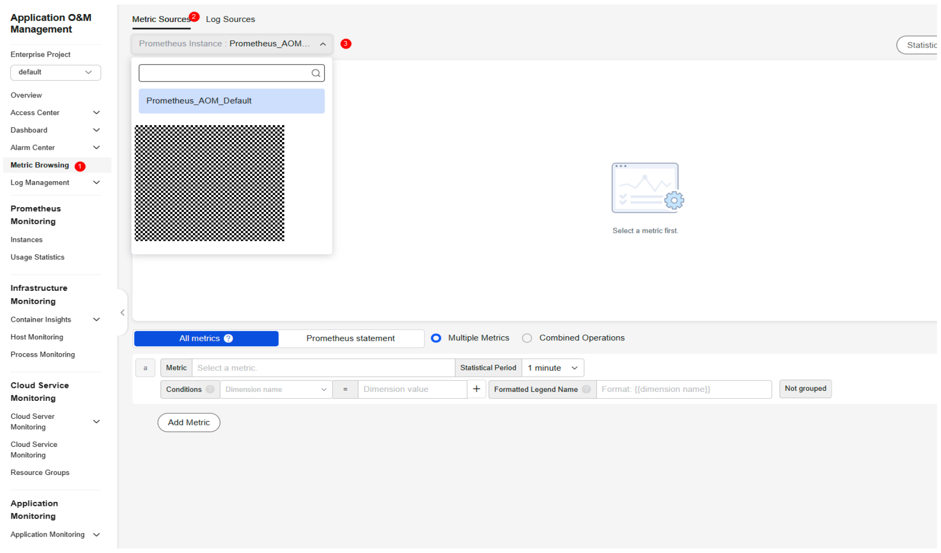
4. Click **Create Now**.

Step 4 View Monitoring Data

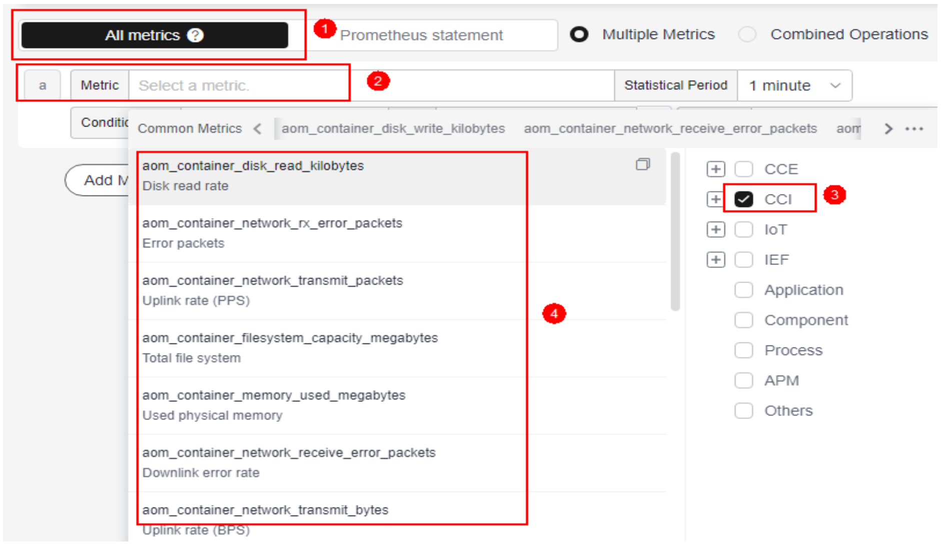
There are two ways to view monitoring data of a pod: All metrics and Prometheus statement.

Method 1: All Metrics

1. Log in to the AOM 2.0 console.
2. In the navigation pane, choose **Metric Analysis** > **Metric Browsing**.
3. On the **Metric Sources** tab, select the Prometheus instance created in **Step 1: Create a Prometheus Instance**.



4. On the **All metrics** tab, click the text box of **Metric**.
5. Select **CCI** on the right and select related common metrics.



aom_container_disk_read_kilobytes
Disk read rate

aom_container_network_rx_error_packets
Error packets

aom_container_network_transmit_packets
Uplink rate (PPS)

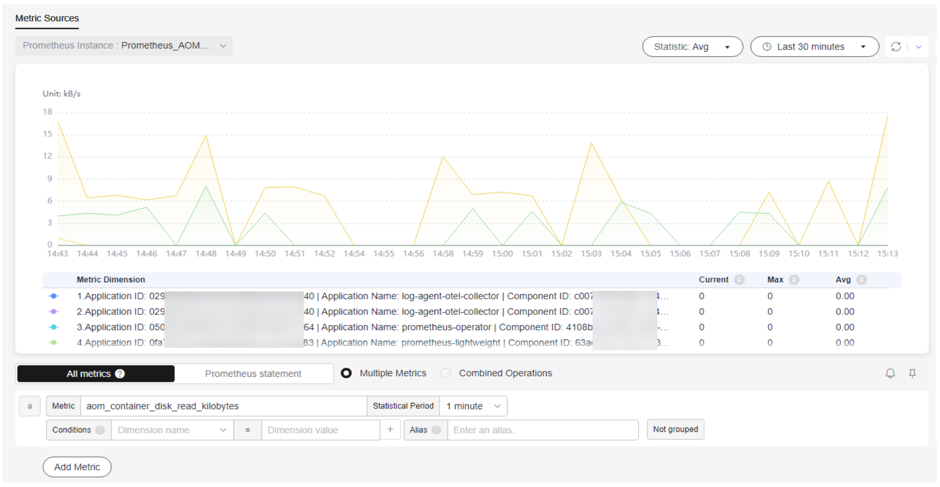
aom_container_filesystem_capacity_megabytes
Total file system

aom_container_memory_used_megabytes
Used physical memory

aom_container_network_receive_error_packets
Downlink error rate

aom_container_network_transmit_bytes
Uplink rate (BPS)

6. View metrics.

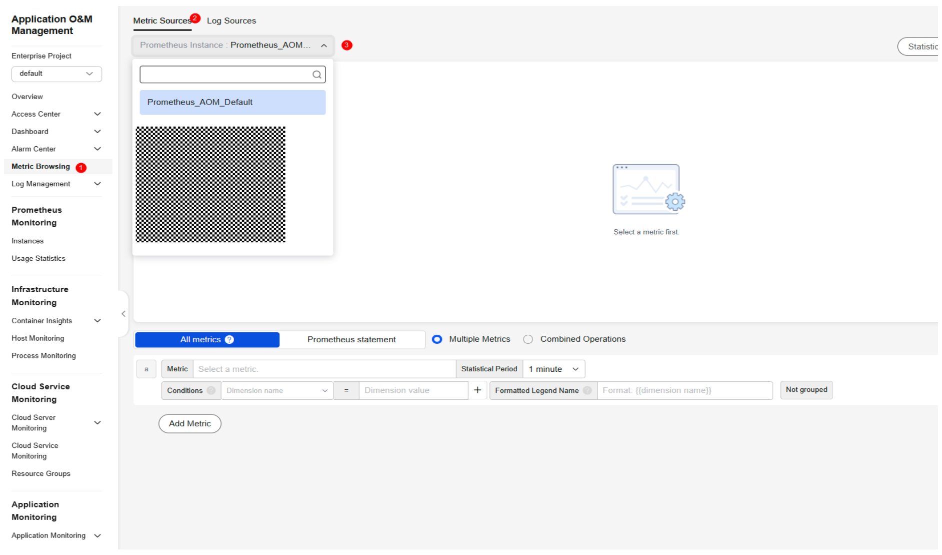


Metric Dimension	Current	Max	Avg
1.Application ID: 029[redacted]40 | Application Name: log-agent-otel-collector | Component ID: c007[redacted]4...	0	0	0.00
2.Application ID: 029[redacted]40 | Application Name: log-agent-otel-collector | Component ID: c007[redacted]4...	0	0	0.00
3.Application ID: 050[redacted]64 | Application Name: prometheus-operator | Component ID: 4108b[redacted]-...	0	0	0.00
4.Application ID: 0fa[redacted]83 | Application Name: prometheus-lightweight | Component ID: 63a[redacted]3...	0	0	0.00

a | Metric: aom_container_disk_read_kilobytes | Statistical Period: 1 minute

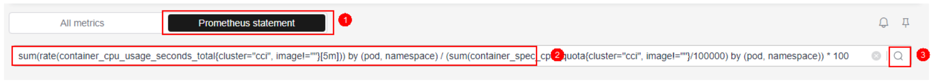
Method 2: Prometheus Statement

1. Log in to the AOM 2.0 console.
2. In the navigation pane, choose **Metric Analysis** > **Metric Browsing**.
3. On the **Metric Sources** tab, select the Prometheus instance created in **Step 1: Create a Prometheus Instance**.



[redacted]

4. On the **Prometheus statement** tab, enter a Prometheus statement in the search box and click .

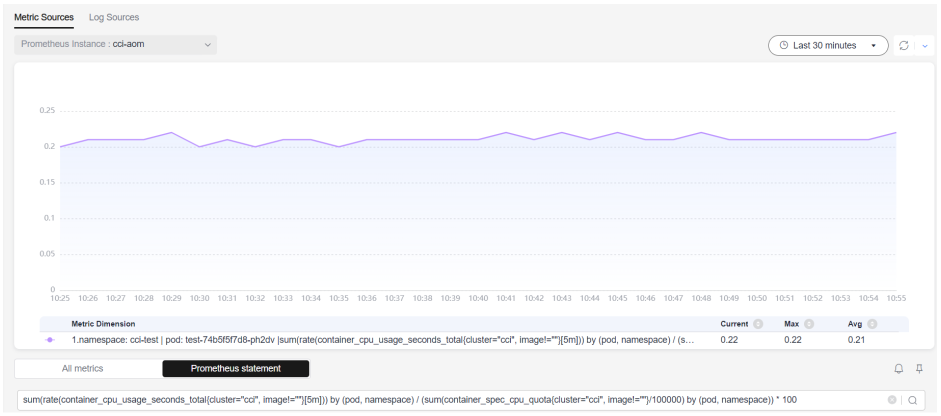


sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!=""}[5m])) by (pod, namespace) / (sum(container_spec_cpu_quota{cluster="cci", image!=""}/100000) by (pod, namespace)) * 100

The following are examples of common Prometheus statements.

a. Prometheus statement for querying the vCPU usages of all pods:

```
sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!=""}[5m])) by (pod,
namespace) / (sum(container_spec_cpu_quota{cluster="cci", image!=""}/100000) by (pod,
namespace)) * 100
```

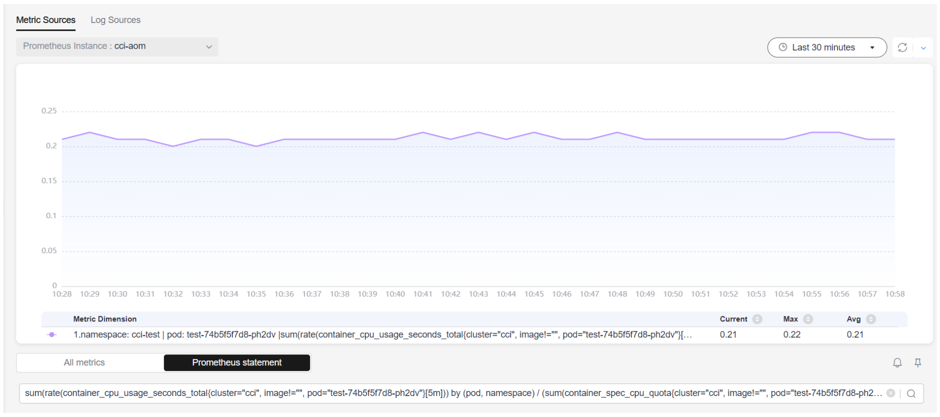


Metric Dimension	Current	Max	Avg
1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!=""}[5m])) by (pod, namespace) / (s...	0.22	0.22	0.21

sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!=""}[5m])) by (pod, namespace) / (sum(container_spec_cpu_quota{cluster="cci", image!=""}/100000) by (pod, namespace)) * 100

b. Prometheus statement for querying the vCPU usage of a pod (*<pod_name>* indicates the pod name):

```
sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="<pod_name>"}
[5m])) by (pod, namespace) / (sum(container_spec_cpu_quota{cluster="cci", image!="",
pod="<pod_name>"}/100000) by (pod, namespace)) * 100
```

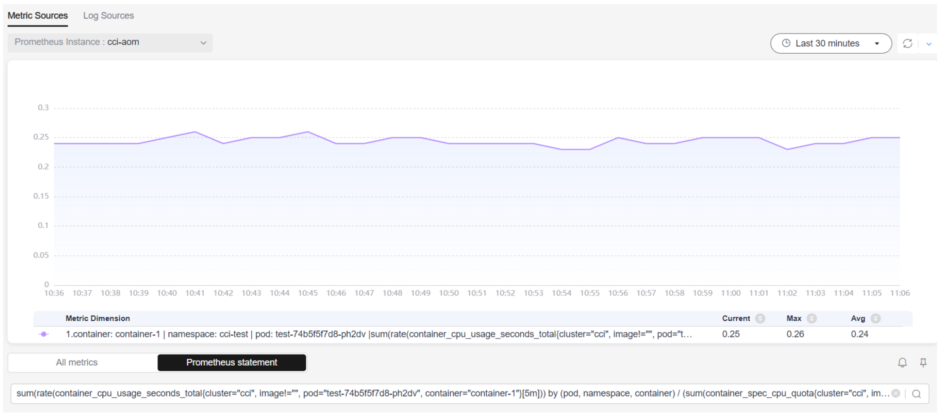


Metric Dimension	Current	Max	Avg
1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}[…	0.21	0.22	0.21

```
sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}[5m])) by (pod, namespace) / (sum(container_spec_cpu_quota{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2…
```

c. Prometheus statement for querying the vCPU usage of a container (*<pod_name>* indicates the pod name, and <container_name> indicates the container name):

```
sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="<pod_name>",
container="<container_name>"}[5m])) by (pod, namespace, container) /
(sum(container_spec_cpu_quota{cluster="cci", image!="", pod="<pod_name>",
container="<container_name>"}/100000) by (pod, namespace, container)) * 100
```

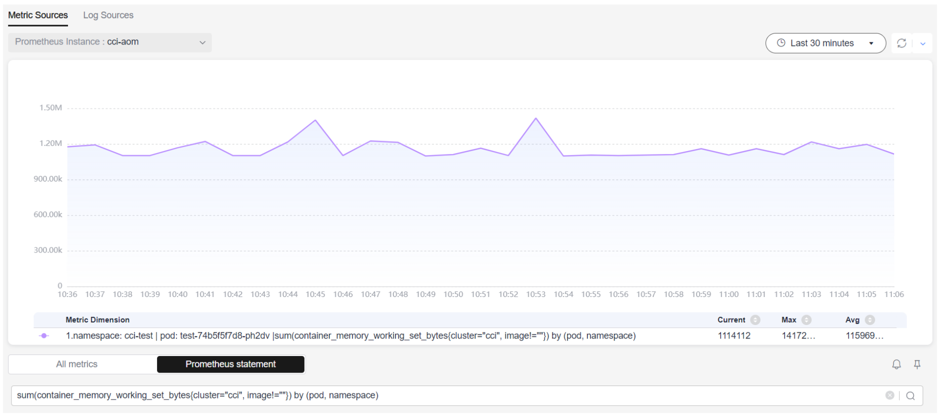


Metric Dimension	Current	Max	Avg
1.container: container-1 | namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="t…	0.25	0.26	0.24

```
sum(rate(container_cpu_usage_seconds_total{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv", container="container-1"}[5m])) by (pod, namespace, container) / (sum(container_spec_cpu_quota{cluster="cci", im…
```

d. Prometheus statement for querying the used memory of all pods:

```
sum(container_memory_working_set_bytes{cluster="cci", image!=""}) by (pod, namespace)
```

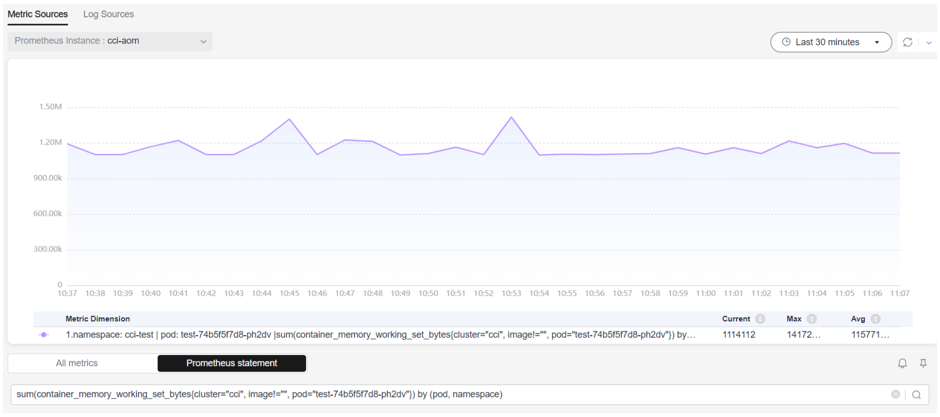


Metric Dimension	Current	Max	Avg
1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum(container_memory_working_set_bytes{cluster="cci", image!=""}) by (pod, namespace)	1114112	14172…	115969…

```
sum(container_memory_working_set_bytes{cluster="cci", image!=""}) by (pod, namespace)
```

e. Prometheus statement for querying the used memory of a pod (*<pod_name>* indicates the pod name):

```
sum(container_memory_working_set_bytes{cluster="cci", image!="", pod="<pod_name>"}) by
(pod, namespace)
```

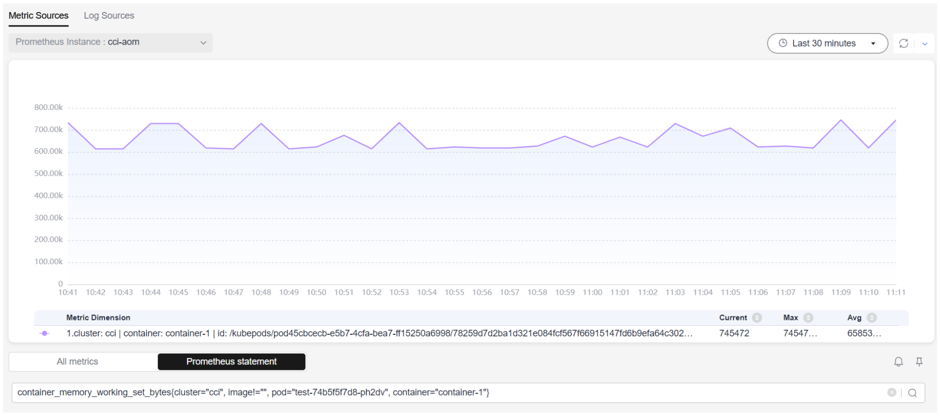


	Metric Dimension	Current	Max	Avg
●	1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum(container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) by…	1114112	14172…	115771…

```
sum(container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) by (pod, namespace)
```

f. Prometheus statement for querying the used memory of a container (*<pod_name>* indicates the pod name, and *<container_name>* indicates the container name):

```
container_memory_working_set_bytes{cluster="cci", image!="", pod="<pod_name>",
container="<container_name>"}
```

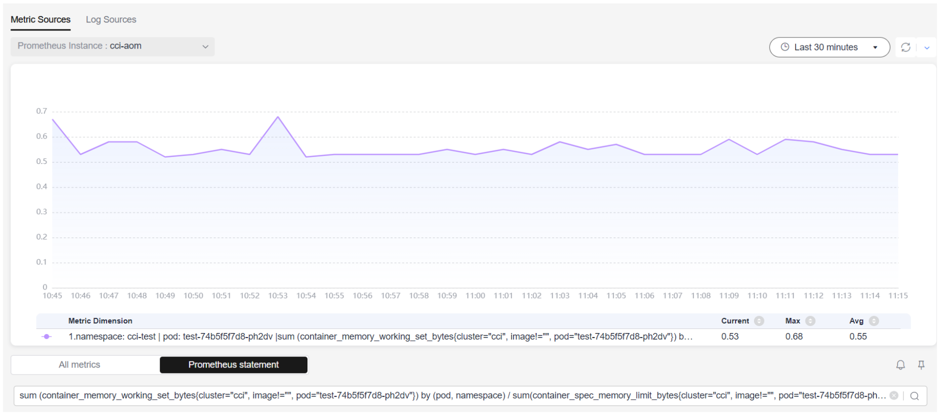


	Metric Dimension	Current	Max	Avg
●	1.cluster: cci | container: container-1 | id: /kubepods/pod45cbcecb-e5b7-4cfa-bea7-ff15250a6998/78259d7d2ba1d321e084fcf567f66915147fd6b9efa64c302…	745472	74547…	65853…

```
container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv", container="container-1"}
```

g. Prometheus statement for querying the memory usage of all pods:

```
sum (container_memory_working_set_bytes{cluster="cci", image!=""}) by (pod, namespace) /
sum(container_spec_memory_limit_bytes{cluster="cci", image!=""}) by (pod, namespace) * 100 !
=+Inf
```

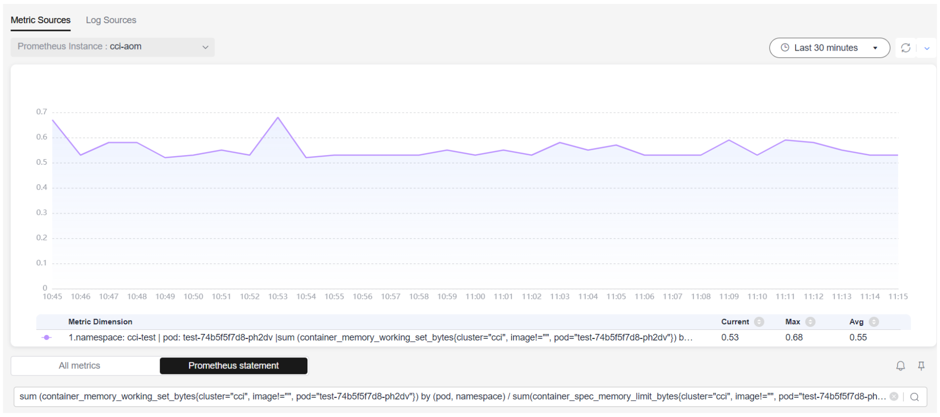


	Metric Dimension	Current	Max	Avg
●	1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum (container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) b…	0.53	0.68	0.55

```
sum (container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) by (pod, namespace) / sum(container_spec_memory_limit_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph…
```

h. Prometheus statement for querying the memory usage of a pod (*<pod_name>* indicates the pod name):

```
sum (container_memory_working_set_bytes{cluster="cci", image!="", pod="<pod_name>"}) by
(pod, namespace) / sum(container_spec_memory_limit_bytes{cluster="cci", image!="",
pod="<pod_name>"}) by (pod, namespace) * 100 !=+Inf
```

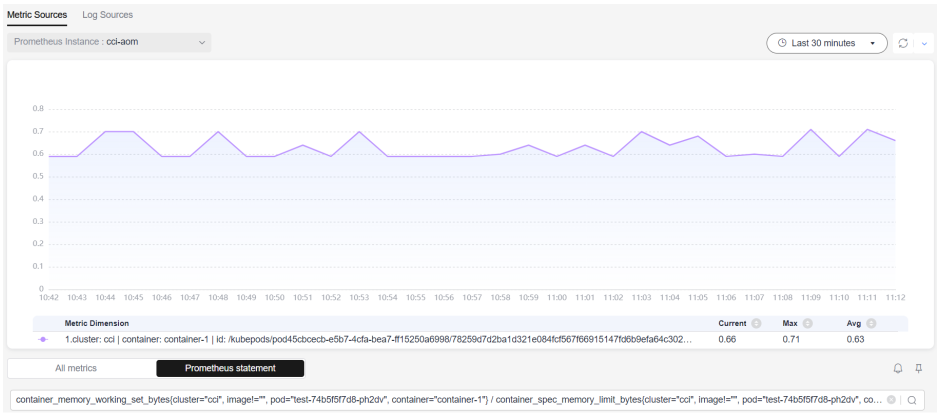


Metric Dimension	Current	Max	Avg
1.namespace: cci-test | pod: test-74b5f5f7d8-ph2dv |sum (container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) b…	0.53	0.68	0.55

```
sum (container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv"}) by (pod, namespace) / sum(container_spec_memory_limit_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph…
```

i. Prometheus statement for querying the memory usage of a container (*<pod_name>* indicates the pod name, and *<container_name>* indicates the container name):

```
container_memory_working_set_bytes{cluster="cci", image!="", pod="<pod_name>", container="<container_name>"} / container_spec_memory_limit_bytes{cluster="cci", image!="", pod="<pod_name>", container="<container_name>"} * 100 !=+Inf
```



Metric Dimension	Current	Max	Avg
1.cluster: cci | container: container-1 | id: /kubepods/pod45cbcecb-e5b7-4cfa-bea7-ff15250a6998/78259d7d2ba1d321e084fcf567f66915147fd6b9efa64c302…	0.66	0.71	0.63

```
container_memory_working_set_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv", container="container-1"} / container_spec_memory_limit_bytes{cluster="cci", image!="", pod="test-74b5f5f7d8-ph2dv", co…
```

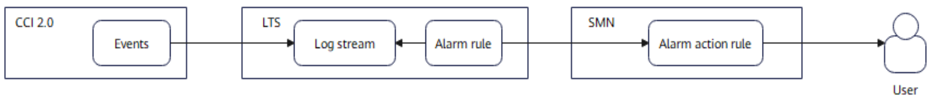
For more monitoring and O&M methods, see **Creating a Dashboard**.

4.8 Event Management

4.8.1 Enabling Event Reporting

CCI 2.0 uses LTS to collect events and allows you to configure alarms to detect workload exceptions in a timely manner.

When you are using CCI 2.0, events will be generated for various resources, such as Deployments, Services, HPA policies, networks, and pods.



Constraints

- CCI 2.0 supports event reporting only in the AP-Singapore and ME-Riyadh regions.

Event Format

The events comply with the **CloudEvents** format standard. The event content is stored using JSON. The following provides an example.

```
{
    "data": {
        "metadata": {
            "name": "service-test-event",
            "namespace": "test",
            "uid": "1******3-2**1-4**a-9**8-2******0",
            "resourceVersion": "61788694",
            "creationTimestamp": "2024-11-30T07: 55: 40Z",
            "annotations": {
                "tenant.cci.io/tenant-id": "a81*******24"
            }
        },
        "involvedObject": {
            "kind": "Service",
            "namespace": "test",
            "name": "service-test",
            "uid": "f******3-d**b-4**e-a**c-9f******e3",
            "apiVersion": "cci/v2",
            "resourceVersion": "7669479"
        },
        "reason": "EnsuringLoadBalancer",
        "message": "Ensuring load balancer",
        "source": {
            "component": "service-controller"
        },
        "firstTimestamp": "2024-11-30T07: 52: 45Z",
        "lastTimestamp": "2024-12-03T02: 30: 02Z",
        "count": 4,
        "type": "Normal",
        "eventTime": null,
        "reportingComponent": "",
        "reportingInstance": ""
    },
    "deprecatedeventreason": "EnsuringLoadBalancer",
    "eventclass": "Normal",
    "datacontenttype": "application/json",
    "time": "2024-04-05T17:31:00Z",
    "id": "b*****9-5737-4**1-8**a-3*******bc",
    "specversion": "1.0",
    "source": "cci:a8*********48b24:service:test:service-test:f******3-d**b-4**e-a**c-4******3",
    "type": "cloudservice.cci.service.publish.processing"
}
```

Table 4-24 Fields in the event

Field	Type	Description
type	String	Current operation status of the resource object, in the format of *cloudservice.cci.${Resource type}.${Resource operation}.${Operation result}.${Additional information}*. Additional information is only included in complex scenarios.
source	String	Resource object context, in the format of *cci:${projectid}:${Resource type}:${Namespace}:${Resource name}:${Unique resource ID}*.
specversion	String	CloudEvents version. The current version is 1.0.

Field	Type	Description
id	String	Unique ID of an event object, which is randomly generated.
time	Timestamp	Time when the event was generated.
eventclass	String	Event class, which can be **Normal** or **Warning**.
deprecatedeventreason	String	The reason that triggered the event. The Kubernetes **reason** field is reused and will be gradually discarded.
datacontenttype	String	Storage type of the **data** field, in the format of *application/json*.
data	k8s event object	Event details, which can be parsed based on the storage type defined by the **datacontenttype** field.

Enabling Event Reporting

You can enable event reporting on demand. This option is disabled by default, and you need to call the CCI 2.0 APIs to enable it.

NOTICE

- Before using the API, you need to create an agency for CCI for the first time.
- Log groups and log streams are free of charge. Each account is granted with a certain amount of storage each month. If the storage quota is exceeded, the excess will be billed. For details, see **Billing Details**. You can control log collection by enabling or disabling **Continue to Collect Logs When the Free Quota Is Exceeded** in the Configuration Center of LTS. For details, see **Setting LTS Log Collection Quota and Usage Alarms**. If this option is enabled, TLS continues to collect logs when the log size exceeds the free quota. You will be billed for extra logs on a pay-per-use basis. If you disable this option, log collection will be suspended when the log size exceeds the free quota. As a result, events reported by CCI will be invisible.

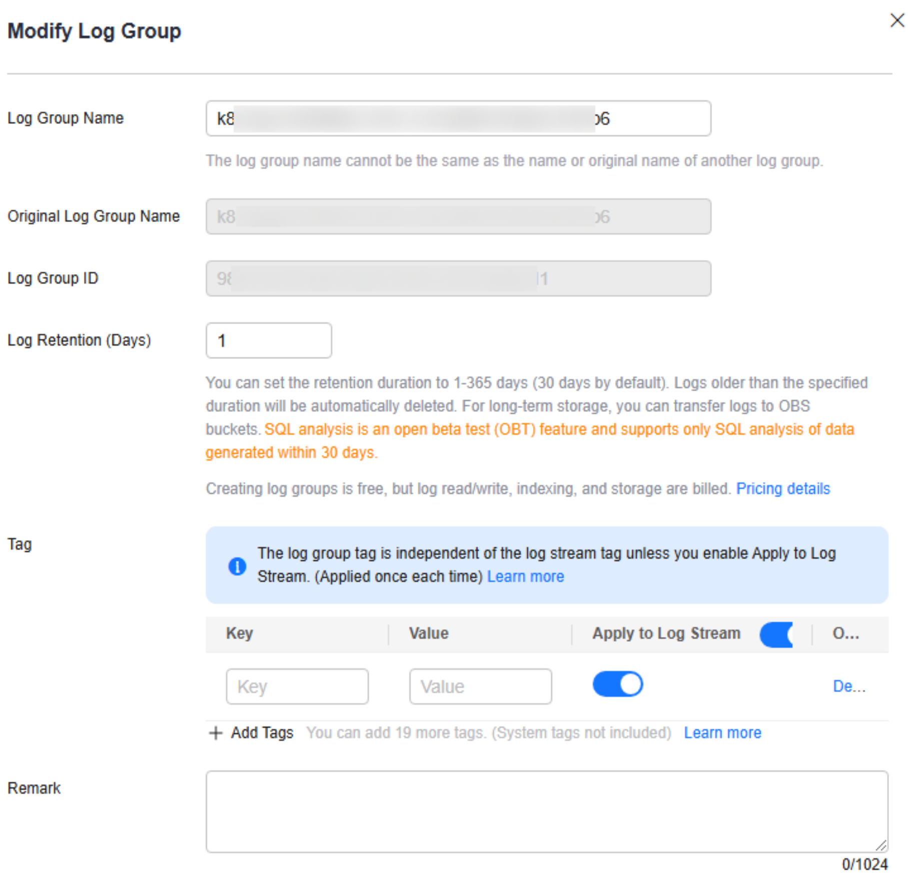
Table 4-25 API URIs

API URIs	Description
GET /v1/observabilityconfiguration	Queries the current observability configuration.
PUT /v1/observabilityconfiguration	Enables/Disables event reporting.

If event reporting is enabled, CCI 2.0 creates a log group and a log stream on LTS automatically. The log group is named in the format of *cci-event-${projectid}*, and the log stream in the format of *event-${projectid}*. Logs are retained for one day by default. Do not delete them.

To change the retention period, go to the LTS console, select the log group from the log group list, and click **Modify** in the **Operation** column.

Figure 4-14 Modifying a log group



Modify Log Group

Log Group Name: k8 ... 6
The log group name cannot be the same as the name or original name of another log group.

Original Log Group Name: k8 ... 6

Log Group ID: 9 ... 1

Log Retention (Days): 1
You can set the retention duration to 1-365 days (30 days by default). Logs older than the specified duration will be automatically deleted. For long-term storage, you can transfer logs to OBS buckets. SQL analysis is an open beta test (OBT) feature and supports only SQL analysis of data generated within 30 days.
Creating log groups is free, but log read/write, indexing, and storage are billed. Pricing details

Tag: The log group tag is independent of the log stream tag unless you enable Apply to Log Stream. (Applied once each time) Learn more

Key	Value	Apply to Log Stream	O...
Key	Value		De...

+ Add Tags You can add 19 more tags. (System tags not included) Learn more

Remark: 0/1024

You can click the log group name in the log group list to view resource events generated by CCI 2.0 in the log stream.

4.8.2 Configuring Alarm Rules on the LTS Console

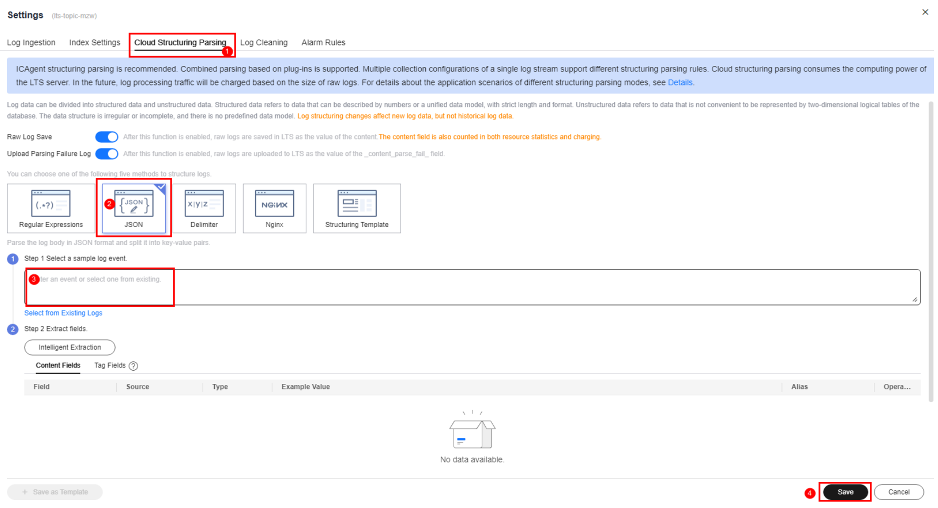
After event reporting is enabled, resource events will be reported to the log stream created by CCI 2.0. You can configure alarm rules on the LTS console. After events are generated and reported, the contacts configured on the SMN console will receive notifications.

Configuring Structured Parsing

1. Log in to the LTS console. In the log group list on the **Log Management** page, click the log group name to go to the log group details page.
2. Click ⚙ in the upper right corner.
3. On the **Cloud Structuring Parsing** tab, select **JSON**, and enter the event content in the **Select a sample log event** step.

4. Click **Intelligent Extraction** and then **Save**. After the configuration is successful, the reported events will be parsed based on the structured analysis rule.

Figure 4-15 Configuring structured parsing



Settings (lts-topic-mzw)

Log Ingestion | Index Settings | Cloud Structuring Parsing | Log Cleaning | Alarm Rules

ICAgent structuring parsing is recommended. Combined parsing based on plug-ins is supported. Multiple collection configurations of a single log stream support different structuring parsing rules. Cloud structuring parsing consumes the computing power of the LTS server. In the future, log processing traffic will be charged based on the size of raw logs. For details about the application scenarios of different structuring parsing modes, see Details.

Log data can be divided into structured data and unstructured data. Structured data refers to data that can be described by numbers or a unified data model, with strict length and format. Unstructured data refers to data that is not convenient to be represented by two-dimensional logical tables of the database. The data structure is irregular or incomplete, and there is no predefined data model. Log structuring changes affect new log data, but not historical log data.

Raw Log Save: After this function is enabled, raw logs are saved in LTS as the value of the content. The content field is also counted in both resource statistics and charging.

Upload Parsing Failure Log: After this function is enabled, raw logs are uploaded to LTS as the value of the _content_parse_fail_ field.

You can choose one of the following five methods to structure logs.

Parse the log body in JSON format and split it into key-value pairs.

Step 1 Select a sample log event.

ter an event or select one from existing.

Select from Existing Logs

Step 2 Extract fields.

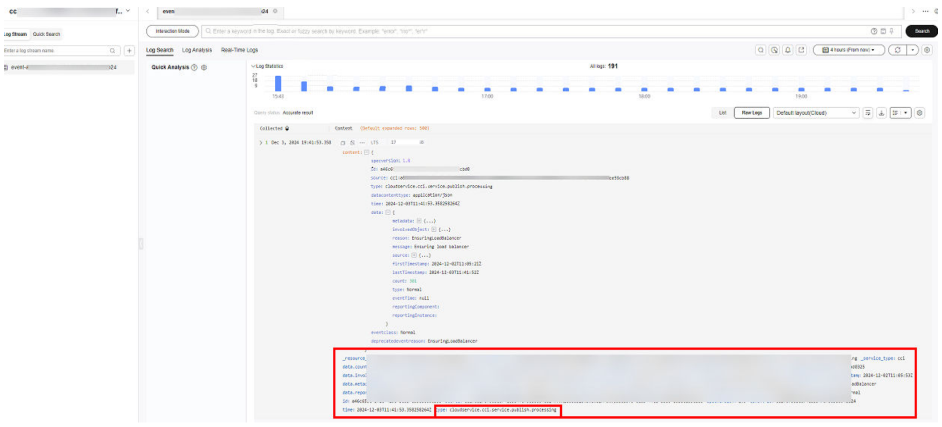
Field	Source	Type	Example Value	Alias	Opera...

No data available.

Creating an Alarm Rule

1. Log in to the LTS console. In the log group list on the **Log Management** page, click the log group name to go to the log group details page.
2. Configure filtering criteria for searing for the target events. You can also refer to **Configuring Log Alarm Rules**.

Figure 4-16 Configuring filtering criteria



Enter a keyword in the log. Exact or fuzzy search by keyword. Example: "error", "er?or", "er*r"

All logs: 191

Query status: Accurate result

```
Collected                     Content  (Default expanded rows: 500)
> 1 Dec 3, 2024 19:41:53.358      LTS   17 [redacted] 18
                              content: {
                                    specversion: 1.0
                                    id: a46c6[redacted]cbd8
                                    source: cci:a[redacted]ce59cb88
                                    type: cloudservice.cci.service.publish.processing
                                    datacontenttype: application/json
                                    time: 2024-12-03T11:41:53.358258264Z
                                    data: {
                                          metadata: {...}
                                          involvedObject: {...}
                                          reason: EnsuringLoadBalancer
                                          message: Ensuring load balancer
                                          source: {...}
                                          firstTimestamp: 2024-12-02T11:05:21Z
                                          lastTimestamp: 2024-12-03T11:41:52Z
                                          count: 301
                                          type: Normal
                                          eventTime: null
                                          reportingComponent:
                                          reportingInstance:
                                    }
                                    eventclass: Normal
                                    deprecatedeventreason: EnsuringLoadBalancer
                              }
                              _resource_[redacted]
                              data.count
                              data.invol
                              data.metad
                              data.repor
                              id: a46c65
                              time: 2024-12-03T11:41:53.358258264Z  type: cloudservice.cci.service.publish.processing
```

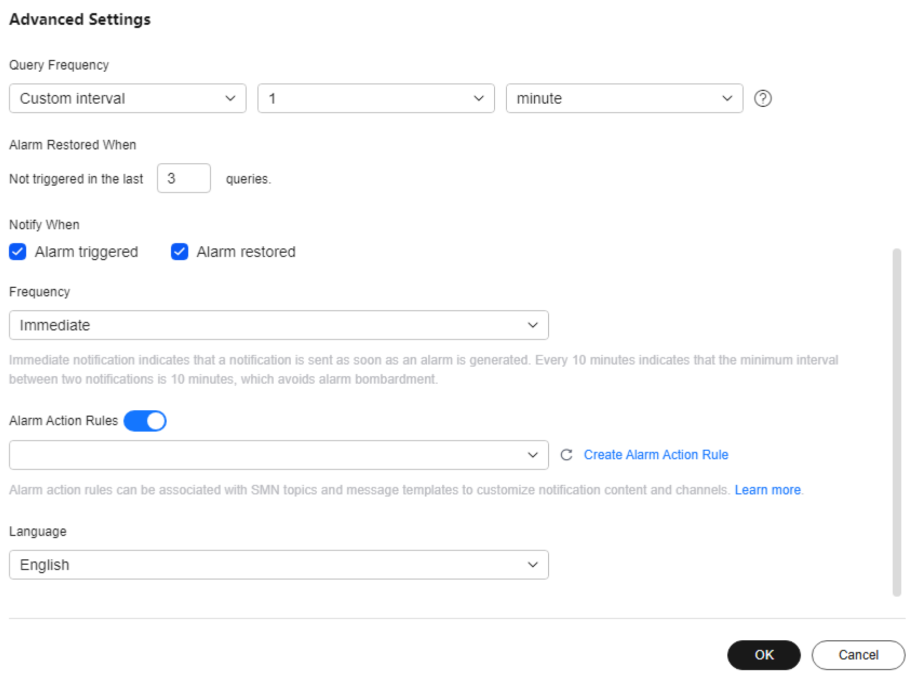
⚠ CAUTION

When searching for events based on the **type** field, you should use fuzzy matching so that you will not miss the events that need special attention.

3. Log in to the LTS console. On the **Log Alarms** page, click **Alarm Rules**.
4. Click **Create**.

5. Enter basic information, select the log group and log stream, and enter keywords.
6. Enable **Alarm Action Rules** and select an alarm action rule. If there are no alarm action rules, click **Create Alarm Action Rule** first.

Figure 4-17 Configuring advanced settings



Advanced Settings

Query Frequency: Custom interval | 1 | minute

Alarm Restored When: Not triggered in the last 3 queries.

Notify When: ☑ Alarm triggered ☑ Alarm restored

Frequency: Immediate

Immediate notification indicates that a notification is sent as soon as an alarm is generated. Every 10 minutes indicates that the minimum interval between two notifications is 10 minutes, which avoids alarm bombardment.

Alarm Action Rules: (enabled)

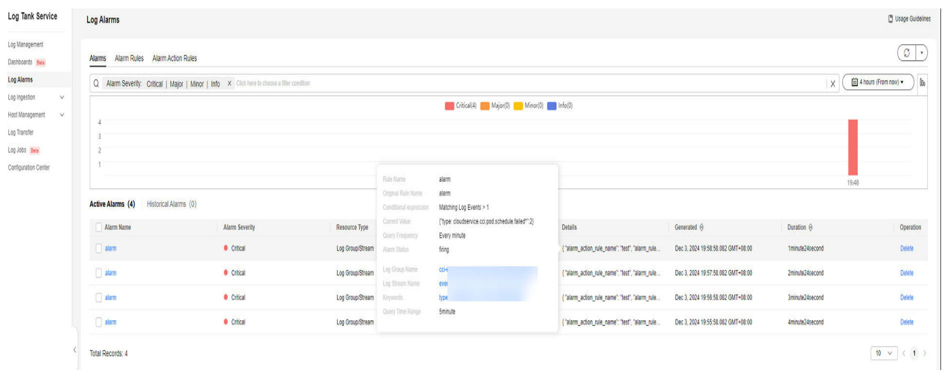
Create Alarm Action Rule

Alarm action rules can be associated with SMN topics and message templates to customize notification content and channels. Learn more.

Language: English

7. Click **OK**.
8. An alarm is triggered after a specific event is reported. You can view the alarm on the LTS console (**Log Alarms** > **Alarms**).

Figure 4-18 Alarm triggered after a specific event is reported



Alarm Severity: Critical | Major | Minor | Info

Rule Name	alarm
Original Rule Name	alarm
Conditional expression	Matching Log Events > 1
Current Value	{"type: cloudservice.cci.pod.schedule.failed"":2}
Query Frequency	Every minute
Alarm Status	firing
Log Group Name	cci-[redacted]
Log Stream Name	ever[redacted]
Keywords	type[redacted]
Query Time Range	5minute

Alarm Name	Alarm Severity	Resource Type	Details	Generated	Duration	Operation
alarm	Critical	Log Group/Stream	{"alarm_action_rule_name": "test", "alarm_rule...	Dec 3, 2024 19:58:58.082 GMT+08:00	1minute24second	Delete
alarm	Critical	Log Group/Stream	{"alarm_action_rule_name": "test", "alarm_rule...	Dec 3, 2024 19:57:58.082 GMT+08:00	2minute24second	Delete
alarm	Critical	Log Group/Stream	{"alarm_action_rule_name": "test", "alarm_rule...	Dec 3, 2024 19:56:58.082 GMT+08:00	3minute24second	Delete
alarm	Critical	Log Group/Stream	{"alarm_action_rule_name": "test", "alarm_rule...	Dec 3, 2024 19:55:58.082 GMT+08:00	4minute24second	Delete

Total Records: 4

4.8.3 Event List

CCI 2.0 generates events when operations are performed on multiple resources, such as Deployments, ReplicaSets, Services, HPA policies, networks, and pods. Events for each resource are classified by the **type** field. Each resource corresponds to several fixed operation types and results. The additional information is only included in some complex scenarios.

Table 4-26 Event list

Resource Object	Operation Type	Operation Result	Additional Information	Details	Example Kubernetes-like Event Reason
Deployment	scale	succeeded	-	The Deployment is scaled.	ScalingReplicaSet
	scale	failed	-	Deployment scaling failed.	ReplicaSetCreateError
	delete	processing	-	Deleting the deployment...	DeleteDeployment
ReplicaSet	scale	succeeded	-	The ReplicaSet is scaled.	SuccessfulCreate and SuccessfulDelete
	scale	failed	-	ReplicaSet scaling failed.	FailedCreate and FailedDelete
Service	publish	failed	-	Failed to publish the Service.	SyncLoadBalancerFailed
	publish	succeeded	-	The Service is published.	EnsuredLoadBalancer
	publish	processing	-	The Service is being published.	EnsuringLoadBalancer
	delete	failed	-	Failed to delete the Service.	DeleteLoadBalancerFailed
	delete	succeeded	-	The Service is deleted.	DeletedLoadBalancer
	delete	processing	-	The Service is being deleted.	DeletingLoadBalancer
Horizontalpodautoscaler	rescale	failed	-	Rescaling failed.	FailedComputeMetricsReplicas and FailedRescale
	rescale	failed	computemetricsfailed	Rescaling failed due to metrics calculation failure.	FailedComputeMetricsReplicas
	rescale	succeeded	-	The Deployment is rescaled.	SuccessfulRescale

Resource Object	Operation Type	Operation Result	Additional Information	Details	Example Kubernetes-like Event Reason
	metricscrape	failed	-	Failed to obtain metrics.	InvalidMetricSourceType and FailedGetResourceMetric
	metricscrape	failed	invalidmetricsourcetype	Failed to obtain metrics due to invalid metric types.	InvalidMetricSourceType
	scaleobjectparse	failed	invalidscaleobject	Failed to parse the scaling object due to invalid objects.	InvalidSelector and SelectorRequired
	scaleobjectparse	failed	ambiguoushpa	Failed to parse the scaling object due to HPA matching disorder.	AmbiguousSelector
Network	networkprepare	failed	subneterror	Network preparation failed due to abnormal subnet synchronization.	FailedSyncNetwork
	networkprepare	processing	-	The network is being prepared.	ExternalDependenciesSyncd
	networkassign	failed	noavailableip	Failed to allocate the network because no IP address is available.	NetworkNoIPAvailable
	networkrelease	failed	-	Network release failed.	NetworkFailed
Pod	networkassign	failed	-	Network allocation failed.	FailedAssignENI and FailedAssignEIP
	networkrelease	failed	-	Network release failed.	FailedReleaseENI
	networkprobe	failed	-	Network detection failed.	warningNetworkNotReady

Resource Object	Operation Type	Operation Result	Additional Information	Details	Example Kubernetes-like Event Reason
	schedule	failed	-	Scheduling failed.	FailedScheduling
	schedule	succeeded	-	Scheduling succeeded.	Scheduled
	logsetup	failed	-	Failed to configure logs.	SetupLogFailed
	imagepull	failed	-	Failed to pull the image.	ImagePullSecretNotFound, FailedPullImage, and BackOffPullImage
	imagepull	failed	imagepullsecretnotfound	Failed to pull the image due to an imagepullsecret exception.	ImagePullSecretNotFound
	imagepull	processing	-	Image is being pulled.	Pulling
	imagepull	succeeded	-	Image is pulled.	Pulled
	containerlifecyclehook	failed	poststartfailed	PostStart failed.	FailedPostStartHook
	containerlifecyclehook	failed	prestopfailed	PreStop failed.	FailedPreStopHook
	containerlifecycleprobe	failed	-	Health check failed.	Unhealthy
	containerlifecycleprobe	succeeded	-	Health check succeeded.	Healthy
	podcreate	failed	-	Failed to create the pod.	FailedCreate, FailedStart, and BackOffStart
	podcreate	processing	-	The pod is being created.	SandboxChanged

Resource Object	Operation Type	Operation Result	Additional Information	Details	Example Kubernetes-like Event Reason
	podcreate	succeeded	-	The pod is created.	SuccessfulCreate and Started
	poddelete	failed	-	Failed to delete the pod.	FailedKillPod and FailedDelete
	poddelete	processing	-	The pod is deleted.	Killing
	metricsetup	succeed	-	The metric is set successfully.	DefaultMetricsPort
	volumemount	failed	-	Failed to mount the volume.	FailedAttachVolume and FailedMount
	volumemount	succeed	-	The volume has been mounted.	SuccessfulAttachVolume and SuccessfulMountVolume

CAUTION

Flow control is performed on event reporting by event source. If the event reporting speed is too high, flow control is triggered and events may be lost. By default, up to 25 events can be reported for each event source. One event is reported every 5 minutes.

4.9 Log Management

4.9.1 Log Collection

CCI 2.0 works with LTS to collect application logs and report these logs to LTS so that you can use them for troubleshooting.

Log Collection Reliability

The log system's main purpose is to record all stages of data for service components, including startup, initialization, exit, runtime details, and exceptions. It is primarily employed in O&M scenarios for tasks like checking component status and analyzing fault causes.

Standard streams (stdout and stderr) and local log files use non-persistent storage. However, data integrity may be compromised due to the following risks:

- Log rotation and compression potentially deleting old files
- Temporary storage volumes being cleared when Kubernetes pods end
- Automatic OS cleanup triggered by limited node storage space

While the Cloud Native Log Collection add-on employs techniques like multi-level buffering, priority queues, and resumable uploads to enhance log collection reliability, logs could still be lost in the following situations:

- The service log throughput surpasses the collector's processing capacity.
- The service pod is abruptly terminated and reclaimed by CCE.
- The log collector pod experiences exceptions.

The following lists some recommended best practices for cloud native log management. You can review and implement them thoughtfully.

- Use dedicated, high-reliable streams to record critical service data (for example, financial transactions) and store the data in persistent storage.
- Avoid storing sensitive information like customer details, payment credentials, and session tokens in logs.

Constraints

- Logs cannot be collected from the directory that a specified system, device, cgroup, or tmpfs is mounted to.
- A single-line log that exceeds 250 KB will not be collected.
- Regular expression match is only supported when a full path with a complete file name is specified.
- The name of each file to be logged must be unique in a container. If there are files with duplicate names, only the logs of one file are collected.
- After a pod is started, the log collection configuration cannot be updated. If the configuration is updated, the pod must be restarted for the configuration to take effect.
- A directory to be logged must exist before the container is started. If a directory is created after the container is started, the logs of the directory and of the files in that directory cannot be collected.
- If the name of a file exceeds 190 characters, its logs cannot be collected.
- Logs of init containers cannot be collected.
- Logs of Kunpeng pods cannot be collected.

Step 1 Create a Log Group

Step 1 Log in to the management console and choose **Management & Deployment** > **Log Tank Service**.

Step 2 Log in to the **LTS console**.

Step 3 On the **Log Management** page, click **Create Log Group**.

Step 4 On the displayed page, set log group parameters by referring to **Table 4-27**.

Table 4-27 Log group parameters

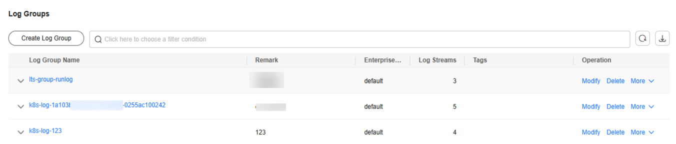
Parameter	Description
Log Group Name	A log group is the basic unit for LTS to manage logs. It is used to classify log streams. If there are too many logs to collect, separate logs into different log groups based on log types, and name log groups in an easily identifiable way. LTS automatically generates a default log group name. You are advised to customize one based on your service. You can also change it after the log group is created. The naming rules are as follows: ● Enter 1 to 64 characters, including only letters, digits, hyphens (-), underscores (_), and periods (.). Do not start with a period or underscore or end with a period. ● Each log group name must be unique.
Enterprise Project Name	Enterprise projects allow you to manage cloud resources and users by project. By default, the **default** enterprise project is used. You are advised to select an enterprise project that fits your service needs. To see all available options, click **View Enterprise Projects**. ● You can use enterprise projects only after enabling the enterprise project function. For details, see **Enabling the Enterprise Project Function**. ● You can remove resources from an enterprise project to another. For details, see **Removing Resources from an Enterprise Project**.

Parameter	Description
Log Retention (Days)	Specify the log retention period for the log group, that is, how many days the logs will be stored in LTS after being reported to LTS. By default, logs are retained for 30 days. You can set the retention period to one to 365 days. LTS periodically deletes logs based on the configured log retention period. For example, if you set the period to 30 days, LTS retains the reported logs for 30 days and then deletes them. **NOTE** The log retention period can be extended to 1,095 days. This feature is available only to whitelisted users. To enable it, **submit a service ticket**. ● By default, logs are retained for seven days. You can set the retention period to one to seven days. The logs that exceed the retention period will be automatically deleted. You can transfer logs to OBS buckets for long-term storage. **NOTE** The retention period for small and medium specifications is seven days by default. For large, ultra-large, and ultra-large II specifications, the retention period can be changed to seven to 365 days. To retain logs for seven to 365 days, you need to set the retention period when deploying LTS of the large, ultra-large, or ultra-large II specifications or when expanding capacity with log incremental packages. For details, see "AOM Installation Guide" > "Installing Cloud Services Using HCC Turnkey" in *Huawei Cloud Stack 8.6.0 Software Installation Guide for gPaaS & AI DaaS Services*. ● Raw logs reported to LTS will be deleted in the early morning of the next day after the log retention period expires.

Parameter	Description
Tag	Tag the log group as required. Click **Add** and enter a tag key and value. If you enable **Apply to Log Stream**, the tag will be applied to all log streams in the log group. To add more tags, repeat this step. A maximum of 20 tags can be added. **Tag key restrictions:** ● A tag key can contain letters, digits, spaces, and special characters (_.:=+-@), but cannot start or end with a space or start with **_sys_**. ● A tag key can contain up to 128 characters. ● Each tag key must be unique. **Tag value restrictions:** ● A tag value can contain letters, digits, spaces, and the following special characters: _.:=+-@ ● A tag value can contain up to 255 characters. **Tag policies:** If your organization has configured tag policies for LTS, follow the policies when adding tags to log groups, log streams, log ingestion configurations, host groups, and alarm rules. Non-compliant tags may cause the creation of these resources to fail. Contact your administrator to learn more about the tag policies. For details about tag policies, see **Overview of a Tag Policy**. For details about tag management, see **Managing Tags**. **Deleting a tag:** Click **Delete** in the **Operation** column of the tag. WARNING Deleted tags cannot be recovered. If a tag is used by a transfer task, you need to modify the task configuration after deleting the tag.
Remark	Enter remarks. The value contains up to 1,024 characters.

Step 5 Click **OK**. The created log group will be displayed in the log group list.

Figure 4-19 Log group



Log Groups

Create Log Group | Click here to choose a filter condition

Log Group Name	Remark	Enterprise...	Log Streams	Tags	Operation
lts-group-runlog	[redacted]	default	3		Modify Delete More
k8s-log-1a103[redacted]-0255ac100242	[redacted]	default	5		Modify Delete More
k8s-log-123	123	default	4		Modify Delete More

- In the log group list, view information such as the log group name, tags, and log streams.
- Click the log group name to access the log details page.

----End

Step 2 Create a Log Stream

1. On the LTS console, click ⌄ on the left of a log group name.
2. In the upper left corner of the displayed page, click **Create Log Stream** and then enter a log stream name. The log stream name:
   - Can contain only letters, numbers, underscores (_), hyphens (-), and periods (.). The name cannot start with a period or underscore, or end with a period.
   - Can contain 1 to 64 characters.

   NOTE

   Collected logs are sent to the created log stream. If there are a large number of logs, you can create multiple log streams and name them for quick log search.
3. Select an enterprise project. You can click **View Enterprise Projects** to view all enterprise projects.
4. If you enable **Log Retention Duration** on this page, you can set the log retention duration specifically for the log stream. If you disable it, the log stream will inherit the log retention setting of the log group.
5. Anonymous write is disabled by default and is suitable for log reporting on Android, iOS, applets, and browsers. If anonymous write is enabled, the anonymous write permission is granted to the log stream, and no valid authentication is performed, which may generate dirty data.
6. Set the tag in the *Key*=*Value* format, for example, a=b.
7. Enter remarks. A maximum of 1,024 characters are allowed.
8. Click **OK**. In the log stream list, you can view information such as the log stream name and operations.

   NOTE

   - You can view the log stream billing information. For details, see **Price Calculator**.
   - The function of reporting SDRs by log stream is in Friendly User Test (FUT). You can **submit a service ticket** to enable it.

Step 3 Obtain the Log Group ID and Log Stream ID

1. In the navigation pane, choose **Log Management**.
2. Select the log group created in **Step 1 Create a Log Group** and click **More** > **Details** in the **Operation** column.
3. Copy the log group ID.
4. Click ⌄ on the left of the log group name.
5. Select the log stream created in **Step 2 Create a Log Stream** and click **Details** in the **Operation** column.
6. Copy the log stream ID.

Step 4 Create a Deployment on the CCI Console

1. Log in to the **CCI 2.0 console**.

1. In the navigation pane, choose **Workloads**. On the displayed page, click the **Deployments** tab.
2. Click **Create from YAML** to create a Deployment using **YAML** or **JSON**.
   – Method 1: Use YAML to create a Deployment.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  annotations:
    description: ''
  labels: {}
  name: test
  namespace: default  # Namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      annotations:
        logconf.k8s.io/fluent-bit-log-type: lts       # (Mandatory) LTS is used to collect logs.
        logconfigs.logging.openvessel.io: |
          {
            "default-config": {# You can set the log collection paths of multiple containers. stdout.log 
indicates standard output. /root/out.log contains the text logs in rootfs (volumes included). /data/
emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
              "container_files": {
                "container-0": "stdout.log;/root/out.log;/data/emptydir-volume/*.log",
                "container-1": "/root/standard.log"
              },
              "regulation": "", #Regular expression for matching multi-line logs. For details, see Regular 
Expression Matching Rules.
              "lts-log-info": {     # Configure a log group and a log stream.
                "<log-group-ID>": "<log-stream-ID>"  #Replace <log-group-ID> and <log-stream-ID> with 
those obtained in Step 3.
              }
            },
            "multi-config": {
              "container_files": {
                "container-0": "/root/multi.log",
                "container-1": "stdout.log;/root/out.log;/data/emptydir-memory-volume/*.log"
              },
              "regulation": "/(?<log>\\d+-\\d+-\\d+ \\d+:\\d+:\\d+.*)/",
              "lts-log-info": {     # Configure the same log group and log stream.
                "<log-group-ID>": "<log-stream-ID>"  #Replace <log-group-ID> and <log-stream-ID> with 
those obtained in Step 3.
              }
            }
          }
        resource.cci.io/pod-size-specs: 2.00_8.0
      labels:
        app: test
        sys_enterprise_project_id: "0"
    spec:
      containers:
      - image: swr.***.com/paas_cci/vk-webhook:8.16.0
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', "while true; do echo hello; touch /root/out.log; echo hello >> /root/out.log; 
touch /data/emptydir-volume/emptydir.log; echo hello >> /data/emptydir-volume/emptydir.log; sleep 
10; done"]
        lifecycle: {}
        volumeMounts:
        - name: emptydir-volume
          mountPath: /data/emptydir-volume
        - name: emptydir-memory-volume
          mountPath: /data/emptydir-memory-volume
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      - image: swr.***.com/paas_cci/vk-webhook:8.16.0
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', "while true; do echo hello; touch /root/out.log; echo hello >> /root/out.log; 
touch /data/emptydir-memory-volume/emptydir-memory.log; echo $(date +'%Y-%m-%d 
%H:%M:%S.%3N') hello >> /data/emptydir-memory-volume/emptydir-memory.log; echo hello >> /
data/emptydir-memory-volume/emptydir-memory.log; sleep 10; done"]
        lifecycle: {}
        volumeMounts:
        - name: emptydir-volume
          mountPath: /data/emptydir-volume
        - name: emptydir-memory-volume
          mountPath: /data/emptydir-memory-volume
        name: container-1
        resources:
          limits:
            cpu: 100m
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: Default
      volumes:
      - name: emptydir-volume
        emptyDir: {}
      - name: emptydir-memory-volume
        emptyDir:
          sizeLimit: 1Gi
          medium: Memory
      enableServiceLinks: false
      restartPolicy: Always
      schedulerName: volcano
      securityContext: {}
      terminationGracePeriodSeconds: 30
  minReadySeconds: 0
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 0
      maxUnavailable: 1
```

Table 4-28 Key parameters

Parameter	Mandatory	Type	Description
logconf.k8s.io/fluent-bit-log-type	Yes	String	• Description: Log collection mode • Constraints: This parameter is mandatory. • Value: **lts**

Parameter	Mandatory	Type	Description
logconfigs.logging.openvessel.io	Yes	String	• Description: Log collection configuration

NOTE

When using YAML, delete the comments of the parameters in the **logconfigs.logging.openvessel.io** field.

– Method 2: Use JSON to create a Deployment.

```
{
    "default-config": {
        "container_files": { // You can set the log collection paths of multiple containers. stdout.log indicates standard output. /root/out.log indicates the text logs in rootfs (volumes included). /data/emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
            "container-0": "stdout.log;/root/out.log;/data/emptydir-volume/*.log",
            "container-1": "stdout.log"
        },
        "regulation": "", // Regular expression matching rule for collecting multi-line logs. For details about regular expression matching rules, see https://docs.fluentbit.io/manual/pipeline/parsers/configuring-parser.
        "lts-log-info": { // Only one log group and one log stream are allowed.
            "log-group-ID": "log-stream-ID"  // Replace log-group-ID and log-stream-ID with those obtained in Step 3.
        }
    },
    "multi-config": {
        "container_files": { // You can set the log collection paths of multiple containers. stdout.log indicates standard output. /root/out.log indicates the text logs in rootfs (volumes included). /data/emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
            "container-0": "stdout.log;/root/out.log;/data/emptydir-memory-volume/*.log"
        },
        "regulation": "/(?<log>\\d+-\\d+-\\d+ \\d+:\\d+:\\d+.*)/",  // Regular expression matching rule for collecting multi-line logs
        "lts-log-info": { // Only one log group and one log stream are allowed.
            "<log-group-ID>": "<log-stream-ID>"  // Replace <log-group-ID> and <log-stream-ID> with those obtained in Step 3.
        }
    }
}
```

CAUTION

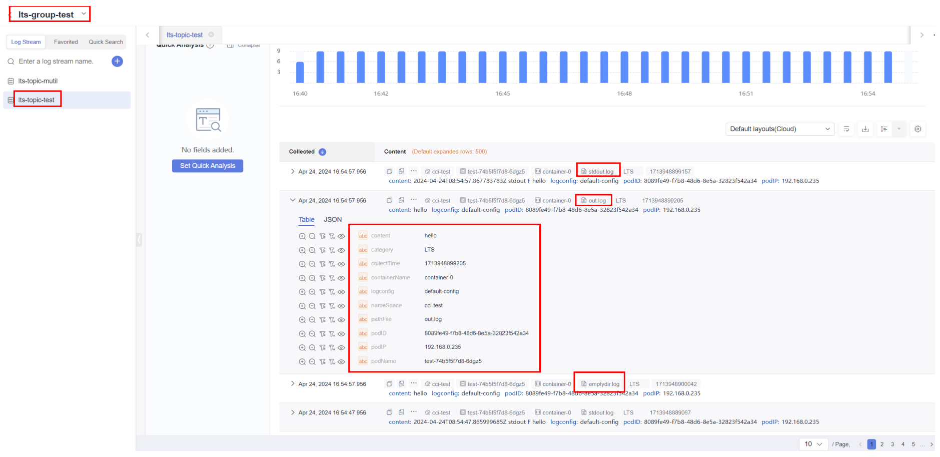
When you are creating a workload using YAML or JSON, you need to add the log group ID and log stream ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID** to the **lts-log-info** parameter, for example, **\"lts-log-info\":**{|"*log-group-ID*|":|"*log-stream-ID*|}

- Replace *log-group-ID* with the log group ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID**.
- Replace *log-stream-ID* with the log stream ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID**.

3. Click **OK**.
4. Click the created Deployment to view its status.

Step 5 Check Log Reporting

1. Log in to the LTS console. In the navigation pane, choose **Log Management** and click the name of the log group.
2. Go to the log details page to view logs.



```
Collected    Content (Default expanded rows: 500)
> Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  stdout.log  LTS  1713948899157
  content: 2024-04-24T08:54:57.867783783Z stdout F hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
v Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  out.log  LTS  1713948899205
  content: hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
    content        hello
    category       LTS
    collectTime    1713948899205
    containerName  container-0
    logconfig      default-config
    nameSpace      cci-test
    pathFile       out.log
    podID          8089fe49-f7b8-48d6-8e5a-32823f542a34
    podIP          192.168.0.235
    podName        test-74b5f5f7d8-6dgz5
> Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  emptydir.log  LTS  1713948900042
  content: hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
> Apr 24, 2024 16:54:47.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  stdout.log  LTS  1713948889067
  content: 2024-04-24T08:54:47.865999685Z stdout F hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
10 / Page,  1 2 3 4 5 ...
```

4.9.2 Mounting Standard Output Logs

This section describes how to mount container standard output logs in a pod to a specified container for easy log retrieval.

Constraints

volume.cci.io/mount-stdlog-containers and **volume.cci.io/mount-stdlog-containers-path** cannot be configured at the same time.

Using a YAML File

This example shows how to configure this function during workload creation, with the key configuration marked in red:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: deploy-example
  namespace: namespace-example
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-example
  template:
    metadata:
      labels:
        app: deploy-example
      annotations:
        volume.cci.io/mount-stdlog-containers: sidecar  #Name of the container where standard output logs are mounted.
    spec:
      containers:
        - name: nginx
```

```
          image: nginx:latest
          resources:
            limits:
              cpu: '1'
              memory: 2Gi
            requests:
              cpu: '1'
              memory: 2Gi
        - name: sidecar
          image: sidecar:latest
          resources:
            limits:
              cpu: '0'
              memory: '0'
            requests:
              cpu: '0'
              memory: '0'
      dnsPolicy: Default
      imagePullSecrets:
        - name: imagepull-secret
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 100%
```

Table 4-29 Pod annotations

Annotation	Type	Description	Example Value
volume.cci.io/mount-stdlog-containers	String	1. Name of the container where standard output logs are mounted. To mount the standard output logs of multiple containers, separate their names with commas (,). You can also set the value to an asterisk (*) to mount the standard output logs of all containers. If the value is set to *****, it cannot be set to any other container name. 2. After a pod that matches the annotation starts, the directory that contains the standard output logs of all containers in the pod is mounted to the **/var/log/pods** directory of the container.	Example 1: `"container-0,container-1"` Example 2: `"*"`

Annotation	Type	Description	Example Value
volume.cci.io/mount-stdlog-containers-path	String	1. Container name and mount path that the standard output logs are mounted. The value is in JSON format. If the container name is set to *****, the logs of all containers are mounted to the specified container. If the value is set to *****, it cannot be set to any other container name. 2. After any container that matches the annotation starts, the directory that contains the standard output logs of all containers in the pod is mounted to the specified path of the container.	Example 1: `"{\"container-0\":\"/var/log/pods\",\"container-1\":\"/tmp/log/pods\"}"` Example 2: `"{\"*\":\"/tmp/log/pods\"}"`

5 Using CCI with CCE

5.1 Overview

This section describes the functions, resource usages, and custom annotations of the CCE Cloud Bursting Engine for CCI add-on.

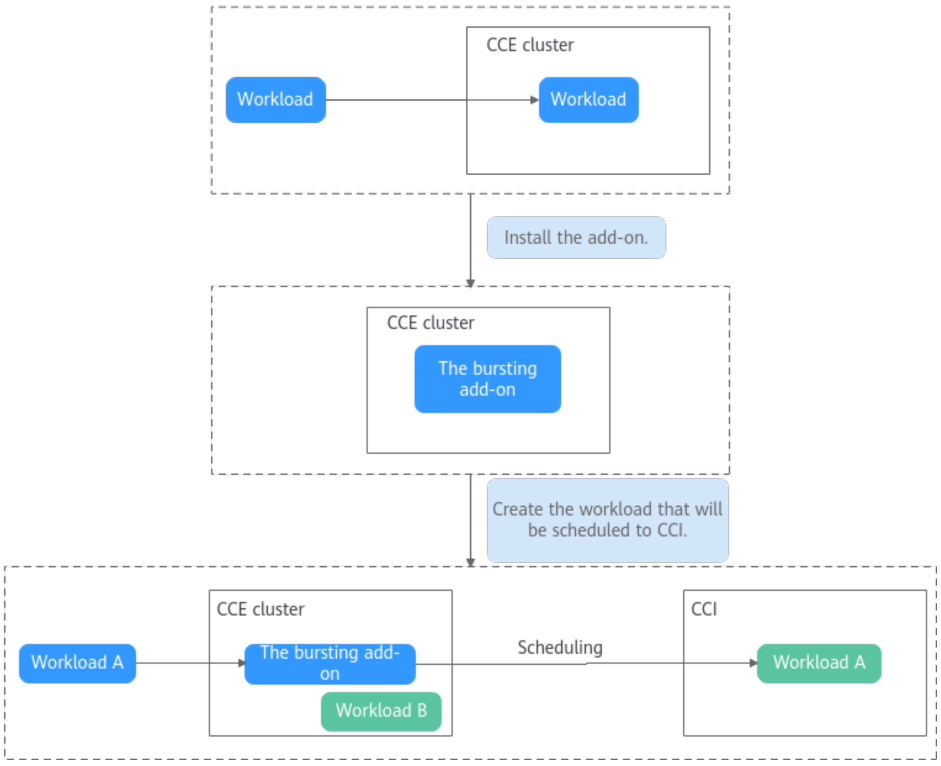
The CCE Cloud Bursting Engine for CCI add-on functions as a virtual kubelet to connect Kubernetes clusters to APIs of other platforms. This add-on is used to extend Kubernetes APIs to CCI 2.0.

With this add-on, you can schedule Deployments, StatefulSets, jobs, and CronJobs in CCE clusters to CCI 2.0 during peak hours. In this way, you can reduce consumption caused by cluster scaling.

Functions

When using the CCE Cloud Bursting Engine for CCI add-on, pay attention to the workload delivery and scheduling.

Figure 5-1 How the CCE Cloud Bursting Engine for CCI add-on schedules extra workloads in a CCE cluster to CCI



CAUTION

Scheduling workloads to CCI 2.0 is closely related to your workload settings.

Configuration Item	Description	Flavors	Reference
Scheduling	You can manage pods in a CCE cluster in different ways to control their scheduling to CCI 2.0. Workloads can be scheduled to CCI 2.0 to improve resource utilization of clusters.	• There are three scheduling policies. • There are two methods for managing scheduling policies. • Multiple virtual nodes can be scheduled.	**Scheduling Workloads to CCI 2.0**
Resource quotas	You can configure fields such as **cpu** and **memory** to specify resource requests and limits for containers in a pod. The resource quotas for the add-on are rounded up to meet the requirements of CCI 2.0.	Appropriate pod resource quotas are selected.	**Rounding Up Pod Resources**

Configuration Item	Description	Flavors	Reference
Images	You can configure service images and run your service containers in Huawei Cloud CCE clusters and CCI 2.0.	• The image configuration mode can be changed. • Images can be upgraded in-place.	**Images**
Storage	You can mount storage volumes to workloads for persistent data storage.	• There are multiple storage volume types. • The hostPath volume of a workload can be replaced.	**Storage**
Networking	You can plan the network topology between CCE clusters and CCI 2.0.	• Pods in CCI 2.0 can be exposed by the Service. • Pods in a CCE cluster can communicate with pods in CCI 2.0 through the Service. • **cluster-dns** can be specified. • Global EIPs can be bound to pods.	**Networking**
Monitoring	You can install add-ons to connect to Monitoring Center for better O&M of pods for workloads scheduled to CCI 2.0.	• View data on the CCI 2.0 console.	**Monitoring**

Resource Usage Description

The following table describes the cloud services involved when the CCE Cloud Bursting Engine for CCI add-on is installed in a CCE cluster.

Involved Cloud Service	Resource Description	Remarks
CCI 2.0	A namespace called **bursting-***{CCE cluster ID}* will be created for the add-on in CCI 2.0.	• Do not use this namespace in CCI 2.0. If you need to use CCI 2.0, create a namespace. • CCI 2.0 namespaces are free for use.
CCE	Workloads, secrets, ConfigMaps, PVs, and PVCs in CCE are synchronized to CCI and occupy CCE node resources.	• If there are 1,000 pod and 1,000 ConfigMaps in your CCE cluster, you can apply for 2 cores and 4 GiB of memory. • If there are 2,000 pod and 2,000 ConfigMaps in your CCE cluster, you can apply for 4 cores and 8 GiB of memory. • If there are 4,000 pod and 4,000 ConfigMaps in your CCE cluster, you can apply for 8 cores and 16 GiB of memory.
ELB	If pods in a CCE cluster can communicate with the pods in CCI through the Service, the CCE Cloud Bursting Engine for CCI add-on automatically creates a shared load balancer.	• The shared load balancer is named **cce-lb-***xxx*. • The load balancer will be automatically deleted if the add-on is uninstalled or the network interconnection is disabled.
VPC	Workloads scheduled to CCI use the same VPC as the CCE cluster.	The CIDR block of the Service in the CCI namespace is 10.247.0.0/16. Do not configure the same CIDR block for the VPC subnet of the CCE cluster.
SWR	When you create a workload in CCE, the image can be pulled from SWR.	Ensure that your image has been pushed to SWR.

Pod Annotations

If the CCE Cloud Bursting Engine for CCI add-on is installed in a CCE cluster, custom annotations are required. The following table describes the annotations.

Annotation Key	Description	Reference
scheduling.cci.io/managed-by-profile	Profile that manages the pods of the workloads scheduled to CCI	**Scheduling Workloads to CCI 2.0**

Annotation Key	Description	Reference
resource.cci.io/size	vCPUs and memory of pods after the workloads are scheduled to CCI	**Rounding Up Pod Resources**
bursting.cci.io/image-replacement	Annotation for replacing image path prefixes	**Images**

Release History

Table 5-1 CCE Cloud Bursting Engine for CCI add-on

Add-on Version	Supported Cluster Version	New Feature
1.5.44	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31 v1.32	● CCE clusters v1.32 are supported. ● Supported automatic injection of sidecar containers. ● Supported ARM64 nodes.
1.5.29	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Specific subnets can be configured for pods.
1.5.28	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	Optimized some functions.

Add-on Version	Supported Cluster Version	New Feature
1.5.27	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30 v1.31	CCE clusters v1.31 are supported.
1.5.26	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Fixed some issues.
1.5.24	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Optimized some functions.
1.5.16	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29 v1.30	Compacted pod CPU and memory resources.
1.5.8	v1.21 v1.23 v1.25 v1.27 v1.28 v1.29	CCE clusters v1.29 are supported.

Add-on Version	Supported Cluster Version	New Feature
1.3.57	v1.21 v1.23 v1.25 v1.27 v1.28	CCE clusters v1.28 are supported.
1.3.48	v1.21 v1.23 v1.25 v1.27	● Clusters v1.25 and v1.27 are supported. ● Supported JuiceFS.
1.3.25	v1.17 v1.19 v1.21 v1.23	● Supported Downward API volumes. ● Supported Projected volumes. ● Supported custom storage classes.
1.3.19	v1.17 v1.19 v1.21 v1.23	Supported schedule profile.
1.3.7	v1.17 v1.19 v1.21 v1.23	Clusters v1.21 and v1.23 are supported.
1.2.12	v1.13 v1.15 v1.17 v1.19	● Added some metrics. ● Supported HPA and CustomHPA. ● Enabled the hostPath in the pod that is scaled to CCI to be converted to other types of storage. ● Fixed an issue that the Kubernetes Dashboard cannot run on terminals.

Add-on Version	Supported Cluster Version	New Feature
1.2.5	v1.13 v1.15 v1.17 v1.19	• Automatically cleared CCI resources that are no longer used by pods. • **Requests** and **Limits** can be set to different values. When pods are scaled to CCI, their resource requests are subject to the resource limits. • Fixed the issue that the add-on fails to be uninstalled when the CCI namespace does not exist. • Added the function of intercepting creation requests when the pod specifications exceed the CCI limit.
1.2.0	v1.13 v1.15 v1.17 v1.19	• Clusters v1.19 are supported. • Supported SFS and SFS Turbo storage. • Supported CronJobs. • Supported envFrom configuration. • Supported automatic logs dumping. • Shielded TCP socket health check. • Supported resource tags (pod-tag). • Improved performance and reliability. • Resolved some known issues.
1.0.5	v1.13 v1.15 v1.17	Clusters v1.17 are supported.

5.2 Quick Start

This section describes how you can use the CCE Cloud Bursting Engine for CCI add-on in a CCE cluster to quickly schedule pods to CCI 2.0.

Prerequisites

- Before using the add-on, go to the CCI console to grant CCI with the permissions to use CCE.
- VPC endpoints have been purchased for using the CCE Cloud Bursting Engine for CCI add-on. For details, see **Environment Configuration**.

Constraints

- Only CCE standard and CCE Turbo clusters that use the VPC network model are supported.
- The CCE Burst Elastic Engine (CCI) add-on earlier than v1.5.37 cannot be used in clusters of the Arm architecture. The pods for the add-on will not be scheduled on Arm nodes, if any, running in the cluster.
- The subnet where the cluster is located cannot overlap with 10.247.0.0/16, or the subnet conflicts with the Service CIDR block in the CCI namespace.
- Currently, Volcano Scheduler of 1.17.10 or earlier cannot be used to schedule pods with cloud storage volumes mounted to CCI.
- If the bursting add-on is used to schedule workloads to CCI 2.0, dedicated load balancers can be configured for ingresses and Services of the LoadBalancer type. The bursting add-on of a version earlier than 1.5.5 does not support Services of the LoadBalancer type.
- DaemonSets are not supported.
- Dynamic resource allocation is not supported, and related configurations are intercepted in plugin 1.5.27.
- After the add-on is installed, a namespace named **bursting-***{Cluster ID}* will be created in CCI and managed by the add-on. Do not use this namespace when manually creating pods in CCI.

Installing the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster **Overview** page.
3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Install**.
5. Select the add-on version. The latest version is recommended.
6. On the **Install Add-on** page, configure the specifications as needed.
   - If you select **Preset**, the system will configure the number of pods and resource quotas for the add-on based on the preset specifications. You can see the configurations on the console.
   - If you select **Custom**, you can adjust the number of pods and resource quotas as needed. High availability is not possible with a single pod. If there is an error on the node where the add-on pod runs, the add-on will not function.

NOTE

- The CCE Cloud Bursting Engine for CCI add-on of v1.5.2 or later uses more node resources. You need to reserve sufficient pods that can be created on a node before upgrading the add-on. For details about the number of pods that can be created on a node, see **Maximum Number of Pods That Can Be Created on a Node**.
- The resource usages of the add-on vary depending on the workloads scaled to CCI. The resource requests and limits of the proxy, resource-syncer, and bursting-resource-syncer components are related to the maximum number of pods that can be scaled out. The resource requests and limits of the virtual-kubelet, bursting-virtual-kubelet, profile-controller, webhook, and bursting-webhook components are related to the maximum number of pods that can be created or deleted concurrently. For details about the recommended formulas for calculating the resource requests and limits of each component, see **Table 5-2**. P indicates the maximum number of pods that can be scaled out, and C indicates the maximum number of pods that can be created or deleted concurrently. You are advised to evaluate your service volume and select appropriate specifications.

Table 5-2 Formulas for calculating resource requests and limits of each component

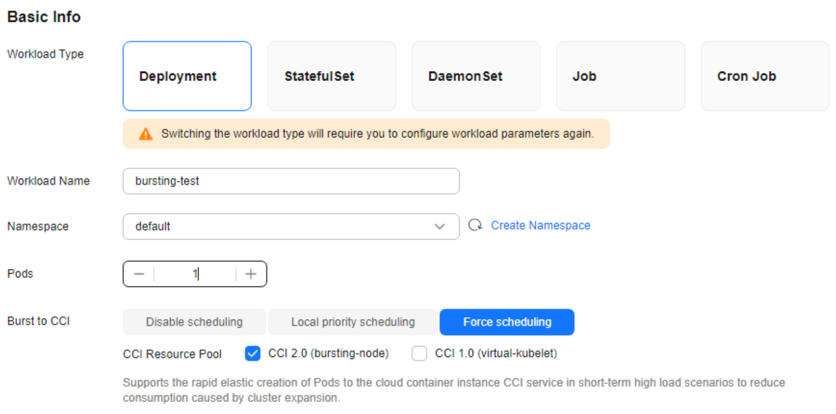
Component	CPU Request (m)	CPU Limit (m)	Memory Request (MiB)	Memory Limit (MiB)
virtual-kubelet/bursting-virtual-kubelet	(C + 400)/2,400 × 1,000	(C + 400)/600 × 1,000	(C + 400)/2,400 × 1,024	(C + 400)/300 × 1,024
profile-controller	(C + 1,000)/6,000 × 1,000	(C + 400)/1,200 × 1,000	(C + 1,000)/6,000 × 1,024	(C + 400)/1,200 × 1,024
proxy	(P + 2,000)/12,000 × 1,000	(P + 800)/2,400 × 1,000	(P + 2,000)/12,000 × 1,024	(P + 800)/2,400 × 1,024
resource-syncer/bursting-resource-syncer	(P + 800)/4,800 × 1,000	(P + 800)/1,200 × 1,000	(P + 800)/4,800 × 1,024	(P + 800)/600 × 1,024
webhook/bursting-webhook	(C + 400)/2,400 × 1,000	(C + 400)/600 × 1,000	(C + 1,000)/6,000 × 1,024	(C + 400)/1,200 × 1,024

– **(Optional) Networking**: If this option is enabled, pods in the CCE cluster can communicate with pods in CCI through Services. The Proxy component will be automatically deployed upon add-on installation. For details, see **Networking**.

7. Configure the add-on parameters.
   - **Subnet**: Pods running workloads scheduled to CCI use IP addresses in the selected subnet. Plan the CIDR block to ensure IP address provisioning is not impacted.
   - **Enterprise Project**: Select an enterprise project.
8. Click **Install**.

Creating a Workload

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Workloads**.
4. Click **Create Workload**. For details, see **Creating a Workload**.
5. Specify basic information. Select **Force scheduling** for **Burst to CCI** and then **CCI 2.0 (bursting node)** for **CCI Resource Pool**. For more information about scheduling policies, see **Scheduling Workloads to CCI**.


Basic Info

Workload Type: [✓ Deployment] StatefulSet DaemonSet Job Cron Job

Switching the workload type will require you to configure workload parameters again.

Workload Name: bursting-test

Namespace: default

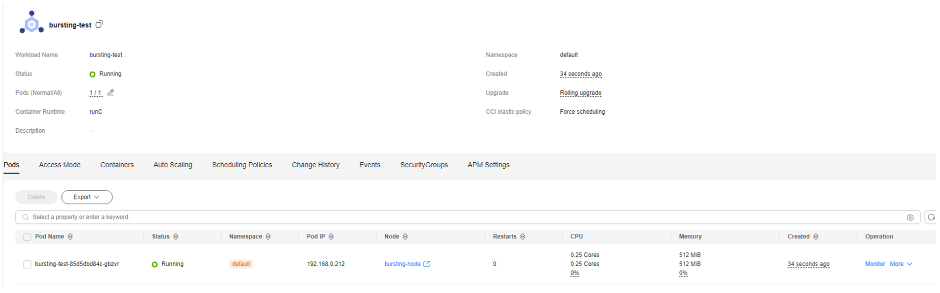
Pods: 1

Burst to CCI: Disable scheduling Local priority scheduling [✓ Force scheduling]

CCI Resource Pool: [✓ CCI 2.0 (bursting-node)] CCI 1.0 (virtual-kubelet)

Supports the rapid elastic creation of Pods to the cloud container instance CCI service in short-term high load scenarios to reduce consumption caused by cluster expansion.

6. Configure the container parameters.
7. Click **Create Workload**.
8. On the **Workloads** page, click the name of the created workload to go to the workload details page.
9. View the node where the workload is running. If the workload is running on bursting-node, it has been scheduled to CCI.


bursting-test

Workload Name	bursting-test	Namespace	default
Status	Running	Created	34 seconds ago
Pods (Normal/All)	1 / 1	Upgrade	Rolling upgrade
Container Runtime	runC	CCI elastic policy	Force scheduling
Description	--		

Pod Name	Status	Namespace	Pod IP	Node	Restarts	CPU	Memory	Created	Operation
bursting-test-85d5dbd84c-gbzvr	Running	default	192.168.0.212	bursting-node	0	0.25 Cores 0.25 Cores 0%	512 MiB 512 MiB 0%	34 seconds ago	Monitor More

Uninstalling the Add-on

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.

3. In the navigation pane, choose **Add-ons**.
4. Select the **CCE Cloud Bursting Engine for CCI** add-on and click **Uninstall**.

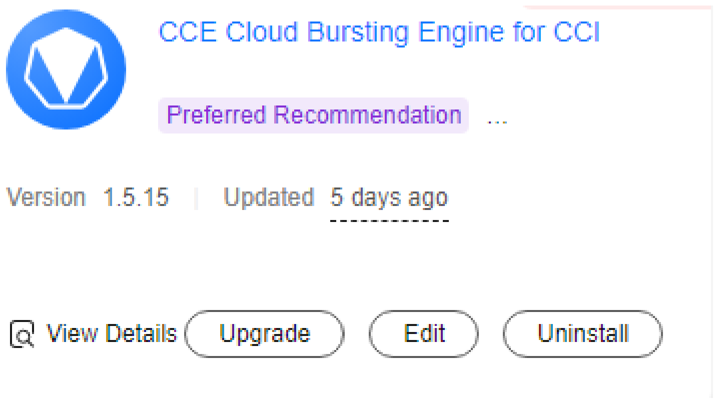

CCE Cloud Bursting Engine for CCI
Preferred Recommendation ...
Version 1.5.15 | Updated 5 days ago
View Details
Upgrade
Edit
Uninstall

Table 5-3 Special scenarios for uninstalling the add-on

Scenario	Symptom	Description
There are no nodes in the CCE cluster that the CCE Cloud Bursting Engine for CCI add-on needs to be uninstalled from.	Failed to uninstall the CCE Cloud Bursting Engine for CCI add-on.	If the CCE Cloud Bursting Engine for CCI add-on is uninstalled from the cluster, a job for clearing resources will be started in the cluster. To ensure that the job can be started, there is at least one node in the cluster that can be scheduled.
The CCE cluster is deleted, but the CCE Cloud Bursting Engine for CCI add-on is not uninstalled.	There are residual resources in the namespace on CCI. If the resources are not free, additional expenditures will be generated.	The cluster is deleted, but the resource clearing job is not executed. You can manually clear the namespace and residual resources.

For more information about the add-on, see **CCE Cloud Bursting Engine for CCI**.

5.3 Scheduling Workloads to CCI 2.0

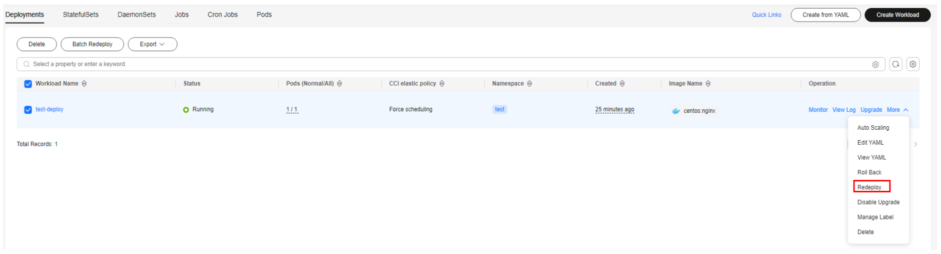
This section describes how you can schedule workloads to CCI 2.0 when needed.

You can use either of the following methods to schedule the workloads from a CCE cluster to CCI 2.0:

- **Method 1: Using labels to schedule workloads to CCI**
- **Method 2: Using a Profile**

Constraints

- You can only use ScheduleProfile to manage a workload and schedule it to CCI 2.0 when native labels of the workload are matched by ScheduleProfile. For example, the labels added to a workload through **ExtensionProfile** cannot be matched by ScheduleProfile. For this reason, the workload cannot be scheduled by ScheduleProfile.
- You can use labels to control pod scheduling to CCI 2.0. You need to add a label before a workload is created. If you add a label to an existing workload, the pods for the workload will not be updated. In this case, you can select the workload and choose **More** > **Redeploy** for update.



Workload Name	Status	Pods (Normal/All)	CCI elastic policy	Namespace	Created	Image Name	Operation
test-deploy	Running	1 / 1	Force scheduling	test	25 minutes ago	centos:nginx	Monitor View Log Upgrade More

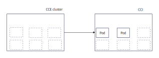
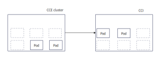
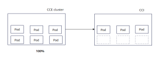
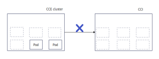
Total Records: 1

Prerequisites

- Add-on version: The CCE Cloud Bursting Engine for CCI add-on has been installed. If you use profiles to control pod scheduling to CCI, the add-on version must be 1.3.19 or later.
- Cluster type: If profiles are used to control pod scheduling to CCI, only CCE standard clusters and CCE Turbo clusters that use VPC networks are supported.

Scheduling Policies

There are four policies for elastically scheduling workloads from a CCE cluster to CCI 2.0.

Scheduling Policy	Policy Diagram	Application Scenario
Forcible scheduling (enforce)		Workloads are forcibly scheduled to CCI 2.0.
Automatic scheduling (auto)		Workloads are scheduled to CCI 2.0 based on the scoring results provided by the cluster scheduler.
Local priority scheduling (localPrefer)		Workloads are preferentially scheduled to a CCE cluster. If cluster resources are insufficient, Workloads are elastically scheduled to CCI 2.0.
Disable scheduling (off)		Workloads will not be scheduled to CCI 2.0.

Method 1: Using a Label

You can configure a label to control pod scheduling to CCI using either the console or YAML files.

Using the Console

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

Step 2 In the navigation pane, choose **Workloads**. On the displayed page, click **Create Workload**.

Step 3 In the **Basic Info** area, select any policy other than **Disable scheduling**.

- **Priority scheduling**: Pods will be preferentially scheduled to nodes in your CCE cluster. When the node resources are insufficient, pods will then be burst to CCI.
- **Force scheduling**: All pods will be scheduled to CCI.

Step 4 Select the required CCI resource pool.

- **CCI 2.0 (bursting-node)**: next-generation serverless resource pool
- **CCI 1.0 (virtual-kubelet)**: existing serverless resource pool, which will be unavailable soon.

NOTE

When creating a workload on the CCE console, you can select either **CCI 2.0 (bursting-node)** or **CCI 1.0 (virtual-kubelet)**. If you use CCI 1.0, select **CCI 1.0 (virtual-kubelet)**. If you use CCI 2.0, select **CCI 2.0 (bursting-node)**.

Step 5 Create a workload on the CCE console by following the instructions in **Creating a Workload**. After the workload is created, click **OK**.

----End

Using YAML

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

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

Step 3 Add the **virtual-kubelet.io/burst-to-cci** label to the YAML file of the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
  labels:
    bursting.cci.io/burst-to-cci: 'auto'    # Schedules the workload to CCI.
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
```

```
containers:
  - image: 'nginx:perl'
    name: container-0
    resources:
      requests:
        cpu: 250m
        memory: 512Mi
      limits:
        cpu: 250m
        memory: 512Mi
    volumeMounts: []
imagePullSecrets:
  - name: default-secret
```

Table 5-4 Key parameters

Parameter	Type	Description
bursting.cci.io/burst-to-cci	String	Policy for automatically scheduling the workload from the CCE cluster to CCI. The options are as follows: ● **enforce**: The workload is forcibly scheduled to CCI 2.0. ● **auto**: The workload is scheduled to CCI 2.0 based on the scoring results provided by the cluster scheduler. ● **localPrefer**: The workload is preferentially scheduled to a CCE cluster. If cluster resources are insufficient, Workloads are elastically scheduled to CCI 2.0. ● **off**: The workload will not be scheduled to CCI 2.0.

Step 4 Click **OK**.

----End

Method 2: Using a Profile

You can configure profiles to control pod scheduling to CCI using either the console or YAML files.

Using the Console

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

Step 2 In the navigation pane, choose **Policies** > **CCI Scaling Policies**.

Step 3 Click **Create CCI Scaling Policy** and configure the parameters.

Table 5-5 Parameters for creating a CCI scaling policy

Parameter	Description
Policy Name	Enter a policy name.
Namespace	Select the namespace where the scheduling policy applies. You can select an existing namespace or create a namespace. For details, see **Creating a Namespace**.
Workload	Enter a key and value or reference a workload label.
Scheduling Policy	Select a scheduling policy. ● **Priority scheduling**: Pods will be preferentially scheduled to nodes in the current CCE cluster. When the node resources are insufficient, pods will be scheduled to CCI. ● **Force scheduling**: All pods will be scheduled to CCI.
Scale to	● **Local**: Set the current CCE cluster. ● **CCI**: Set the maximum number of pods that can run on CCI.
Maximum Pods	Enter the maximum number of pods that can run in the CCE cluster or on CCI.
CCE Scale-in Priority	Value range: **–100** to **100**. A larger value indicates a higher priority.
CCI Scale-in Priority	Value range: **–100** to **100**. A larger value indicates a higher priority.
CCI Resource Pool	● **CCI 2.0 (bursting-node)**: next-generation serverless resource pool ● **CCI 1.0 (virtual-kubelet)**: existing serverless resource pool, which will be unavailable soon.

Step 4 Click **OK**.

Step 5 In the navigation pane, choose **Workloads**. Then click **Create Workload**. In **Advanced Settings** > **Labels and Annotations**, add a pod label. The key and value must be the same as those for the workload in **Step 3**. For other parameters, see **Creating a Workload**.

Step 6 Click **Create Workload**.

----End

Using YAML

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

Step 2 In the navigation pane, choose **Policies** > **CCI Scaling Policies**.

Step 3 Click **Create from YAML** to create a profile.

Example 1: Configure **local maxNum** and **scaleDownPriority** to limit the maximum number of pods for the workloads that can be scheduled to the CCE cluster.

```
apiVersion: scheduling.cci.io/v2
kind: ScheduleProfile
metadata:
  name: test-local-profile
  namespace: default
spec:
  objectLabels:
    matchLabels:
      app: nginx
  strategy: localPrefer
  virtualNodes:
    - type: bursting-node
  location:
    local:
      maxNum: 20 # maxNum can be configured either for local or cci.
      scaleDownPriority: 2
    cci:
      scaleDownPriority: 10
```

Example 2: Configure **maxNum** and **scaleDownPriority** for **cci** to limit the maximum number of pods that are allowed for the workloads scheduled to CCI.

```
apiVersion: scheduling.cci.io/v2
kind: ScheduleProfile
metadata:
  name: test-cci-profile
  namespace: default
spec:
  objectLabels:
    matchLabels:
      app: nginx
  strategy: localPrefer
  virtualNodes:
    - type: bursting-node
  location:
    local: {}
    cci:
      maxNum: 20 # maxNum can be configured either for local or cci.
      scaleDownPriority: 10
```

Table 5-6 Key parameters

Parameter	Type	Description
strategy	String	Policy for automatically scheduling a workload from a CCE cluster to CCI 2.0. The options are as follows: • **enforce**: The workload is forcibly scheduled to CCI 2.0. • **auto**: The workload is scheduled to CCI 2.0 based on the scoring results provided by the cluster scheduler. • **localPrefer**: The workload is preferentially scheduled to a CCE cluster. If cluster resources are insufficient, Workloads are elastically scheduled to CCI 2.0.
maxNum	int	Maximum number of pods. The value ranges from **0** to **int32**.

Parameter	Type	Description
scaleDownPriority	int	Scale-in priority. The larger the value, the earlier the associated pods are removed. The value ranges from **-100** to **100**.

NOTE

- In the **location** field, configure the **local** field for CCE and **cci** for CCI to control the number of pods and scale-in priority.
- **maxNum** can be configured either for **local** or **cci**.
- Scale-in priority is optional. If it is not specified, the default value is set to **nil**.

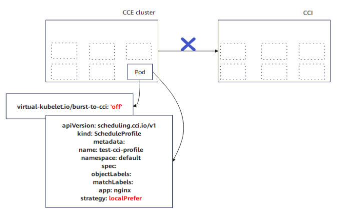
Step 4 Click **OK**.

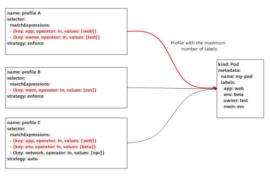
Step 5 Create a Deployment, use the selector to select the pods labeled with **app:nginx**, and associate the pods with the profile.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: nginx
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: container-1
          image: nginx:latest
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
        - name: default-secret
```

Table 5-7 Special scenarios

Scenario	How to Schedule

Both a label and a profile are used to schedule the workloads to CCI 2.0.	The scheduling priority of the label is higher than that of the profile.
	For example, if the scheduling policy of the label is **off** and the scheduling policy of the profile is set to **enforce**, the workloads will not be scheduled to CCI 2.0.
	
	
	`virtual-kubelet.io/burst-to-cci: 'off'`
	`apiVersion: scheduling.cci.io/v1`
	`kind: ScheduleProfile`
	`metadata:`
	`name: test-cci-profile`
	`namespace: default`
	`spec:`
	`objectLabels:`
	`matchLabels:`
	`app: nginx`
	`strategy: localPrefer`

There are multiple profiles specified for a pod.	A pod can only have one profile. If a pod has multiple profiles, the profile that can associate the maximum of labels is used. If there are multiple profiles that can associate an equal number of labels, the profile whose name has the smallest alphabetical order is used.
	In this figure, the pod is finally associated with profileA.
	
	
	`name: profile A`
	`selector:`
	`  matchExpressions:`
	`  - {key: app, operator: In, values: [web]}`
	`  - {key: owner, operator: In, values: [test]}`
	`strategy: enforce`
	
	`name: profile B`
	`selector:`
	`  matchExpressions:`
	`  - {key: mem, operator: In, values: [evs]}`
	`strategy: enforce`
	
	`name: profile C`
	`selector:`
	`  matchExpressions:`
	`  - {key: app, operator: In, values: [web]}`
	`  - {key: env, operator: In, values: [beta]}`
	`  - {key: network, operator: In, values: [vpc]}`
	`strategy: auto`
	
	`kind: Pod`
	`metadata:`
	`  name: my-pod`
	`  labels:`
	`    app: web`
	`    env: beta`
	`    owner: test`
	`    mem: evs`

Step 6 Click **OK**.

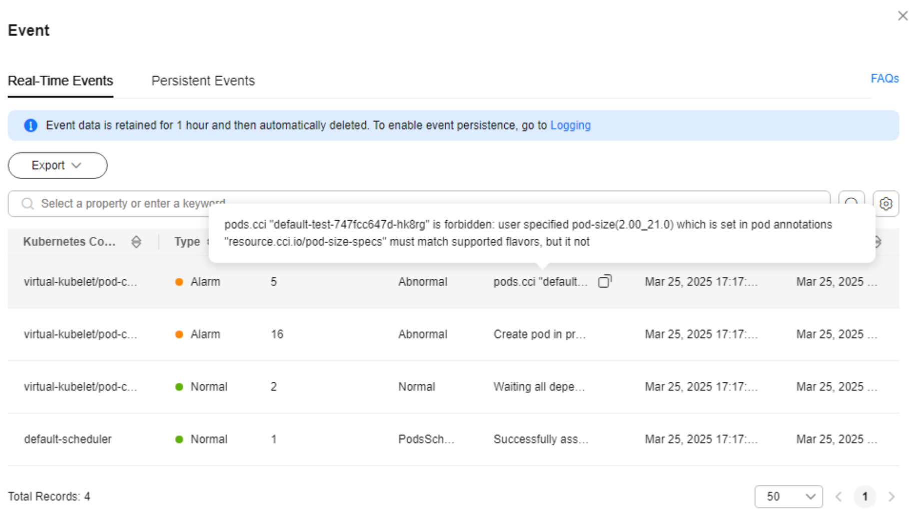
----End

5.4 Rounding Up Pod Resources

The CCE Cloud Bursting Engine for CCI add-on can select appropriate resource specifications based on the vCPUs and memory of the workloads to improve resource utilization. This section describes how appropriate resource specifications are selected.

Constraints

- At least one container in the pod that is scheduled to CCI must have vCPU and memory limits or requests specified or the **resource.cci.io/pod-size-specs** annotation.
- Pods that are scheduled to CCI 2.0 must comply with the flavors of CCI 2.0. If a pod does not comply with the flavors of CCI 2.0, an error similar to the following will be reported in the pod event on CCE, and the pod will not be created on CCI 2.0. For details about the flavors, see **Upgrading Pod Flavors**.



Event

Real-Time Events | Persistent Events | FAQs

Event data is retained for 1 hour and then automatically deleted. To enable event persistence, go to Logging

pods.cci "default-test-747fcc647d-hk8rg" is forbidden: user specified pod-size(2.00_21.0) which is set in pod annotations "resource.cci.io/pod-size-specs" must match supported flavors, but it not

Kubernetes Co...	Type					
virtual-kubelet/pod-c...	Alarm	5	Abnormal	pods.cci "default...	Mar 25, 2025 17:17:...	Mar 25, 2025 ...
virtual-kubelet/pod-c...	Alarm	16	Abnormal	Create pod in pr...	Mar 25, 2025 17:17:...	Mar 25, 2025 ...
virtual-kubelet/pod-c...	Normal	2	Normal	Waiting all depe...	Mar 25, 2025 17:17:...	Mar 25, 2025 ...
default-scheduler	Normal	1	PodsSch...	Successfully ass...	Mar 25, 2025 17:17:...	Mar 25, 2025 ...

Total Records: 4

How Resource Specifications Are Calculated

Some necessary system components required by the pods occupy system resources. There is a difference between the memory in the pod flavor and the allocatable pod memory. CCI calculates the pod memory that can be allocated as follows:

- Memory in the pod flavor ≤ 2 GiB

  Allocatable pod memory = **Memory in the pod flavor**

- Memory in the pod flavor > 2 GiB

  Allocatable pod memory = **Memory in the pod flavor** – **Memory reserved for CCI** – **Memory reserved for the OS**

CAUTION

Memory in the pod flavor is the value of **${memoryCeil}** displayed in the pod annotation in the format of *resource.cci.io/size=${cpuCeil}_${memoryCeil}*.

For details, see **Reserved System Overhead**.

Reserved System Overhead

For pods that are scheduled to CCI 2.0, you can also use the **resource.cci.io/memory-reservation** and **resource.cci.io/memory-burst-size** annotations in **pod/podTemplate** to enable system overhead reservation. For details about the rules, see **Reserved System Overhead**.

5.5 Images

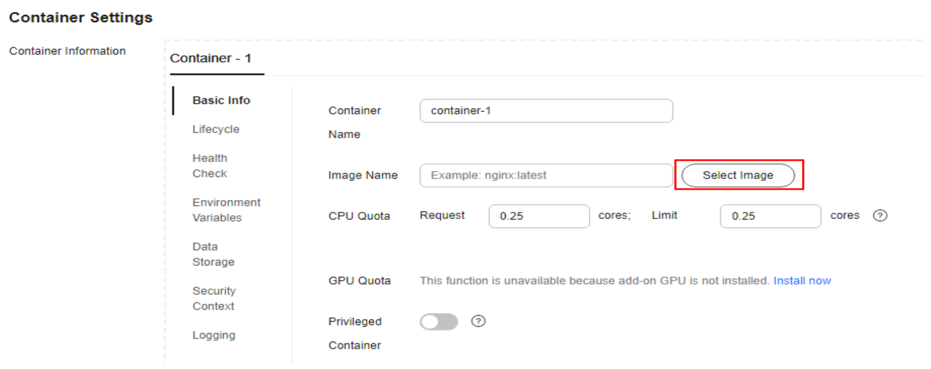
You can manage images using SWR. This section describes how images are pulled when the CCE Cloud Bursting Engine for CCI add-on is installed in a CCE cluster.

- **Pulling an Image from SWR on the Console**

- **Using Both SWR and a Third-Party Image Repository**

Pulling an Image from SWR on the Console

- **Method 1: Selecting an Image from SWR on the CCE Console**
  a. Upload an image to SWR. For details, see the **SWR documentation**.
  b. Create a workload on the CCE cluster console and select an image.



  c. The image will be pulled from SWR. Ensure that your image has been pushed to SWR. For details, see the **SWR documentation**.

- **Method 2: Select an SWR Image by Configuring a Node in a CCE Cluster**
  a. View the image path in the SWR image repository, for example, swr.cn-north-4.myhuaweicloud.com/cci-test/nginx:1.0.0.x86_64_test.

     cn-north-4 indicates the region, and **cci-test** is the SWR organization.
  b. Log in to a CCE cluster node.
  c. Configure the YAML file of the workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  namespace: default
  labels:
    bursting.cci.io/burst-to-cci: enforce    # Schedules the workload to CCI 2.0.
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - image: swr.***.com/cci-test/nginx:1.0.0.x86_64_test
          name: container-0
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts: []
      imagePullSecrets:
        - name: default-secret
```

d. Deploy the workload.
```
kubectl apply -f  test-deployment.yaml
```

Using Both SWR and a Third-Party Image Repository

Scenario

In some cases, an image can be pulled from a third-party image repository when you create a workload on the CCE cluster console. The pulled image can be synchronized to SWR so that the workload scheduled to CCI uses the image during traffic spikes. This speeds up image pull.

Procedure

Configure **annotations** in the YAML file of the workload. The following is an example:

```
bursting.cci.io/image-replacement: '[
    {"repositoryPrefix":"harbor.domain","replaceWith":"swr.***.com/org1"},
    {"repositoryPrefix":"","replaceWith":"swr.***.com/org1"},
    {"repositoryPrefix":"harbor.domain/a/b/c/d","replaceWith":"swr.***.com/org2"}
]'
```

NOTE

- Replacement policies can be executed in any sequence.
- Multiple replacement policies can be configured. The value of **repositoryPrefix** for each policy must be unique.

Replacement Policy Key	Description	Remarks
repositoryPrefix	Image prefix that you want to match and replace.	• If this field is left empty, all containers whose **image** values do not contain slashes (/) are matched. • If this field is not empty, all containers whose **image** values have the same prefix and end with slashes (/) are matched. • This field cannot end with a slash (/) and is verified the same as the container image name.
replaceWith	Image prefix to be used.	• The value of this field cannot be the same as that of **repositoryPrefix**. • This field cannot end with a slash (/) and is verified the same as the container image name.

Table 5-8 Annotations

Annotation	Before Replacement	After Replacement	Description
bursting.cci.io/image-replacement: '[{"repositoryPrefix":"harbor.domain","replaceWith":"swr.***.com/org1"}]'	containers: - name: container-0 image: 'harbor.domain/ ubuntu:latest'	containers: - name: container-0 image: 'swr.***.com/ org1/ubuntu:latest'	**repositoryPrefix** matches the domain name of a third-party repository.
bursting.cci.io/image-replacement: '[{"repositoryPrefix":"","replaceWith":"swr.***.com/org1"}]'	containers: - name: container-1 image: 'nginx:latest'	containers: - name: container-1 image: 'swr.***.com/ org1/nginx:latest'	**repositoryPrefix** is left empty.
bursting.cci.io/image-replacement: '[{"repositoryPrefix":"harbor.domain/a/b/c/ d","replaceWith":"swr.***.com/ org2"}]'	containers: - name: container-2 image: 'harbor.domain/a/b/c/d/ redis:latest'	containers: - name: container-2 image: 'swr.***.com/ org2/redis:latest'	**repositoryPrefix** matches the domain name of a third-party repository and the organization directory.

5.6 Storage

There are multiple storage volume types that can be used by the pods for the workloads scheduled to CCI 2.0. In this section, you can learn about:

- Storage volume types used by the pods for the workloads scheduled to CCI 2.0.

- Typical scenarios of hostPath volumes and how to use them.

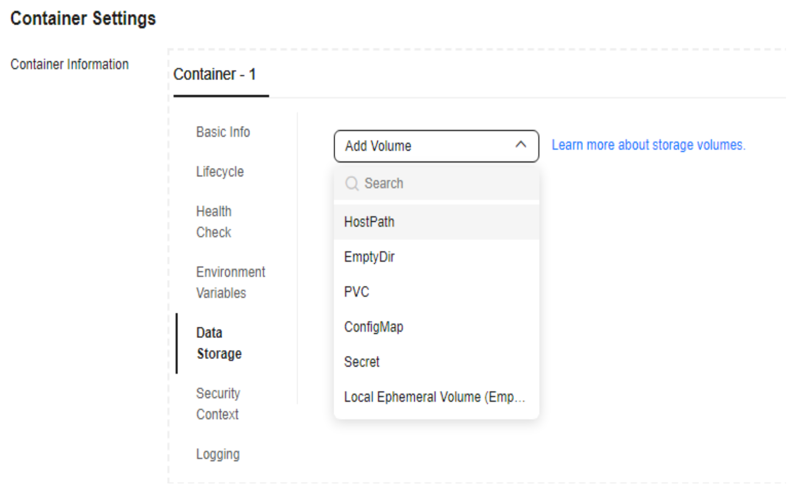
Constraints

CCE cluster workloads that can be scheduled to CCI include ConfigMap, Secret, emptyDir, DownwardAPI, Projected, and PersistentVolumeClaims volumes, and the DownwardAPI and Projected volumes can only be used in the CCE Cloud Bursting Engine for CCI add-on of v1.3.25 or later.

- emptyDir: Subpaths are not supported.
- persistentVolumeClaim: Only SFS Turbo is supported, with StorageClass set to CSI. Volcano Scheduler v1.17.10 or earlier does not support scheduling of all cloud storage types.
- Projected: If a source of the serviceAccountToken type is configured, the token in the corresponding service-account-token secret is mounted after the pod is scheduled to CCI. The token is valid for a long time and has no expected audience. This means the **expirationSeconds** and **audience** configurations do not take effect.

Storage Volume Types

There are various storage volume types on the CCE cluster console.



The following table lists the storage volume types.

Volume Type	Supported by CCI	Remarks
hostPath	No	● CCI underlying clusters are used by all users so using the hostPath volumes presents many security risks. As a result, hostPath volumes are unavailable. ● The CCE Cloud Bursting Engine for CCI add-on of v1.5.9 or later supports hostPath volumes whose path is **/etc/localtime**. After the configuration, the time zone of CCI containers is the same as that of CCE nodes.
ConfigMap	Yes	-
Secret	Yes	-
emptyDir	Yes	The **sizeLimit** parameter of emptyDir is only valid when **emptyDir.medium** is set to **Memory**.
DownwardAPI	Yes	-
Projected	Yes	If a source of the serviceAccountToken type is configured, the token in the corresponding service-account-token secret is injected into the pod scheduled to CCI. The token is valid for a long time and has no audience. This means the settings of **expirationSeconds** and **audience** do not take effect.
PersistentVolume-Claims	Yes	Only SFS Turbo is supported, with StorageClass set to CSI.

How to Use hostPath

Scenario

A hostPath volume can be used for storage when CCE or other Kubernetes clusters are used. However, CCI underlying clusters are used by all users so using hostPath volumes presents many security risks. As a result, hostPath volumes are unavailable. When a pod with a hostPath volume mounted is scheduled to CCI, the pods will be rejected. If hostPath configured in **spec.volumes** for a pod cannot be changed, you can configure annotations to allow the pod to be scheduled to CCI. During the bursting verification, **hostPath** needs to be removed or replaced with **emptyDir**.

Constraints

- localDir and flexVolume are not supported currently.

Procedure

You can add annotations to **Pod.Annotations** to convert **hostPath** to **emptyDir**.

- Replace **hostPath** with **emptyDir**.
```
bursting.cci.io/hostpath-replacement: '[{"name":"source-hostpath-volume-3","policyType":"replaceByEmptyDir","emptyDir":{"sizeLimit":"10Gi"}}]'
```
- Ignore a single hostPath volume:
```
bursting.cci.io/hostpath-replacement: '[{"name":"source-hostpath-volume-1","policyType":"remove"}]'
```
- Ignore all **hostPath** volumes.
```
bursting.cci.io/hostpath-replacement: '[{"name":"*","policyType":"remove"}]'
```
- Replace each hostPath volume with a different storage type.
```
bursting.cci.io/hostpath-replacement: '[{"name":"source-hostpath-volume-1","policyType":"remove"},
{"name":"source-hostpath-volume-3","policyType":"replaceByEmptyDir","emptyDir":
{"sizeLimit":"10Gi"}}]'
```

NOTE

For hostPath volumes whose path is **/etc/localtime**, if the name of a hostPath volume is the same as that of a replacement policy, the hostPath volume will be replaced. If the replacement policy name is *, hostPath volumes whose path is **/etc/localtime** will not be replaced.

Example (a Deployment):
```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    description: ''
  labels:
    bursting.cci.io/burst-to-cci: enforce
    appgroup: ''
    version: v1
  name: test
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
      version: v1
  template:
    metadata:
      labels:
        app: test
        version: v1
      annotations:
        bursting.cci.io/hostpath-replacement: '[{"name": "test-log2", "policyType": "remove"}, {"name":
"test-log", "policyType": "replaceByEmptyDir", "emptyDir":{"sizeLimit":"10Gi"}}, {"name": "test-log1",
"policyType": "remove" }]'
    spec:
      containers:
        - name: container-1
          image: nginx
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: test-log
              mountPath: /tmp/log
            - name: test-log1
              mountPath: /tmp/log1
            - name: test-log2
              mountPath: /tmp/log2
      volumes:
      - hostPath:
```

```
      path: /var/paas/sys/log/virtual-kubelet
      type: ""
    name: test-log
  - hostPath:
      path: /var/paas/sys/log
      type: ""
    name: test-log1
  - hostPath:
      path: /var/paas/sys/log2
      type: ""
    name: test-log2
```

5.7 Logging

5.7.1 Collecting Logs Using LTS

CCI 2.0 works with LTS to collect application logs and report these logs to LTS so that you can use them for troubleshooting.

Log Collection Reliability

The log system's main purpose is to record all stages of data for service components, including startup, initialization, exit, runtime details, and exceptions. It is primarily employed in O&M scenarios for tasks like checking component status and analyzing fault causes.

Standard streams (stdout and stderr) and local log files use non-persistent storage. However, data integrity may be compromised due to the following risks:

- Log rotation and compression potentially deleting old files
- Temporary storage volumes being cleared when Kubernetes pods end
- Automatic OS cleanup triggered by limited node storage space

While the Cloud Native Log Collection add-on employs techniques like multi-level buffering, priority queues, and resumable uploads to enhance log collection reliability, logs could still be lost in the following situations:

- The service log throughput surpasses the collector's processing capacity.
- The service pod is abruptly terminated and reclaimed by CCE.
- The log collector pod experiences exceptions.

The following lists some recommended best practices for cloud native log management. You can review and implement them thoughtfully.

- Use dedicated, high-reliable streams to record critical service data (for example, financial transactions) and store the data in persistent storage.
- Avoid storing sensitive information like customer details, payment credentials, and session tokens in logs.

Constraints

- Logs cannot be collected from the directory that a specified system, device, cgroup, or tmpfs is mounted to.
- A single-line log that exceeds 250 KB will not be collected.

- Regular expression match is only supported when a full path with a complete file name is specified.
- The name of each file to be logged must be unique in a container. If there are files with duplicate names, only the logs of one file are collected.
- After a pod is started, the log collection configuration cannot be updated. If the configuration is updated, the pod must be restarted for the configuration to take effect.
- A directory to be logged must exist before the container is started. If a directory is created after the container is started, the logs of the directory and of the files in that directory cannot be collected.
- If the name of a file exceeds 190 characters, its logs cannot be collected.
- Logs of init containers cannot be collected.
- Logs of Kunpeng pods cannot be collected.

Step 1 Create a Log Group

Step 1 Log in to the management console and choose **Management & Deployment** > **Log Tank Service**.

Step 2 Log in to the **LTS console**.

Step 3 On the **Log Management** page, click **Create Log Group**.

Step 4 On the displayed page, set log group parameters by referring to **Table 5-9**.

Table 5-9 Log group parameters

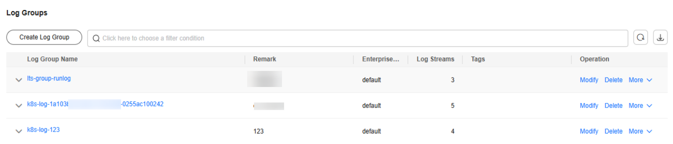
Parameter	Description
Log Group Name	A log group is the basic unit for LTS to manage logs. It is used to classify log streams. If there are too many logs to collect, separate logs into different log groups based on log types, and name log groups in an easily identifiable way. LTS automatically generates a default log group name. You are advised to customize one based on your service. You can also change it after the log group is created. The naming rules are as follows: ● Enter 1 to 64 characters, including only letters, digits, hyphens (-), underscores (_), and periods (.). Do not start with a period or underscore or end with a period. ● Each log group name must be unique.

Parameter	Description
Enterprise Project Name	Enterprise projects allow you to manage cloud resources and users by project. By default, the **default** enterprise project is used. You are advised to select an enterprise project that fits your service needs. To see all available options, click **View Enterprise Projects**. ● You can use enterprise projects only after enabling the enterprise project function. For details, see **Enabling the Enterprise Project Function**. ● You can remove resources from an enterprise project to another. For details, see **Removing Resources from an Enterprise Project**.
Log Retention (Days)	Specify the log retention period for the log group, that is, how many days the logs will be stored in LTS after being reported to LTS. By default, logs are retained for 30 days. You can set the retention period to one to 365 days. LTS periodically deletes logs based on the configured log retention period. For example, if you set the period to 30 days, LTS retains the reported logs for 30 days and then deletes them. **NOTE** The log retention period can be extended to 1,095 days. This feature is available only to whitelisted users. To enable it, **submit a service ticket**. ● By default, logs are retained for seven days. You can set the retention period to one to seven days. The logs that exceed the retention period will be automatically deleted. You can transfer logs to OBS buckets for long-term storage. **NOTE** The retention period for small and medium specifications is seven days by default. For large, ultra-large, and ultra-large II specifications, the retention period can be changed to seven to 365 days. To retain logs for seven to 365 days, you need to set the retention period when deploying LTS of the large, ultra-large, or ultra-large II specifications or when expanding capacity with log incremental packages. For details, see "AOM Installation Guide" > "Installing Cloud Services Using HCC Turnkey" in *Huawei Cloud Stack 8.6.0 Software Installation Guide for gPaaS & AI DaaS Services*. ● Raw logs reported to LTS will be deleted in the early morning of the next day after the log retention period expires.

Parameter	Description
Tag	Tag the log group as required. Click **Add** and enter a tag key and value. If you enable **Apply to Log Stream**, the tag will be applied to all log streams in the log group. To add more tags, repeat this step. A maximum of 20 tags can be added. **Tag key restrictions:** ● A tag key can contain letters, digits, spaces, and special characters (_.:=+-@), but cannot start or end with a space or start with **_sys_**. ● A tag key can contain up to 128 characters. ● Each tag key must be unique. **Tag value restrictions:** ● A tag value can contain letters, digits, spaces, and the following special characters: _.:=+-@ ● A tag value can contain up to 255 characters. **Tag policies:** If your organization has configured tag policies for LTS, follow the policies when adding tags to log groups, log streams, log ingestion configurations, host groups, and alarm rules. Non-compliant tags may cause the creation of these resources to fail. Contact your administrator to learn more about the tag policies. For details about tag policies, see **Overview of a Tag Policy**. For details about tag management, see **Managing Tags**. **Deleting a tag:** Click **Delete** in the **Operation** column of the tag. **WARNING** Deleted tags cannot be recovered. If a tag is used by a transfer task, you need to modify the task configuration after deleting the tag.
Remark	Enter remarks. The value contains up to 1,024 characters.

Step 5 Click **OK**. The created log group will be displayed in the log group list.

Figure 5-2 Log group



Log Groups

Log Group Name	Remark	Enterprise...	Log Streams	Tags	Operation
lts-group-runlog	[redacted]	default	3		Modify Delete More
k8s-log-1a103[redacted]-0255ac100242	[redacted]	default	5		Modify Delete More
k8s-log-123	123	default	4		Modify Delete More

- In the log group list, view information such as the log group name, tags, and log streams.
- Click the log group name to access the log details page.

----End

Step 2 Create a Log Stream

1. On the LTS console, click ⌄ on the left of a log group name.
2. In the upper left corner of the displayed page, click **Create Log Stream** and then enter a log stream name. The log stream name:
   - Can contain only letters, numbers, underscores (_), hyphens (-), and periods (.). The name cannot start with a period or underscore, or end with a period.
   - Can contain 1 to 64 characters.

   NOTE

   Collected logs are sent to the created log stream. If there are a large number of logs, you can create multiple log streams and name them for quick log search.
3. Select an enterprise project. You can click **View Enterprise Projects** to view all enterprise projects.
4. If you enable **Log Retention Duration** on this page, you can set the log retention duration specifically for the log stream. If you disable it, the log stream will inherit the log retention setting of the log group.
5. Anonymous write is disabled by default and is suitable for log reporting on Android, iOS, applets, and browsers. If anonymous write is enabled, the anonymous write permission is granted to the log stream, and no valid authentication is performed, which may generate dirty data.
6. Set the tag in the *Key*=*Value* format, for example, a=b.
7. Enter remarks. A maximum of 1,024 characters are allowed.
8. Click **OK**. In the log stream list, you can view information such as the log stream name and operations.

   NOTE

   - You can view the log stream billing information. For details, see **Price Calculator**.
   - The function of reporting SDRs by log stream is in Friendly User Test (FUT). You can **submit a service ticket** to enable it.

Step 3 Obtain the Log Group ID and Log Stream ID

1. In the navigation pane, choose **Log Management**.
2. Select the log group created in **Step 1 Create a Log Group** and click **More** > **Details** in the **Operation** column.
3. Copy the log group ID.
4. Click ⌄ on the left of the log group name.
5. Select the log stream created in **Step 2 Create a Log Stream** and click **Details** in the **Operation** column.
6. Copy the log stream ID.

Step 4 Create a Deployment on the CCE Console

1. Log in to the **CCE console**.
2. In the navigation pane, choose **Workloads**. On the displayed page, click the **Deployments** tab.

3. Click **Create from YAML** to create a Deployment using **YAML** or **JSON**.
   - Method 1: Use YAML to create a Deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test
  labels:
    bursting.cci.io/burst-to-cci: enforce
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      annotations:
        logconf.k8s.io/fluent-bit-log-type: lts      # (Mandatory) LTS is used to collect logs.
        logconfigs.logging.openvessel.io: |
          {
            "default-config": {   # You can set the log collection paths of multiple containers.
stdout.log indicates standard output. /root/out.log contains the text logs in rootfs (volumes
included). /data/emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
              "container_files": {
                "container-0": "stdout.log;/root/out.log;/data/emptydir-volume/*.log"
              },
              "regulation": "", #Regular expression for matching multi-line logs. For details, see
Regular Expression Matching Rules.
              "lts-log-info": {    #Configure a log group and a log stream.
                "<log-group-ID>": "<log-stream-ID>"  #Replace <log-group-ID> and <log-stream-
ID> with those obtained in Step 3.
              }
            }
          }
        resource.cci.io/pod-size-specs: 2.00_8.0
      labels:
        app: test
        sys_enterprise_project_id: "0"
    spec:
      containers:
      - image: busybox:latest
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', "while true; do echo hello; touch /root/out.log; echo hello >> /root/
out.log; touch /data/emptydir-volume/emptydir.log; echo hello >> /data/emptydir-volume/
emptydir.log; sleep 10; done"]
        lifecycle: {}
        volumeMounts:
        - name: emptydir-volume
          mountPath: /data/emptydir-volume
        - name: emptydir-memory-volume
          mountPath: /data/emptydir-memory-volume
        name: container-0
        resources:
          limits:
            cpu: 100m
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: Default
      volumes:
      - name: emptydir-volume
        emptyDir: {}
      - name: emptydir-memory-volume
        emptyDir:
          sizeLimit: 1Gi
          medium: Memory
```

Table 5-10 Key parameters

Parameter	Mandatory	Type	Description
logconf.k8s.io/fluent-bit-log-type	Yes	String	• Description: Log collection mode • Constraints: This parameter is mandatory. • Value: **lts**
logconfigs.logging.openvessel.io	Yes	String	• Description: Log collection configuration

NOTE

When using YAML, delete the comments of the parameters in the **logconfigs.logging.openvessel.io** field.

– Method 2: Use JSON to create a Deployment.

```
{
   "default-config": {
           "container_files": {   // You can set the log collection paths of multiple containers. stdout.log indicates standard output. /root/out.log contains the text logs in rootfs (volumes included). /data/emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
           "container-0": "stdout.log;/root/out.log;/data/emptydir-volume/*.log",
           "container-1": "stdout.log"
        },
        "regulation": "", // Regular expression matching rule for collecting multi-line logs. For details about regular expression matching rules, see https://docs.fluentbit.io/manual/pipeline/parsers/configuring-parser.
        "lts-log-info": { // Configure a log group and a log stream.
           "<log-group-ID>": "<log-stream-ID>"  // Replace <log-group-ID> and <log-stream-ID> with those obtained in Step 3.
        }
   },
   "multi-config": {
           "container_files": {   // You can set the log collection paths of multiple containers. stdout.log indicates standard output. /root/out.log contains the text logs in rootfs (volumes included). /data/emptydir-xxx/*.log indicates the directories in rootfs (volumes included).
           "container-0": "stdout.log;/root/out.log;/data/emptydir-memory-volume/*.log"
        },
        "regulation": "/(?<log>\\d+-\\d+-\\d+ \\d+:\\d+:\\d+.*)/",  // Regular expression matching rule for collecting multi-line logs
        "lts-log-info": { // Configure the same log group and log stream.
           "<log-group-ID>": "<log-stream-ID>"  // Replace <log-group-ID> and <log-stream-ID> with those obtained in Step 3.
        }
   }
}
```

CAUTION

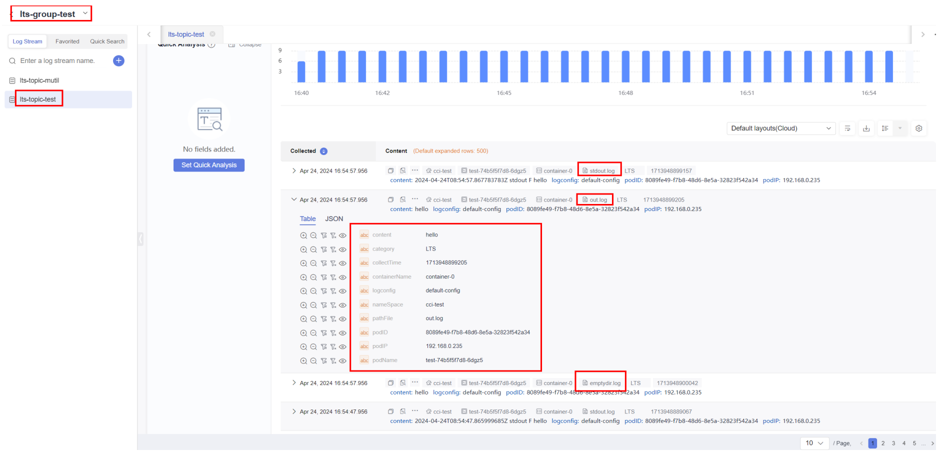
When you are creating a workload using YAML or JSON, you need to add the log group ID and log stream ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID** to the **lts-log-info** parameter, for example, **\"lts-log-info\":***{\"log-group-ID\":\"log-stream-ID\"}*

- Replace *log-group-ID* with the log group ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID**.
- Replace *log-stream-ID* with the log stream ID obtained in **Step 3 Obtain the Log Group ID and Log Stream ID**.

4. Click **OK**.
5. Click the created Deployment to view its status.

Step 5 Check Log Reporting

1. Log in to the LTS console. In the navigation pane, choose **Log Management** and click the name of the log group.
2. Go to the log details page to view logs.



```
Collected   Content (Default expanded rows: 500)
> Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  stdout.log  LTS  1713948899157
    content: 2024-04-24T08:54:57.867783783Z stdout F hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
v Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  out.log  LTS  1713948899205
    content: hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
    content        hello
    category       LTS
    collectTime    1713948899205
    containerName  container-0
    logconfig      default-config
    nameSpace      cci-test
    pathFile       out.log
    podID          8089fe49-f7b8-48d6-8e5a-32823f542a34
    podIP          192.168.0.235
    podName        test-74b5f5f7d8-6dgz5
> Apr 24, 2024 16:54:57.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  emptydir.log  LTS  1713948900042
    content: hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
> Apr 24, 2024 16:54:47.956   cci-test  test-74b5f5f7d8-6dgz5  container-0  stdout.log  LTS  1713948889067
    content: 2024-04-24T08:54:47.865999685Z stdout F hello  logconfig: default-config  podID: 8089fe49-f7b8-48d6-8e5a-32823f542a34  podIP: 192.168.0.235
10 / Page,  1 2 3 4 5 ...
```

5.7.2 Collecting Logs Using a Sidecar Container

You can use **hostpath-replacement** to replace hostPath with emptyDir supported by CCI, inject a sidecar container to collect logs through **pod-sidecars**, and mount the standard output logs to the sidecar container through **mount-stdlog-containers**.

Prerequisites

You have configured the sidecar container by referring to **Configuring a Sidecar Container**.

Constraints

volume.cci.io/mount-stdlog-containers and **volume.cci.io/mount-stdlog-containers-path** cannot be configured at the same time.

Using a YAML File

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    version: v1
    bursting.cci.io/burst-to-cci: enforce
  name: deploy-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-example
      version: v1
  template:
    metadata:
      annotations:
        bursting.cci.io/hostpath-replacement: '[{"name":"log","policyType":"replaceByEmptyDir","emptyDir":
{"sizeLimit":"10Gi"}}]'
        bursting.cci.io/pod-sidecars: '{"containers":[{"name":"log-sidecar","image":"fluent-
bit:latest","resources":{},"volumeMounts":[{"name":"log","mountPath":"/tmp/log"}],"position":"tail"}]}'
        volume.cci.io/mount-stdlog-containers: log-sidecar # Name of the container where standard output
logs are to be mounted
      labels:
        app: deploy-example
        version: v1
    spec:
      containers:
      - image: nginx:latest
        name: nginx
        resources:
          limits:
            cpu: "1"
            memory: 2Gi
          requests:
            cpu: "1"
            memory: 2Gi
        volumeMounts:
        - mountPath: /tmp/log
          name: log
      imagePullSecrets:
        - name: default-secret
      volumes:
      - hostPath:
          path: /local/log
        name: log
```

With the preceding configurations, the sidecar container can collect two types of logs:

- Standard output logs. The sidecar container can collect the standard output logs of service containers as files and store these standard output logs to the **/var/log/pods** directory.
- File logs. Service containers output log files to the emptyDir volume shared with the sidecar container, which then collects the files from the volume.

Table 5-11 Pod annotations

Annotation	Type	Description	Example Value

volume.cci.io/mount-stdlog-containers	String	1. Name of the container where standard output logs are mounted. To mount the standard output logs of multiple containers, separate their names with commas (,). You can also set the value to an asterisk (*) to mount the standard output logs of all containers. If the value is set to *****, it cannot be set to any other container name.
		2. After a pod that matches the annotation starts, the directory that contains the standard output logs of all containers in the pod is mounted to the **/var/log/pods** directory of the container.
			Example 1:
			`"container-0,container-1"`
			Example 2:
			`"*"`

volume.cci.io/mount-stdlog-containers-path	String	1. Container name and mount path that the standard output logs are mounted. The value is in JSON format. If the container name is set to *****, the logs of all containers are mounted to the specified container. If the value is set to *****, it cannot be set to any other container name.
		2. After any container that matches the annotation starts, the directory that contains the standard output logs of all containers in the pod is mounted to the specified path of the container.
			Example 1:
			`"{\"container-0\":\"/var/log/pods\",\"container-1\":\"/tmp/log/pods\"}"`
			Example 2:
			`"{\"*\":\"/tmp/log/pods\"}"`

5.8 Configuring a ClusterExtensionProfile or ExtensionProfile

You can configure **ClusterExtensionProfile** or **ExtensionProfile** objects to schedule workloads to CCI 2.0. ClusterExtensionProfiles are a cluster-level object, and ExtensionProfiles are of the namespace level. They can be used to configure pods scheduled to CCI 2.0 to reduce the modification to YAML files.

Configuring a ClusterExtensionProfile

Step 1 Generate a YAML file.

The following is an example YAML file:

```
apiVersion: bursting.cci.io/v1
kind: ClusterExtensionProfile
metadata:
  name: test-cluster-profile
spec:
  actions:
    annotations:
      annotation1: value1
    hostPathReplacement:
    - emptyDir:
        sizeLimit: 10Gi
      name: volume1
      policyType: replaceByEmptyDir
    labels:
      label1: value1
  namespaceLabels:
    matchLabels:
      key2: value2
  policy: override
```

Step 2 Create the object.

```
kubectl apply -f <test-cluster-profile>     # YAML file name (Replace it with the actual file name.)
```

Step 3 (Optional) Verify that the object has been created.

```
kubectl get cextp
```

----End

Configuring an ExtensionProfile

Step 1 Generate a YAML file.

The following is an example YAML file:

```
apiVersion: bursting.cci.io/v1
kind: ExtensionProfile
metadata:
  name: test-profile
spec:
  actions:
    annotations:
      annotation1: value1
    hostPathReplacement:
    - emptyDir:
        sizeLimit: 10Gi
      name: volume1
      policyType: replaceByEmptyDir
    labels:
```

```
    label1: value1
  namespaceLabels:
    matchLabels:
      key2: value2
  policy: override
```

Step 2 Create the object.

```
kubectl apply -f <test-profile>    # YAML file name (Replace it with the actual file name.)
```

Step 3 (Optional) Verify that the object has been created.

```
kubectl get extp
```

----End

spec Parameters

Table 5-12 Parameters in **spec**

Parameter	Description
policy	Indicates the policy to be configured. The value can be **addOnly** or **override**. When a configuration conflicts with the original definition of a pod, setting the parameter of the configuration to **override** will overwrite the original definition, but setting it to **addOnly** will not. The default value is **addOnly**.
namespaceLabels	Filters namespaces by namespace label.
objectLabels	Filters pods by pod label. **CAUTION** For details about matchLabels and matchExpressions, see **Kubernetes official documentation**.

actions Parameters

Table 5-13 Key parameters in **actions**

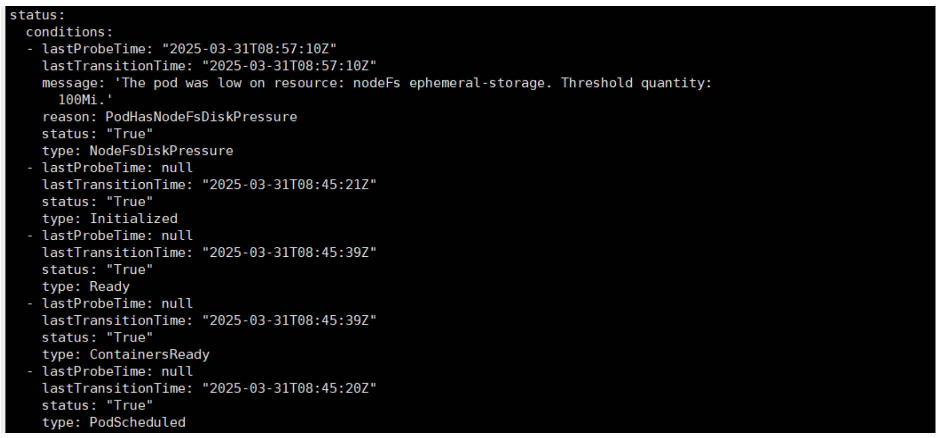
Parameter	Description
annotations	Adds annotations to a pod.
labels	Adds labels to a pod.
imageReplacement	Adds an **image path conversion policy** to a pod.
hostPathReplacement	Adds a **hostPath volume conversion policy** to a pod.
evictions	Configures the pod eviction policy in case of disk pressure.
podSidecars	Adds a **sidecar** to a pod.

Rules for an Effective Profile

- The configuration only takes effect for new pods when they are created. The existing pods are not modified.
- Only one ClusterExtensionProfile and one ExtensionProfile can take effect for a pod. If both types of profiles are matched, the ClusterExtensionProfile takes the precedence.
- If the label of the namespace where the pod is located can be matched by multiple ClusterExtensionProfiles, only the most exactly matched profile takes effect.
- If the label of a pod can be matched by multiple ExtensionProfiles, only the most exactly matched profile takes effect.
- Definition of the most exactly matched profile:
  - The larger the sum of the values of all **matchLabels** and **matchExpressions** in **objectLabels** or **namespaceLabels**, the more accurate the profile is.
  - If the number of profiles is the same, the profile whose name has the smallest alphabetical order is used.

Pod Eviction After Disk Pressure

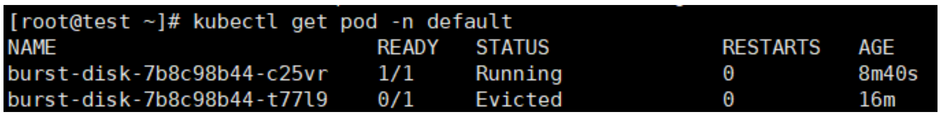
If **image snapshots** are not used and the disk space of a node is less than 100 MiB, a condition of NodeFsDiskPressure is reported for the pod. If image snapshots are used and the temporary disk space of a node is less than 100 MiB or the disk space of the container image is less than 100 MiB, a condition of NodeFsDiskPressure or ImageFsDiskPressure is reported. When the temporary disk space of a node or the disk space of the container image is greater than 100 MiB, the corresponding condition is removed.



```
status:
  conditions:
  - lastProbeTime: "2025-03-31T08:57:10Z"
    lastTransitionTime: "2025-03-31T08:57:10Z"
    message: 'The pod was low on resource: nodeFs ephemeral-storage. Threshold quantity:
      100Mi.'
    reason: PodHasNodeFsDiskPressure
    status: "True"
    type: NodeFsDiskPressure
  - lastProbeTime: null
    lastTransitionTime: "2025-03-31T08:45:21Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2025-03-31T08:45:39Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2025-03-31T08:45:39Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2025-03-31T08:45:20Z"
    status: "True"
    type: PodScheduled
```

In either scenarios where the disk space is insufficient, the system does not perform any operation by default, which may affect the workloads. Based on the Kubernetes eviction and scheduling mechanism, you can configure eviction policies for specified ClusterExtensionProfiles or ExtensionProfiles to automatically evict pods.

Most workloads are Deployments. If a pod is evicted from a Deployment, a new pod is automatically created for the Deployment.



```
[root@test ~]# kubectl get pod -n default
NAME                          READY   STATUS             RESTARTS   AGE
burst-disk-7b8c98b44-c25vr    1/1     Running            0          8m40s
burst-disk-7b8c98b44-t77l9    0/1     Evicted            0          16m
```

Add the following content to the ClusterExtensionProfile or ExtensionProfile to configure the pod eviction policy:

```
actions:
  evictions:
    - type: "Hard"
      signal: "DiskPressure"
```

- Method 1: Use a ClusterExtensionProfile to configure an eviction policy for a pod with a specified namespace label.

```
apiVersion: bursting.cci.io/v1
kind: ClusterExtensionProfile
metadata:
  name: eviction-cluster-profile
spec:
  actions:
    evictions:
      - type: "Hard"
        signal: "DiskPressure"
  namespaceLabels:
    matchLabels:
        key: value
```

- Method 2: Use an ExtensionProfile to configure an eviction policy for a pod with a specified namespace label.

```
apiVersion: bursting.cci.io/v1
kind: ExtensionProfile
metadata:
  name: eviction-cluster-profile
  namespace: default
spec:
  actions:
    evictions:
      - type: "Hard"
        signal: "DiskPressure"
  objectLabels:
    matchExpressions:
    - key: key2
      operator: In
      values:
      - v2
      - v3
    - key: key3
      operator: Exists
    matchLabels:
      key1: value1
  policy: addOnly
```

5.9 Using a Sidecar Container

5.9.1 Configuring a Sidecar Container

Scenarios

The CCE Cloud Bursting Engine for CCI add-on can automatically inject your desired containers into a pod. Injected containers run as sidecar containers.

For example, if you want to use the add-on to burst the workloads to CCI 2.0 and report collected logs to your log center, you can configure the policy for automatically injecting a log collection container.

There are three ways to configure sidecar containers:

- **Method 1: Using Annotations**
- **Method 2: Using a ClusterExtensionProfile**
- **Method 3: Using an ExtensionProfile**

Prerequisites

If you want to configure sidecar containers using a ClusterExtensionProfile or ExtensionProfile, you need to configure the ClusterExtensionProfile or ExtensionProfile first. For details, see **Configuring a ClusterExtensionProfile or ExtensionProfile**.

Constraints

- Multiple containers (including init containers) can be injected. The injected containers cannot have the same names as other containers in the pod.
- Multiple volumes can be injected. Only volumes of the types supported by CCI can be injected. The injected volumes cannot have the same names as those in **pod.spec.volumes**.
- When a ClusterExtensionProfile or ExtensionProfile is used:
  - An annotation with the key set to **bursting.cci.io/pod-sidecars** cannot be configured in both **actions.podSidecars** and **actions.annotations**.
  - If there is already an annotation with the key set to **bursting.cci.io/pod-sidecars** for the pod matched by the profile, the annotation is overwritten or kept as it is based on the policy of the profile.

Method 1: Using Annotations

You can add annotations to **Pod.Annotations** to inject sidecar containers into pods deployed on CCI.

```
bursting.cci.io/pod-sidecars: '{
"containers":[{"name":"container-sidecar1","image":"busybox:latest","command":["sh","-c","sleep
3600"],"position":"head"}],
"initContainers":[{"command":["sh","-c","sleep
10"],"image":"busybox:latest","name":"initbusybox","position":"tail","resources":{},"volumeMounts":
[{"mountPath":"/tmp/auth","name":"mysecret"}]}],
"volumes":[{"name":"mysecret","secret":{"secretName":"testsecret"}}]}'
```

Method 2: Using a ClusterExtensionProfile

YAML file template for a ClusterExtensionProfile:

```
apiVersion: bursting.cci.io/v1
kind: ClusterExtensionProfile
metadata:
  name: podsidecar-cluster-profile
spec:
  actions:
    podSidecars:
      initContainers:
      - command:
        - sh
        - -c
        - sleep 10
        image: busybox:latest
        name: initbusybox
        position: tail
        resources: {}
        volumeMounts:
        - mountPath: /tmp/auth
          name: mysecret
      volumes:
      - name: mysecret
        secret:
          secretName: testsecret
  namespaceLabels:
    matchLabels:
        key: value
```

Method 3: Using an ExtensionProfile

YAML file template for an ExtensionProfile:

```
apiVersion: bursting.cci.io/v1
kind: ExtensionProfile
metadata:
  name: podsidecar-profile
  namespace: default
spec:
  actions:
    podSidecars:
      containers:
      - command:
        - sh
        - -c
        - sleep 3600
        image: busybox:latest
        name: busybox
        volumeMounts:
        - mountPath: /tmp/auth
          name: mysecret
      volumes:
      - name: mysecret
        secret:
          secretName: testsecret
  objectLabels:
    matchLabels:
      key1: value1
```

Parameter Description

Table 5-14 Key parameters

Parameter	Description
InitContainers	The list of init containers injected on CCI through **pod.spec.initContainers**. The structure of **initContainer** is similar to that of **initContainer** for the pod. **restartPolicy** cannot be configured. **position**, **envFromContainer**, and **volumeMountFromContainer** are added. For details, see **Table 5-15**.
Containers	The list of containers injected on CCI through **pod.spec.containers**. For details, see **Table 5-15**.

Parameter	Description
Volumes	The list of volumes injected on CCI through **pod.spec.volumes**.

Table 5-15 Container parameters

Parameter	Mandatory	Value Type	Description
position	No	String	You can configure the **position** field for a container to specify where the injection is to be performed. If **position** is set to **head**, the container is injected at the beginning. If **position** is set to **tail**, the container is injected at the end. By default, the container is injected at the end. Containers with the same **position** settings are injected in the sequence specified by **actions.podSidecars**.

5.9.2 Replicating Environment Variables and Volume Mount Points of a Service Container

Scenarios

In scenarios such as container log collection and processing, configuration management, and data synchronization, you can transfer the configuration file path through environment variables or mount the volume of the main service container to the sidecar container through the shared volume mount point to implement specific functions. CCI allows you to replicate the environment variables (env) and volume mount points (volumeMounts) of a service container to implement specific functions. This simplifies the operations.

You can replicate environment variables and volume mount points of a service container in the following ways:

- **Method 1: Using Annotations**
- **Method 2: Using a ClusterExtensionProfile**
- **Method 3: Using an ExtensionProfile**

Prerequisites

If you use a ClusterExtensionProfile or ExtensionProfile to replicate environment variables and volume mount points of a service container, you need to configure the sidecar container first. For details, see **Configuring the Sidecar Container**.

Functions

Table 5-16 Functions

Function	Description
Replicate environment variables of the service container when the sidecar container is injected.	• Replicate environment variables by container name and environment variable name. • Replicate environment variables by container name and environment variable name.
Replicate the volume mount points of the service container when the sidecar container is injected.	• Replicate volume mount points by container name and volumeMountName. • Replicate volume mount points by container name and volumeMountName.

Constraints

Table 5-17 Constraints

Field	Constraints
envFromContainer	• **name** in **envFromContainer** cannot be empty. • **name** in **envFromContainer** cannot be duplicate or the same as any **env** name in the **env** configuration of the sidecar container. • Either **name** or **index** in **containerFieldRef** must be configured. • The value of **fieldPath** in **containerFieldRef** must be **name** or start with **env.**.
volumeMountFromContainer	• **volumeMountName** in **volumeMountFromContainer** cannot be empty. • **volumeMountName** in **volumeMountFromContainer** cannot be duplicate or the same as any **name** in the **volumeMounts** configuration of the sidecar container. • Either **name** or **index** in **containerRef** must be configured.

Method 1: Using Annotations

Add the **envFromContainer** and **volumeMountFromContainer** annotations to copy the environment variables and volume mount points of the service container.

```
bursting.cci.io/pod-sidecars: '{"containers":[{"name":"busybox","image":"busybox:latest","command":["sh","-c","sleep 3600"],"resources":{},"envFromContainer":[{"name":"POD_IP","containerFieldRef":{"index":0,"fieldPath":"env.POD_IP"}},{"name":"POD_NAME","containerFieldRef":{"name":"nginx","fieldPath":"env.POD_NAME"}},{"name":"CONTAINER_NAME","containerFieldRef":{"index":0,"fieldPath":"name"}}],"volumeMountFromContainer":[{"volumeMountName":"log-storage","containerRef":{"index":0}},{"volumeMountName":"log-storage1","containerRef":{"name":"nginx"}}]}]}'
```

Method 2: Using a ClusterExtensionProfile

Step 1 Generate a YAML file. Use a ClusterExtensionProfile to configure the environment variables and volume mount points of the sidecar container.

The following is an example YAML file:

```
apiVersion: bursting.cci.io/v1
kind: ClusterExtensionProfile
metadata:
  name: podsidecar-cluster-profile
spec:
  actions:
    podSidecars:
      containers:
      - command:
        - sh
        - -c
        - sleep 3600
        image: busybox:latest
        name: busybox
        envFromContainer:
        - containerFieldRef:
            fieldPath: env.POD_IP
            index: 0
          name: POD_IP
        - containerFieldRef:
            fieldPath: env.POD_NAME
            name: nginx
          name: POD_NAME
        - containerFieldRef:
            fieldPath: name
            index: 0
          name: CONTAINER_NAME
        volumeMountFromContainer:
        - containerRef:
            index: 0
          volumeMountName: log-storage
        - containerRef:
            name: nginx
          volumeMountName: log-storage1
  namespaceLabels:
    matchLabels:
        key: value
```

Step 2 Create the object.

```
kubectl apply -f <podsidecar-cluster-profile>     # YAML file name (Replace it with the actual file name.)
```

Step 3 (Optional) Verify that the object has been created.

```
kubectl get cextp
```

----End

Method 3: Using an ExtensionProfile

Step 1 Generate a YAML file. Use an ExtensionProfile to configure the environment variables and volume mount points of the sidecar container.

The following is an example YAML file:

```
apiVersion: bursting.cci.io/v1
kind: ExtensionProfile
metadata:
  name: podsidecar-profile
spec:
  actions:
    podSidecars:
      containers:
      - command:
        - sh
        - -c
        - sleep 3600
        image: busybox:latest
        name: busybox
        envFromContainer:
        - containerFieldRef:
            fieldPath: env.POD_IP
            index: 0
          name: POD_IP
        - containerFieldRef:
            fieldPath: env.POD_NAME
            name: nginx
          name: POD_NAME
        - containerFieldRef:
            fieldPath: name
            index: 0
          name: CONTAINER_NAME
        volumeMountFromContainer:
        - containerRef:
            index: 0
          volumeMountName: log-storage
        - containerRef:
            name: nginx
          volumeMountName: log-storage1
  objectLabels:
    matchLabels:
      key1: value1
```

Step 2 Create the object.

```
kubectl apply -f <podsidecar-profile>     # YAML file name (Replace it with the actual file name.)
```

Step 3 (Optional) Verify that the object has been created.

```
kubectl get extp
```

----End

Parameter Description

Table 5-18 Key parameters

Parameter	Description
InitContainers	The list of init containers injected on CCI through **pod.spec.initContainers**. The structure of **initContainer** is similar to that of **initContainer** for the pod. **restartPolicy** cannot be configured. **position**, **envFromContainer**, and **volumeMountFromContainer** are added. For details, see **Table 5-19**.
Containers	The list of containers injected on CCI through **pod.spec.containers**. For details, see **Table 5-19**.

Parameter	Description
Volumes	The list of volumes injected on CCI through **pod.spec.volumes**.

Table 5-19 Container parameters

Parameter	Mandatory	Value Type	Description
position	No	String	You can configure the **position** field for a container to specify where the injection is to be performed. If **position** is set to **head**, the container is injected at the beginning. If **position** is set to **tail**, the container is injected at the end. By default, the container is injected at the end. Containers with the same **position** settings are injected in the sequence specified by **actions.podSidecars**.

5.10 Networking

5.10.1 Configuring Networking

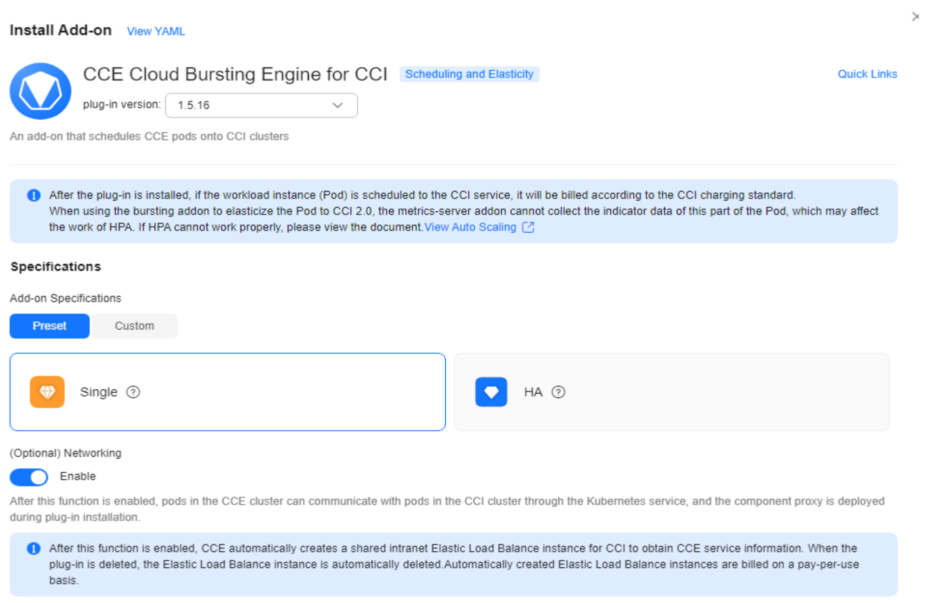
Constraints

- Networking cannot be enabled for CCE clusters that use a shared VPC.
- hostNetwork is not supported.
- If a workload is scheduled from CCE to CCI, only ClusterIP Services can be used for communication between the pods in CCE and those in CCI.
- Networking depends on the startup of sidecar containers. To use this feature, you need to ensure the version of the CCE Cloud Bursting Engine for CCI add-on is 1.5.42 or later.
  - PostStart needs to be configured for service containers.
  - If init container networking is required, you need to enable this option by following the instructions in **Enabling Init Container Networking**.
- You can enable networking and init container networking when the add-on rolling upgrade is complete. (Deliver workloads when the add-on is in the **Running** state again.) Otherwise, the add-on will be abnormal.
- When installing the CCE Cloud Bursting Engine for CCI add-on to schedule workloads to CCI 2.0, you can configure dedicated load balancers for ingresses and Services of the LoadBalancer type. If the add-on version is earlier than 1.5.5, Services of the LoadBalancer type are not supported.
- When you associate the pods with Services or ingresses of the LoadBalancer type:

- Do not re-specify the health check port. After a workload in a CCE cluster is scheduled to CCI, the pods in CCI use different backend ports from those in CCE. If you change the health check port, the health check of some pods will be abnormal.
- If the Services are associated with the same listener of the same load balancer, you need to confirm the health check settings to prevent access exceptions.
- If the pods use a LoadBalancer Service that has a shared load balancer associated, you need to configure the security group of the CCE cluster nodes to allow traffic from **100.125.0.0/16** over the container ports.

Using a Service to Enable Communications Between Pods in CCE and in CCI

Step 1 Install the add-on and enable **Networking**.



CCE Cloud Bursting Engine for CCI

plug-in version: 1.5.16

An add-on that schedules CCE pods onto CCI clusters

After the plug-in is installed, if the workload instance (Pod) is scheduled to the CCI service, it will be billed according to the CCI charging standard.
When using the bursting addon to elasticize the Pod to CCI 2.0, the metrics-server addon cannot collect the indicator data of this part of the Pod, which may affect the work of HPA. If HPA cannot work properly, please view the document.View Auto Scaling

Specifications

Add-on Specifications: Preset

Single

(Optional) Networking: Enable

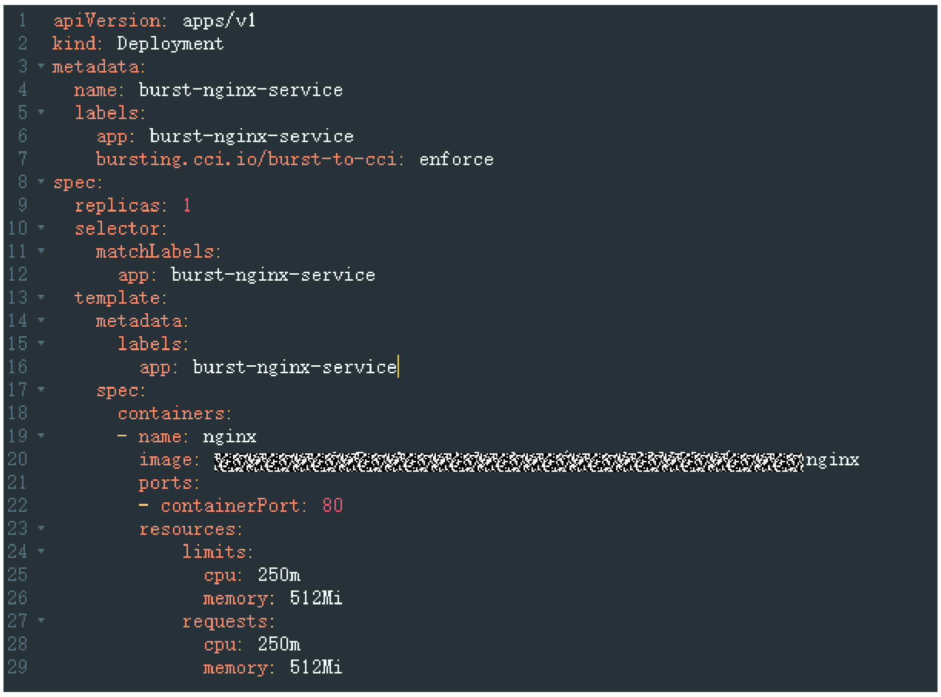
After this function is enabled, pods in the CCE cluster can communicate with pods in the CCI cluster through the Kubernetes service, and the component proxy is deployed during plug-in installation.

After this function is enabled, CCE automatically creates a shared intranet Elastic Load Balance instance for CCI to obtain CCE service information. When the plug-in is deleted, the Elastic Load Balance instance is automatically deleted.Automatically created Elastic Load Balance instances are billed on a pay-per-use basis.

Step 2 On the Network Console, view the load balancer that is automatically created in your account after the add-on installation is successful.

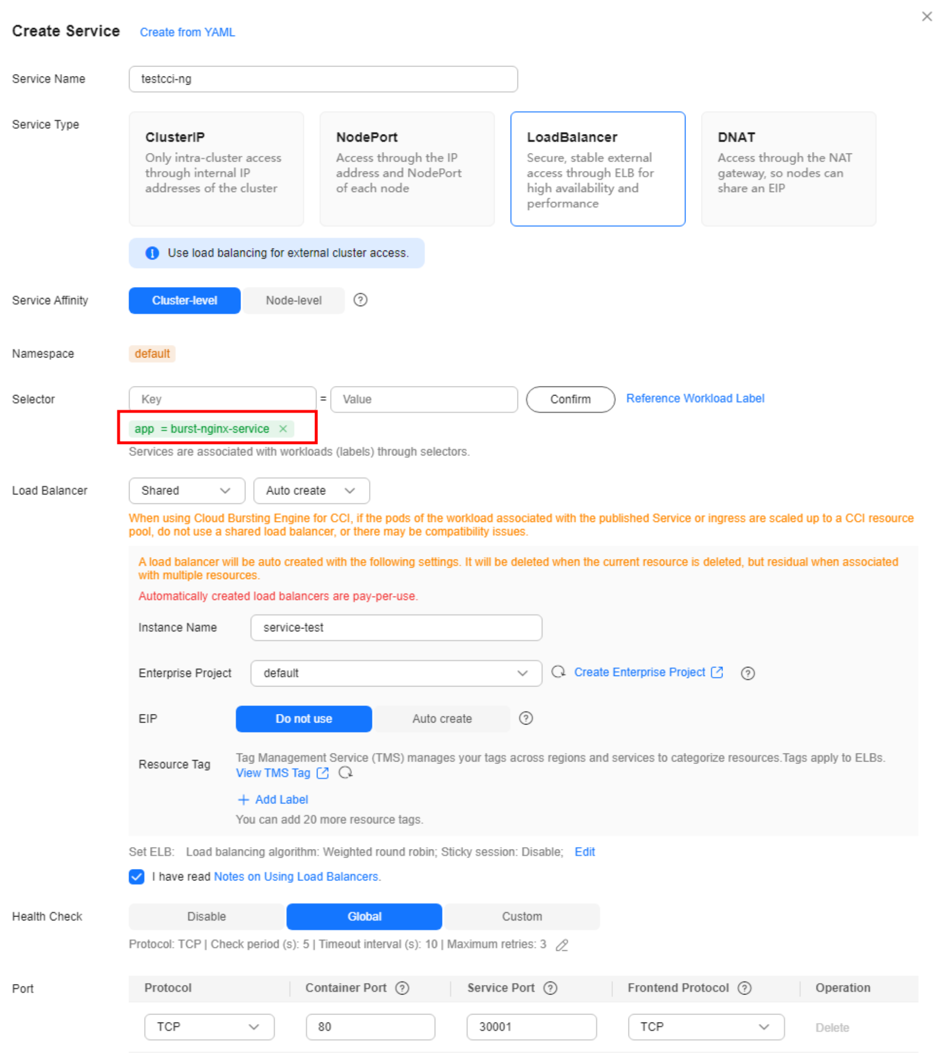
Step 3 Create a pod in CCI and configure a Service to expose the pod.

- To facilitate verification, select the **nginx** image that uses port 80.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: burst-nginx-service
  labels:
    app: burst-nginx-service
    bursting.cci.io/burst-to-cci: enforce
spec:
  replicas: 1
  selector:
    matchLabels:
      app: burst-nginx-service
  template:
    metadata:
      labels:
        app: burst-nginx-service
    spec:
      containers:
      - name: nginx
        image: [redacted]nginx
        ports:
        - containerPort: 80
        resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
```

- Configure a Service that uses the same label as the workload. Select **Auto create** so that a load balancer can be created automatically. This avoids conflicts with the load balancer automatically created for the add-on.


Create Service Create from YAML

Service Name: testcci-ng

Service Type: ClusterIP (Only intra-cluster access through internal IP addresses of the cluster) | NodePort (Access through the IP address and NodePort of each node) | [✓ LoadBalancer (Secure, stable external access through ELB for high availability and performance)] | DNAT (Access through the NAT gateway, so nodes can share an EIP)

Use load balancing for external cluster access.

Service Affinity: [✓ Cluster-level] Node-level

Namespace: default

Selector: Key = Value Confirm Reference Workload Label
app = burst-nginx-service
Services are associated with workloads (labels) through selectors.

Load Balancer: Shared, Auto create
When using Cloud Bursting Engine for CCI, if the pods of the workload associated with the published Service or ingress are scaled up to a CCI resource pool, do not use a shared load balancer, or there may be compatibility issues.

A load balancer will be auto created with the following settings. It will be deleted when the current resource is deleted, but residual when associated with multiple resources.
Automatically created load balancers are pay-per-use.

Instance Name: service-test

Enterprise Project: default Create Enterprise Project

EIP: [✓ Do not use] Auto create

Resource Tag: Tag Management Service (TMS) manages your tags across regions and services to categorize resources.Tags apply to ELBs. View TMS Tag
+ Add Label
You can add 20 more resource tags.

Set ELB: Load balancing algorithm: Weighted round robin; Sticky session: Disable; Edit

[✓] I have read Notes on Using Load Balancers.

Health Check: Disable [✓ Global] Custom
Protocol: TCP | Check period (s): 5 | Timeout interval (s): 10 | Maximum retries: 3

Port:

Protocol	Container Port	Service Port	Frontend Protocol	Operation
TCP	80	30001	TCP	Delete

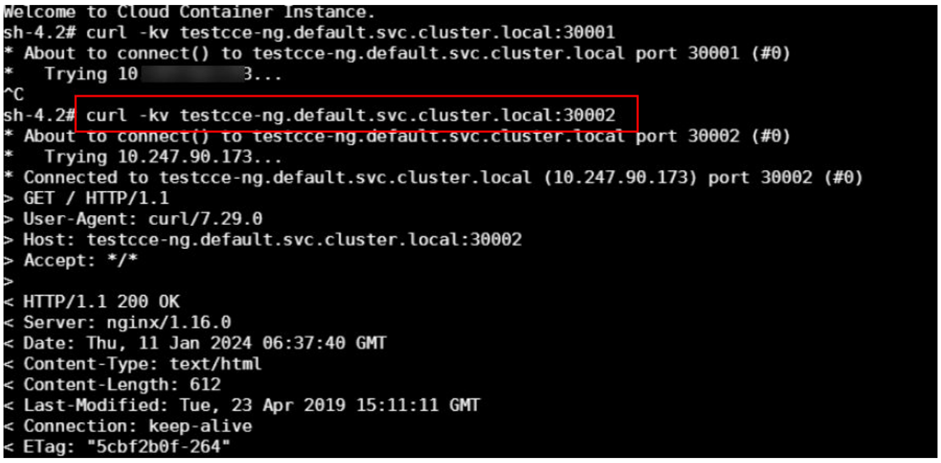
Step 4 Obtain the access mode of the pod on the CCE console.

Step 5 Create a pod in CCE and configure a Service to expose the pod. For details, see **Step 3**. Do not configure the **bursting.cci.io/burst-to-cci** label in **metadata.labels**.

Step 6 Verify network connectivity.

- Enter the pod in CCI, so that this pod can access the Service of the pod in CCE through *<Service-name>.<namespace-name>.svc.cluster.local:port*. Check whether the pod in CCI can access the pod in CCE through the Service.

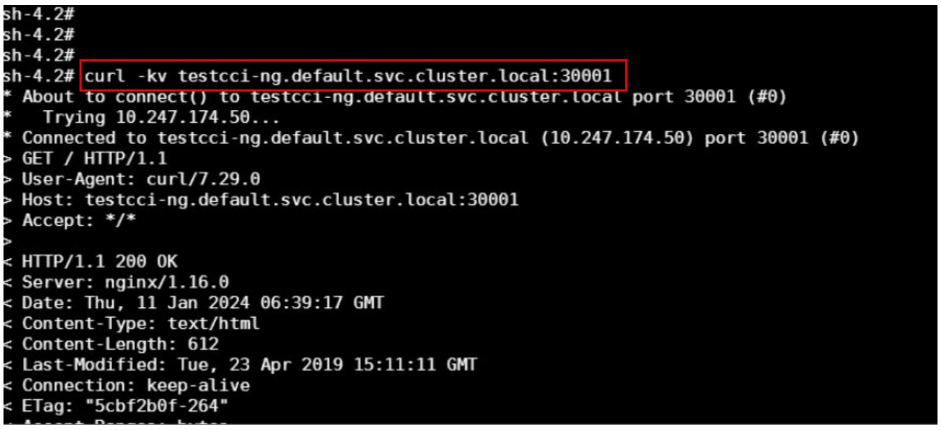
Figure 5-3 Service for accessing the pod in CCE



```
Welcome to Cloud Container Instance.
sh-4.2# curl -kv testcce-ng.default.svc.cluster.local:30001
* About to connect() to testcce-ng.default.svc.cluster.local port 30001 (#0)
*   Trying 10[redacted]3...
^C
sh-4.2# curl -kv testcce-ng.default.svc.cluster.local:30002
* About to connect() to testcce-ng.default.svc.cluster.local port 30002 (#0)
*   Trying 10.247.90.173...
* Connected to testcce-ng.default.svc.cluster.local (10.247.90.173) port 30002 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: testcce-ng.default.svc.cluster.local:30002
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.16.0
< Date: Thu, 11 Jan 2024 06:37:40 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Tue, 23 Apr 2019 15:11:11 GMT
< Connection: keep-alive
< ETag: "5cbf2b0f-264"
```

- Enter the pod in CCE, so that the pod can access the Service of the pod in CCI. Check whether the pod in CCE can access the pod in CCI through the Service.

Figure 5-4 Service for accessing the pod in CCI



```
sh-4.2#
sh-4.2#
sh-4.2#
sh-4.2# curl -kv testcci-ng.default.svc.cluster.local:30001
* About to connect() to testcci-ng.default.svc.cluster.local port 30001 (#0)
*   Trying 10.247.174.50...
* Connected to testcci-ng.default.svc.cluster.local (10.247.174.50) port 30001 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: testcci-ng.default.svc.cluster.local:30001
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.16.0
< Date: Thu, 11 Jan 2024 06:39:17 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Tue, 23 Apr 2019 15:11:11 GMT
< Connection: keep-alive
< ETag: "5cbf2b0f-264"
```

----End

Enabling Init Container Networking

By default, init container networking is disabled. To use this feature, take the following steps after you enable networking for the add-on:

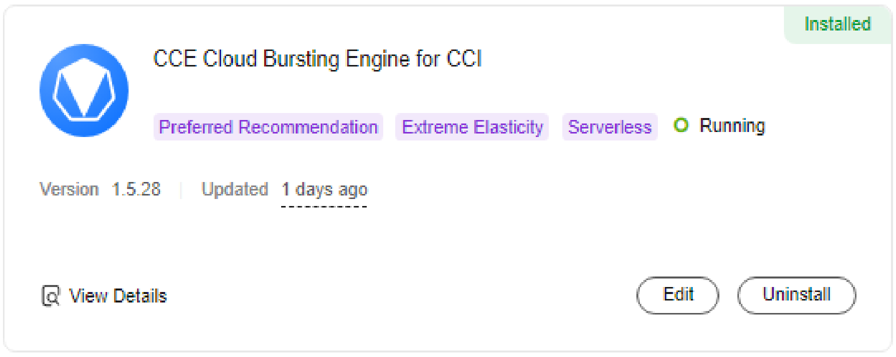
Step 1 Log in to the CCE console.

Step 2 Click the name of the target CCE cluster to go to the cluster **Overview** page.

Step 3 In the navigation pane, choose **Add-ons**.

Step 4 Select the CCE Cloud Bursting Engine for CCI add-on and click **Edit**.

Figure 5-5 CCE Cloud Bursting Engine for CCI



Installed

CCE Cloud Bursting Engine for CCI

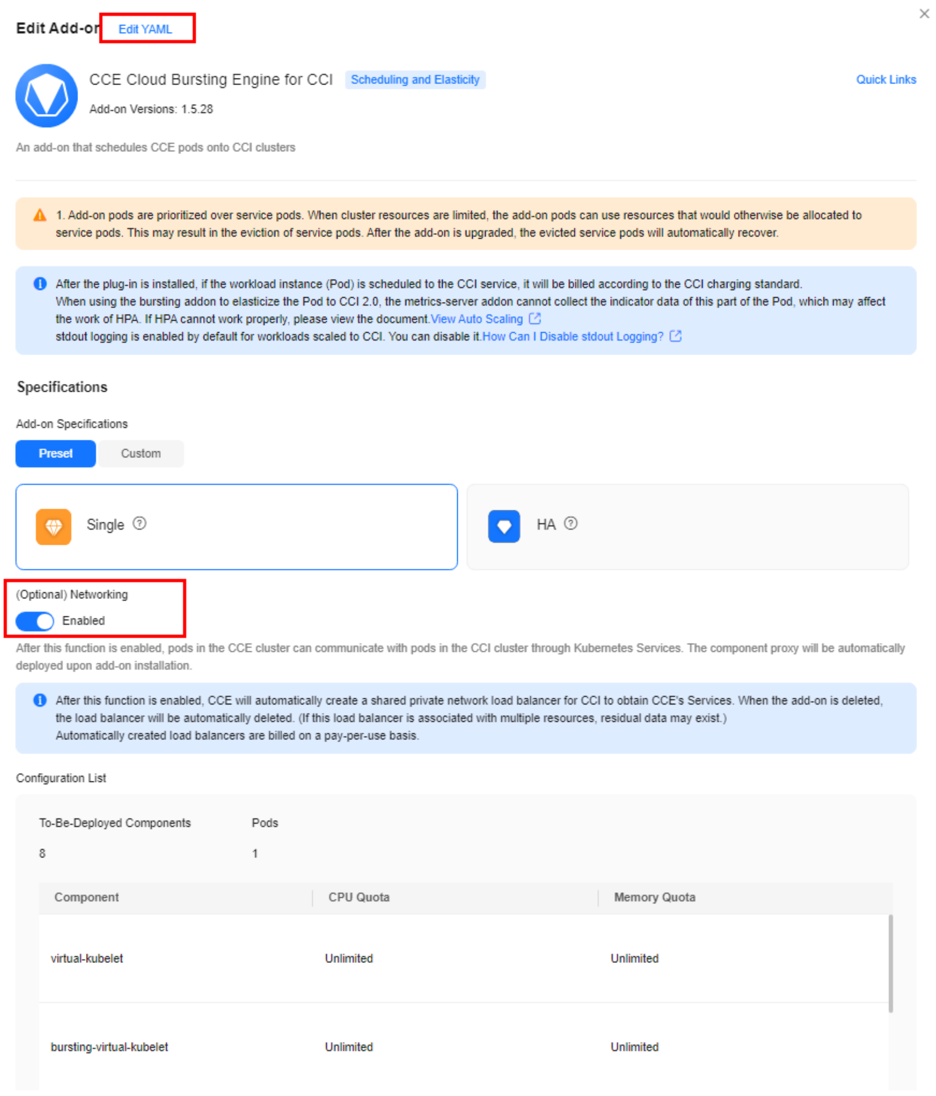
Preferred Recommendation Extreme Elasticity Serverless Running

Version 1.5.28 | Updated 1 days ago

View Details Edit Uninstall

Step 5 Enable **Networking** and then click **Edit YAML**.

Figure 5-6 Editing the add-on



Edit Add-on Edit YAML

CCE Cloud Bursting Engine for CCI Scheduling and Elasticity Quick Links

Add-on Versions: 1.5.28

An add-on that schedules CCE pods onto CCI clusters

1. Add-on pods are prioritized over service pods. When cluster resources are limited, the add-on pods can use resources that would otherwise be allocated to service pods. This may result in the eviction of service pods. After the add-on is upgraded, the evicted service pods will automatically recover.

After the plug-in is installed, if the workload instance (Pod) is scheduled to the CCI service, it will be billed according to the CCI charging standard.
When using the bursting addon to elasticize the Pod to CCI 2.0, the metrics-server addon cannot collect the indicator data of this part of the Pod, which may affect the work of HPA. If HPA cannot work properly, please view the document. View Auto Scaling
stdout logging is enabled by default for workloads scaled to CCI. You can disable it. How Can I Disable stdout Logging?

Specifications

Add-on Specifications

Preset Custom

Single HA

(Optional) Networking

Enabled

After this function is enabled, pods in the CCE cluster can communicate with pods in the CCI cluster through Kubernetes Services. The component proxy will be automatically deployed upon add-on installation.

After this function is enabled, CCE will automatically create a shared private network load balancer for CCI to obtain CCE's Services. When the add-on is deleted, the load balancer will be automatically deleted. (If this load balancer is associated with multiple resources, residual data may exist.)
Automatically created load balancers are billed on a pay-per-use basis.

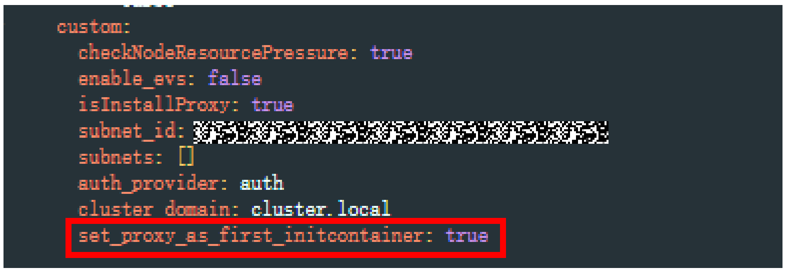
Configuration List

To-Be-Deployed Components	Pods
8	1

Component	CPU Quota	Memory Quota
virtual-kubelet	Unlimited	Unlimited
bursting-virtual-kubelet	Unlimited	Unlimited

Step 6 Set **set_proxy_as_first_initcontainer** to **true**.

Figure 5-7 Modifying the parameter



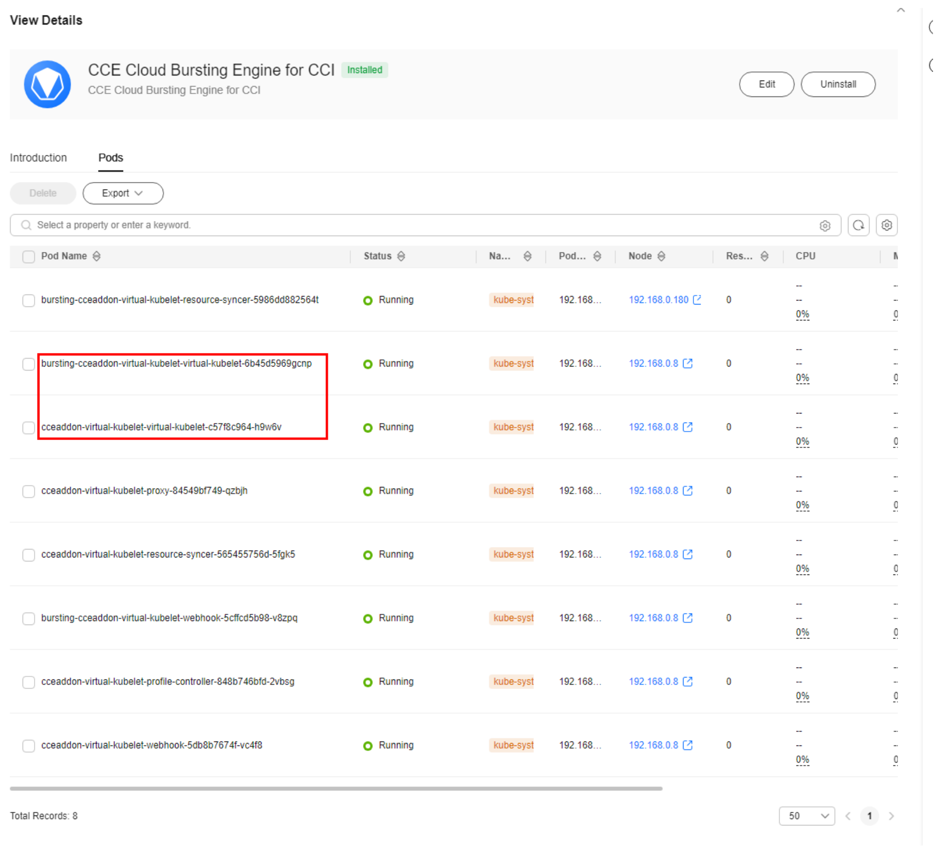
```
custom:
  checkNodeResourcePressure: true
  enable_evs: false
  isInstallProxy: true
  subnet_id: [redacted]
  subnets: []
  auth_provider: auth
  cluster_domain: cluster.local
  set_proxy_as_first_initcontainer: true
```

Table 5-20 Parameter description

Parameter	Mandatory	Type	Description
set_proxy_as_first_initcontainer	No	Bool	Description: Controls whether init container networking is enabled. Value options: • **false** (default): Init container networking is disabled. • **true**: Init container networking is enabled.

Step 7 In the instance list of the add-on, check whether the instance name is **bursting-cceaddon-virtual-kubelet-virtual-kubelet-xxx** and the status is **Running**. If yes, the deployment is complete, and networking is normal.

Figure 5-8 Checking the add-on deployment status


View Details

CCE Cloud Bursting Engine for CCI Installed
CCE Cloud Bursting Engine for CCI

Edit Uninstall

Introduction Pods

Delete Export

Select a property or enter a keyword.

Pod Name	Status	Na...	Pod...	Node	Res...	CPU
bursting-cceaddon-virtual-kubelet-resource-syncer-5986dd882564t	Running	kube-syst	192.168...	192.168.0.180	0	-- -- 0%
bursting-cceaddon-virtual-kubelet-virtual-kubelet-6b45d5969gcnp	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
cceaddon-virtual-kubelet-virtual-kubelet-c57f8c964-h9w6v	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
cceaddon-virtual-kubelet-proxy-84549bf749-qzbjh	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
cceaddon-virtual-kubelet-resource-syncer-565455756d-5fgk5	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
bursting-cceaddon-virtual-kubelet-webhook-5cffcd5b98-v8zpq	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
cceaddon-virtual-kubelet-profile-controller-848b746bfd-2vbsg	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%
cceaddon-virtual-kubelet-webhook-5db8b7674f-vc4f8	Running	kube-syst	192.168...	192.168.0.8	0	-- -- 0%

Total Records: 8

50 1

----End

5.10.2 Specifying Default DNS Servers

If you want to use CCI pods to handle extra workloads, you need to specify DNS servers. The CCE Cloud Bursting Engine for CCI add-on allows you to specify DNS servers, so you do not need to configure the **dnsConfig** field for each pod, reducing network O&M costs.

Step 1 Log in to the CCE console.

Step 2 Click the name of the target CCE cluster to go to the cluster **Overview** page.

Step 3 In the navigation pane, choose **Add-ons**.

Step 4 Select the CCE Cloud Bursting Engine for CCI add-on and click **Edit**.

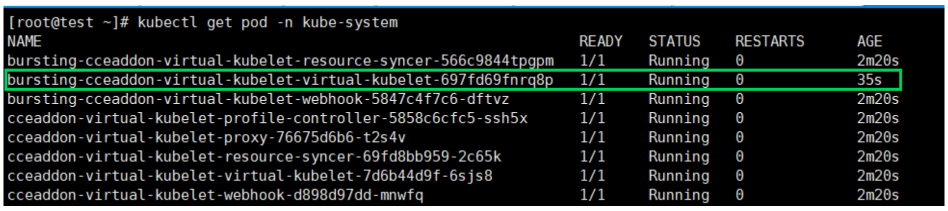
Step 5 Click **Edit YAML**.

Step 6 Log in to a CCE cluster node and edit the YAML file of the add-on.

```
kubectl edit deploy bursting-cceaddon-virtual-kubelet-virtual-kubelet -n kube-system
```

Step 7 Add **--cluster-dns=***x.x.x.x* to the startup parameters and replace *x.x.x.x* with the DNS server address.

Step 8 Save the modification and wait for the bursting-virtual-kubelet workload to restart.



```
[root@test ~]# kubectl get pod -n kube-system
NAME                                                        READY   STATUS    RESTARTS   AGE
bursting-cceaddon-virtual-kubelet-resource-syncer-566c9844tpgpm   1/1     Running   0          2m20s
bursting-cceaddon-virtual-kubelet-virtual-kubelet-697fd69fnrq8p   1/1     Running   0          35s
bursting-cceaddon-virtual-kubelet-webhook-5847c4f7c6-dftvz        1/1     Running   0          2m20s
cceaddon-virtual-kubelet-profile-controller-5858c6cfc5-ssh5x      1/1     Running   0          2m20s
cceaddon-virtual-kubelet-proxy-76675d6b6-t2s4v                    1/1     Running   0          2m20s
cceaddon-virtual-kubelet-resource-syncer-69fd8bb959-2c65k         1/1     Running   0          2m20s
cceaddon-virtual-kubelet-virtual-kubelet-7d6b44d9f-6sjs8          1/1     Running   0          2m20s
cceaddon-virtual-kubelet-webhook-d898d97dd-mnwfq                  1/1     Running   0          2m20s
```

Step 9 Verify the DNS server address. Run the **exec** command to access a container running in CCI and check whether the IP address following **nameserver** in the first line is the address configured for **cluster-dns** in the **/etc/resolv.conf** file.

Table 5-21 Constraints in different scenarios

Scenario	Constraints
There are workloads scheduled to CCI before the DNS server address is specified.	• The DNS server address is only available for new workloads that are scheduled to CCI. • To make the DNS server address available for the workloads that are scheduled to CCI before the modification, you need to rebuild these workloads.
There is a limit for **cluster-dns**.	• You can specify a maximum of three DNS servers in **dnsConfig**. • Ensure that the sum of the **nameserver** value in **cluster-dns** and the **nameservers** value in **spec.dnsConfig** does not exceed 3.

----End

5.10.3 Configuring Subnets

Constraints

- The subnets configured for the namespace are specified by both **spec.custom.subnet_id** and **spec.custom.subnets** configured in the add-on YAML file.
- If you need to delete a subnet, you cannot delete **spec.custom.subnet_id**. You must use another subnet to replace it.

Adding Subnets for a Namespace

By default, the CCI namespace associated with the CCE Cloud Bursting Engine for CCI add-on uses the subnet selected during the add-on installation. If you want to use other subnets, take the following steps:

Step 1 Log in to the CCE console.

Step 2 Click the name of the target CCE cluster to go to the cluster **Overview** page.

Step 3 In the navigation pane, choose **Add-ons**.

Step 4 Select the CCE Cloud Bursting Engine for CCI add-on and click **Edit**.

Step 5 Click **Edit YAML**.

Step 6 Configure the array of subnets in **spec.custom**. **subnetID** indicates the ID of each IPv4 subnet in the VPC of the CCE cluster, expect the default IPv4 subnet.

The following is an example:

```
custom:
  subnet_id: 11c2a970-xxxx-xxxx-xxxx-dfxxxxxxxx7d
  subnets: [
  {
        "subnetID": "efd70252-xxxx-xxxx-xxxx-72xxxxxxxxde"
  },
  {
        "subnetID": "55038f35-xxxx-xxxx-xxxx-ecxxxxxxxx49"
  },
  {
        "subnetID": "99b2a9f5-xxxx-xxxx-xxxx-64xxxxxxxx5a"
  }
  ]
```

Table 5-22 Parameter description

Parameter	Mandatory	Type	Description
subnet_id	Yes	String	Indicates the ID of the default IPv4 subnet selected during the add-on installation. The default subnet quantity is 1.
subnets	No	Array of subnet	Indicates an array of subnets in a namespace.

Table 5-23 Data structure of the subnets field

Parameter	Mandatory	Type	Description
subnetID	Yes	String	ID of each IPv4 subnet to be added

Step 7 Click **Submit**. In the instance list of the add-on, if the instance status is **Running**, the add-on has been deployed and is functioning normally.

----End

5.11 Monitoring

Constraints

For workloads scheduled from CCE to CCI2.0, ensure that the namespace and pod names created in CCE are different from those created in CCI2.0. Otherwise, pod monitoring data in CCI2.0 may be abnormal.

Viewing Pod Monitoring Data on the CCE Console

Step 1 Log in to the CCE console.

Step 2 Click the name of the target CCE cluster to go to the cluster console.

Step 3 In the navigation pane, choose **Monitoring Center**.

Step 4 Click **Enable**. For details, see **Enabling Monitoring Center**.

Step 5 On the **Monitoring Center** page, click the **Pods** tab. The pod list displays all pods. To view the monitoring data of a single pod, click the pod name. The **Overview** tab is displayed. You can click the **Containers** and **Monitoring** tabs to view the corresponding information. For details, see **Pod Monitoring**.

----End

5.12 Auto Scaling

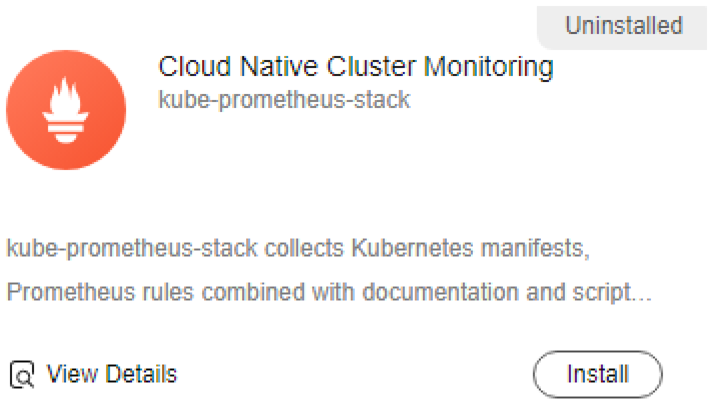
When the CCE Cloud Bursting Engine for CCI add-on is used to schedule workloads to CCI 2.0, the Kubernetes Metrics Server add-on cannot collect metric data of these pods, which may affect the HPA. You can use the Cloud Native Cluster Monitoring add-on to replace Kubernetes Metrics Server add-on to ensure that HPA functions properly.

Procedure

Step 1 Install the Cloud Native Cluster Monitoring add-on.

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Add-ons**.
4. Select the Cloud Native Cluster Monitoring add-on and click **Install**.

Figure 5-9 Installing the Cloud Native Cluster Monitoring add-on


Uninstalled
Cloud Native Cluster Monitoring
kube-prometheus-stack
kube-prometheus-stack collects Kubernetes manifests,
Prometheus rules combined with documentation and script...
View Details
Install

5. **Local Data Storage** must be enabled for **Data Storage Configuration**.
6. Click **Install**.

Step 2 Install the Cloud Native Cluster Monitoring add-on for monitoring.

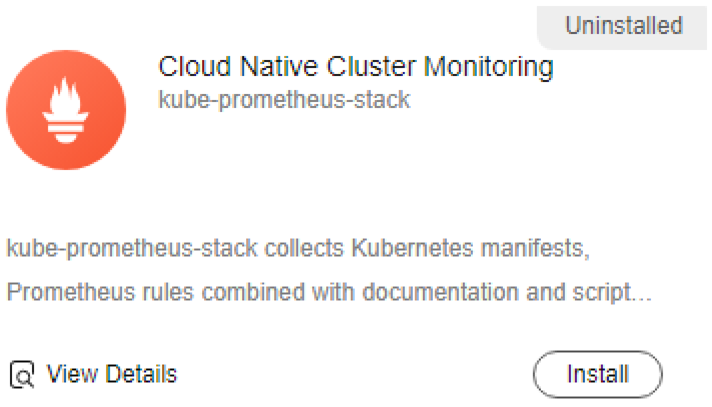
- If the Kubernetes Metrics Server add-on is not installed in the CCE cluster, enable the Metric API. For details, see **Providing Resource Metrics Through the Metrics API**. After the configuration is complete, Prometheus is used to collect system resource metrics.
- If the Kubernetes Metrics Server add-on has been installed in the CCE cluster, use either of the following methods to enable the Metric API:
  - **Method 1: Uninstall the Kubernetes Metrics Server add-on and enable the Metric API.**

    i. Log in to the CCE console.

    ii. Click the name of the target CCE cluster to go to the cluster console.

    iii. In the navigation pane, choose **Add-ons**.

    iv. Select the Kubernetes Metrics Server add-on and click **Uninstall**.


Uninstalled
Cloud Native Cluster Monitoring
kube-prometheus-stack
kube-prometheus-stack collects Kubernetes manifests,
Prometheus rules combined with documentation and script...
View Details
Install

    v. Enable the Metric API. For details, see **Providing Resource Metrics Through the Metrics API**. After the configuration is complete, Prometheus is used to collect system resource metrics.

  - **Method 2: Modify the APIService object.**

    Configuration for updating the APIService object v1beta1.metrics.k8s.io:

```
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  labels:
    app: custom-metrics-apiserver
    release: cceaddon-prometheus
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  groupPriorityMinimum: 100
  insecureSkipTLSVerify: true
  service:
    name: custom-metrics-apiserver
    namespace: monitoring
    port: 443
  version: v1beta1
  versionPriority: 100
```

You can save the object as a file, name it **metrics-apiservice.yaml**, and run the following command:

```
kubectl apply -f metrics-apiservice.yaml
```

Run the **kubectl top pod -n monitoring** command. If the following information is displayed, the Metrics API can be accessed:

```
# kubectl top pod -n monitoring
NAME                                              CPU(cores)   MEMORY(bytes)
......
custom-metrics-apiserver-d4f556ff9-l2j2m                38m         44Mi
......
```

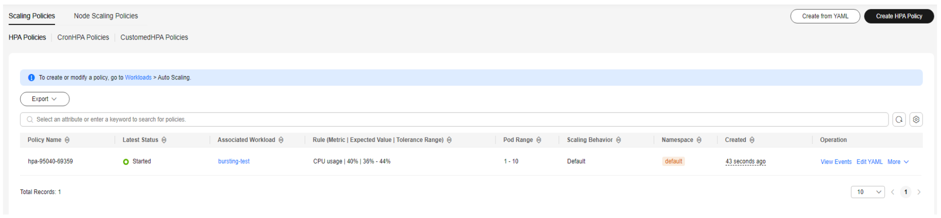
NOTICE

To uninstall the add-on, run the following kubectl command and delete the APIService object first or the add-on cannot be installed due to residual APIService resources.

```
kubectl delete APIService v1beta1.metrics.k8s.io
```

Step 3 Create an HPA policy.

1. Log in to the CCE console.
2. Click the name of the target CCE cluster to go to the cluster console.
3. In the navigation pane, choose **Policies**.
4. Click the **Scaling Policies** tab.
5. In the right corner, click **Create HPA Policy**.

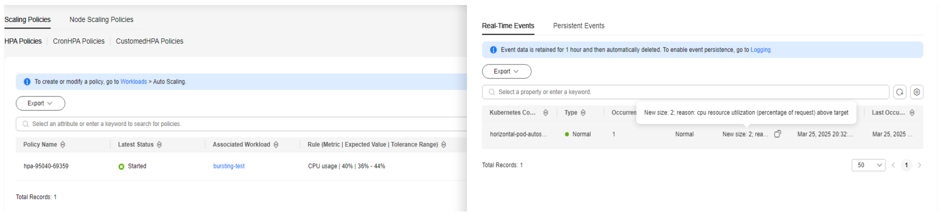


To create or modify a policy, go to Workloads > Auto Scaling.

Policy Name	Latest Status	Associated Workload	Rule (Metric | Expected Value | Tolerance Range)	Pod Range	Scaling Behavior	Namespace	Created	Operation
hpa-95040-69359	Started	bursting-test	CPU usage | 40% | 36% - 44%	1 - 10	Default	default	43 seconds ago	View Events Edit YAML More

Total Records: 1

6. Configure the policy as needed. For example, to increase the CPU usage of a pod, you configure a CPU usage rule, and you can see that the HPA policy scales out the workload.



To create or modify a policy, go to Workloads > Auto Scaling.

Select an attribute or enter a keyword to search for policies.

Policy Name	Latest Status	Associated Workload	Rule (Metric | Expected Value | Tolerance Range)
hpa-95040-69359	Started	bursting-test	CPU usage | 40% | 36% - 44%

Total Records: 1

Event data is retained for 1 hour and then automatically deleted. To enable event persistence, go to Logging

Select a property or enter a keyword.

New size: 2; reason: cpu resource utilization (percentage of request) above target

Kubernetes Co...	Type	Occurren				Last Occu...
horizontal-pod-autos...	Normal	1	Normal	New size: 2; rea...	Mar 25, 2025 20:32:...	Mar 25, 2025 ...

Total Records: 1

50 1

----End

5.13 Virtual Node Configurations

Description

- A cluster can contain one global virtual node and multiple AZ-level virtual nodes. Pods for the workloads scheduled to the global virtual node are randomly scheduled to any supported AZ for CCI, and pods for the workloads scheduled to an AZ-level virtual node are only scheduled to the corresponding AZ for CCI. If you do not want random scheduling on CCI, you need to manually set the global node (bursting-node) to the unschedulable state.
- An AZ-level virtual node has two labels: **failure-domain.beta.kubernetes.io/zone** and **topology.kubernetes.io/zone**. The value of each label is the AZ name, based on which pods can be scheduled.
- The global virtual node cannot be deleted. AZ-level nodes can be dynamically added or deleted based on the ConfigMap configuration.
- vCPU and memory capacities and allocatable vCPUs and memory on the nodes can be configured.
- From v1.5.29 of the add-on, CCI sets the corresponding virtual node to be unschedulable when an AZ becomes faulty. You can view the **status.conditions** information of the virtual node to check the faulty AZ and fault cause. When CCI sets the virtual node to be unschedulable, your existing services may be affected. New services should not be scheduled to the virtual node.

Constraints

- If a key does not contain a period (.), the key name is the AZ name (the key cannot be set to **global**, which indicates a global node). If the key is set to **true**, the virtual node in the corresponding AZ is enabled.
- If the key contains periods (.), the part before the first period is the AZ name, and the part after the first period indicates the node parameters. Currently, **cpu** and **memory** are supported. The values are the same as those for Kubernetes resources (in units such as k, Mi, and Gi). **cpu** and **memory** are optional. If they are not configured, the default values (cpu: 200000; memory: 1600000Gi) are used.
- The key can contain only lowercase letters and must comply with the RFC 1123 subdomain format (same as the node naming rule). The AZ name can contain a maximum of 47 characters.
- A maximum of 20 AZ-level virtual nodes are supported.

- The default AZ fault detection period is 5 minutes. You can set the startup parameter **--az-status-sync-period** of **resource-syncer** to specify the detection period. The minimum period is 1 minute.

Example Configuration

Configure the following ConfigMap in the **kube-system** namespace:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bursting-node
  namespace: kube-system
data:
  global.cpu: 2k          # global indicates the global virtual node, which is created by default and cannot be deleted. You can configure the available vCPUs and memory of the virtual node. If the available vCPUs and memory are not configured, the default values (cpu: 200000, memory: 1600000Gi) are used.
  global.memory: 4000Gi
  region-a: "true"      # If a ConfigMap is added and its key is the AZ name and its value set to true, virtual nodes in the corresponding AZ are automatically created.
  region-a.cpu: 5k     # cpu and memory are optional. If they are not configured, the default values (cpu: 200000, memory: 1600000Gi) are used.
  region-a.memory: 8000Gi
  region-b: "true"
```

The cluster is expected to have three nodes:

- bursting-node
- bursting-node-region-a
- bursting-node-region-b

5.14 Troubleshooting

Symptom 1: Elastic scheduling to CCI is unavailable.

Cause: The subnet where the CCE cluster resides overlaps with 10.247.0.0/16, which is the CIDR block reserved for the Service in the CCI namespace.

Solution: Reset a subnet for the CCE cluster.

Symptom 2: After the CCE Cloud Bursting Engine for CCI add-on is rolled back from 1.5.18 or later to a version earlier than 1.5.18, pods cannot be accessed through the Service.

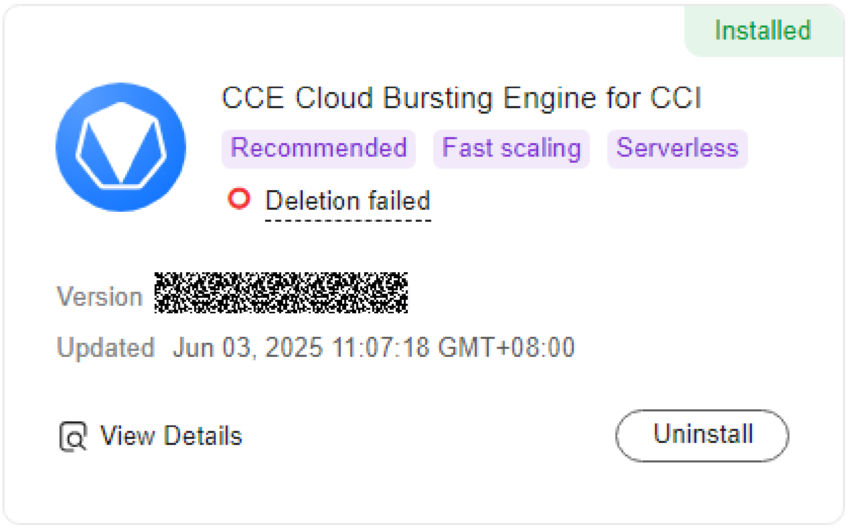
Cause: Once the add-on is upgraded to 1.5.18 or later, the sidecar in each pod that is newly scaled to CCI is incompatible with the add-on of a version earlier than 1.5.18. So, after the add-on is rolled back, the access to the pods is abnormal. If the add-on version is earlier than 1.5.18, pods scaled to CCI are not affected.

Solutions:

- Solution 1: Upgrade the add-on to 1.5.18 or later again.
- Solution 2: Delete the pods that failed to be accessed through the Service and create pods. The new pods scaled to CCI can be accessed normally.

Symptom 3: The add-on cannot be uninstalled.

Scenario: The add-on fails to be uninstalled due to incorrect modification of **swr_addr** and **swr_user**.


Installed

CCE Cloud Bursting Engine for CCI

Recommended Fast scaling Serverless

Deletion failed

Version [redacted]

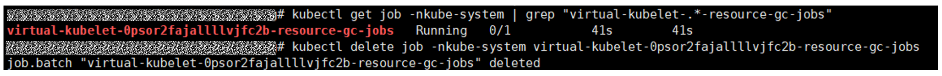
Updated Jun 03, 2025 11:07:18 GMT+08:00

Cause: The add-on uninstallation depends on gc-jobs. If the image fails to be pulled, gc-jobs cannot be executed successfully. As a result, the uninstallation fails.

Solution: Uninstall the add-on again and then delete gc-jobs in sequence.

1. If the add-on fails to be uninstalled, log in to the node where kubectl is configured in the CCE cluster and click **Uninstall**.
2. Run the following commands within 210 seconds.
   a. Delete resource-gc-jobs.

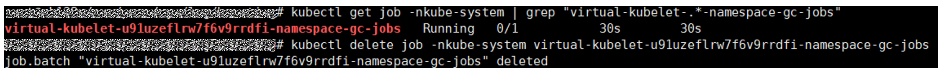
      ```
      kubectl get job -nkube-system | grep "virtual-kubelet-.*-resource-gc-jobs"
      kubectl delete job -nkube-system xxx
      ```

      ```
      [redacted]# kubectl get job -nkube-system | grep "virtual-kubelet-.*-resource-gc-jobs"
      virtual-kubelet-0psor2fajallllvjfc2b-resource-gc-jobs   Running   0/1           41s        41s
      [redacted]# kubectl delete job -nkube-system virtual-kubelet-0psor2fajallllvjfc2b-resource-gc-jobs
      job.batch "virtual-kubelet-0psor2fajallllvjfc2b-resource-gc-jobs" deleted
      ```

   b. Delete namespace-gc-jobs.

      ```
      kubectl get job -nkube-system | grep "virtual-kubelet-.*-namespace-gc-jobs"
      kubectl delete job -nkube-system yyy
      ```

      ```
      [redacted]# kubectl get job -nkube-system | grep "virtual-kubelet-.*-namespace-gc-jobs"
      virtual-kubelet-u91uzeflrw7f6v9rrdfi-namespace-gc-jobs   Running   0/1           30s        30s
      [redacted]# kubectl delete job -nkube-system virtual-kubelet-u91uzeflrw7f6v9rrdfi-namespace-gc-jobs
      job.batch "virtual-kubelet-u91uzeflrw7f6v9rrdfi-namespace-gc-jobs" deleted
      ```

3. For other exceptions, submit a service ticket.

Symptom 4: Service containers that can be accessed through a Service fail to start.

Scenario: Service containers fail to start at the first time because they require a Service for access or PostStart relies on the Service. After the sidecar containers start, the service containers successfully restart.

Cause: The pods on CCI depend on the sidecar containers to access the Service. If the Service synchronization is not complete, the service containers fail to access the Service. After the synchronization is complete, the service containers can start normally.

Solution: Upgrade the add-on to 1.5.28 or later.

6 O&M Management

6.1 Configuring Core Dump

Scenario

Before creating or updating a pod, you can use the **system.cci.io/coredump** annotation to set the **core_pattern** of the kernel, so that you can use generated core dump files to locate the cause of the pod failure.

How to Use

When creating a pod, the core dump configuration (**system.cci.io/coredump : '{"core_pattern": "/var/cores/core.%h.%e.%p.%t," max_size": 2048}'**) can be added to the **annotations** field. If the maximum size is not specified, the default value is **1024**. Core dump will be used in the pod.

Constraints

- The value of the **system.cci.io/core_pattern** field in the request must start with **/var/**, and the file path cannot contain vertical bars (|).
- The configured path must exist in the container. You are advised to mount volumes to store core dump files persistently.

Creating a Pod Using ccictl

You can add an annotation to the workload to control whether to enable core dump for a pod.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  annotations:
    description: ''
  labels: {}
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        vm.cci.io/pod-size-specs: 2.00_4.0
        resource.cci.io/pod-size-specs: 2.00_4.0
        metrics.alpha.kubernetes.io/custom-endpoints: '[{api:'''',path:'''',port:'''',names:''''}]'
        system.cci.io/coredump: "{\"core_pattern\": \"/var/cores/core.%h.%e.%p.%t\",\"max_size\":
1024}"     # Enable core dump.
        log.stdoutcollection.kubernetes.io: '{"collectionContainers": ["container-0"]}'
      labels:
        app: nginx
    spec:
      containers:
        - image: library/nginx:stable-alpine-perl
          name: container-0
          resources:
            limits:
              cpu: 2000m
              memory: 4096Mi
            requests:
              cpu: 2000m
              memory: 4096Mi
          command: []
          lifecycle: {}
      dnsPolicy: ''
      imagePullSecrets:
        - name: imagepull-secret
      dnsConfig: {}
  minReadySeconds: 0
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 0
      maxUnavailable: 1
```

NOTE

system.cci.io/coredump indicates that core dump will be configured for the pod.

6.2 Configuring a Header for Images

Scenario

When an image is being pulled, the **cci.io/registry-headers** annotation can be used to configure the registry and header and provide more information about the image.

How to Use

When creating a pod, **cci.io/registry-headers: "[{\"registry\":\"swr.cn-north-4.myhuaweicloud.com\",\"headers\":{\"key1\":\"value1\",\"key2\":\"value2\"}},{\"registry\":\"swr.cn-north-8.myhuaweicloud.com\",\"headers\":{\"key1\":\"value1\",\"key2\":\"value2\"}}]"** can be added to the **annotations** field to configure the registries and headers for the container in the pod.

Constraints

- The registry and header in the **cci.io/registry-headers** field in the request must be verified.

- The **cci.io/registry-headers** field in the request is obtained from **annotations**.

Creating a Pod Using ccictl

You can add an annotation to the workload to control whether to configure the registry and header for a pod.

```
apiVersion: cci/v2
kind: Deployment
metadata:
  annotations:
    description: ''
  labels: {}
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        vm.cci.io/pod-size-specs: 2.00_4.0
        resource.cci.io/pod-size-specs: 2.00_4.0
        metrics.alpha.kubernetes.io/custom-endpoints: '[{api:'''',path:'''',port:'''',names:''''}]'
        cci.io/registry-headers: "[{\"registry\":\"{Image repository address}\",\"headers\":
{\"key1\":\"value1\",\"key2\":\"value2\"}},{\"registry\":\"{Image repository address}\",\"headers\":
{\"key1\":\"value1\",\"key2\":\"value2\"}}]"     # Enable registry-headers.
        log.stdoutcollection.kubernetes.io: '{"collectionContainers": ["container-0"]}'
      labels:
        app: nginx
    spec:
      containers:
        - image: library/nginx:stable-alpine-perl
          name: container-0
          resources:
            limits:
              cpu: 2000m
              memory: 4096Mi
            requests:
              cpu: 2000m
              memory: 4096Mi
          command: []
          lifecycle: {}
      dnsPolicy: ''
      imagePullSecrets:
        - name: imagepull-secret
      dnsConfig: {}
  minReadySeconds: 0
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 0
      maxUnavailable: 1
```

NOTE

cci.io/registry-headers indicates that the registry and header will be configured for the pod.

7 Cost Tag Management

Scenario

You can configure cost tags to classify and track pod costs.

Tag Naming Rules

- Each tag consists of a key-value pair. Example: `pod-tag.cci.io/<tag-key>:tag-value`
- Each pod can have a maximum of 20 tags.
- A tag key must be unique for each pod, and each tag key can have only one tag value.
- A tag consists of a tag key and a tag value. **Table 7-1** lists the tag key and value requirements.

Table 7-1 Tag naming rules

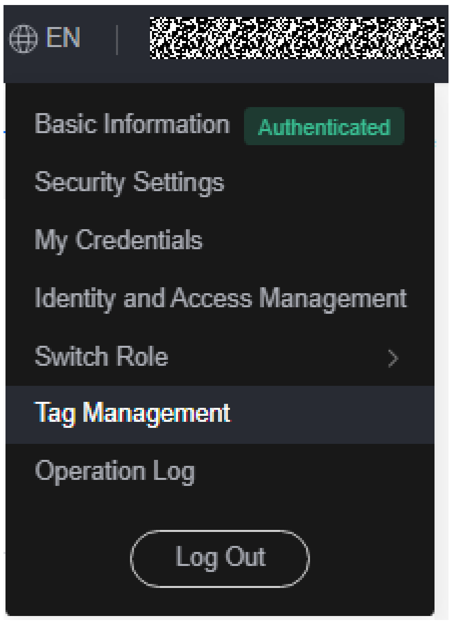
Item	Rules	Example
Tag key	• Cannot be empty. • Must be unique for each pod. • Can contain a maximum of 128 characters. • Can contain letters, digits, spaces, and special characters _.:=+-@. • Cannot start or end with a space.	Organization
Tag value	• Can contain a maximum of 255 characters. • Can contain letters, digits, spaces, and special characters _.:=+-@/.	Apache

Procedure

Step 1 Log in to the management console.

Step 2 In the upper right corner of the page, click the username and select **Tag Management** from the drop-down list.

Figure 7-1 Tag Management



Step 3 Create a predefined tag. In the navigation pane, choose **Predefined Tags**. Click **Create Tag**, enter the tag key and value, and click **OK**.

Step 4 In the top navigation bar, choose **Billing** > **Cost Center**.

Step 5 In the navigation pane, choose **Cost Organization** > **Cost Tags**. On the **Cost Tags** page, select the tag created in **Step 3**, click **Activate**, and then click **OK**.

Step 6 Wait until the tag status changes to **Activated**.

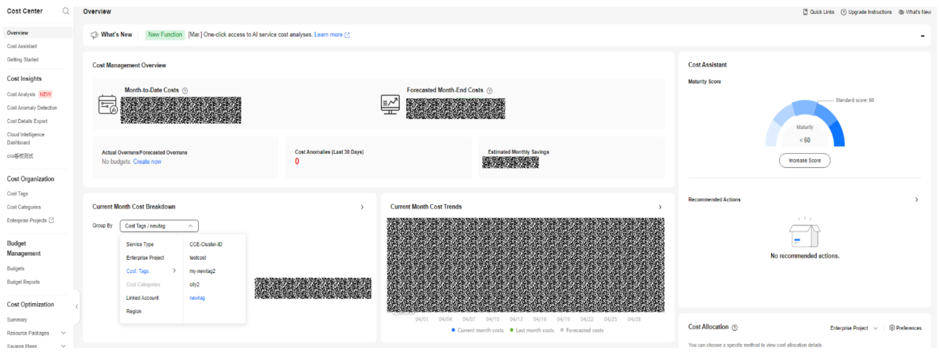
Step 7 Add the following fields to the **Podtemplate.metadata.labels** file when creating a pod in CCI. The following is an example YAML file:

```
kind: Deployment
apiVersion: cci/v2
metadata:
  name: tag-example
  namespace: bursting-d0089910820250427-2031
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tag-example
  template:
    metadata:
      labels:
        app: tag-example
        pod-tag.cci.io/newtag: tag-example-1   # pod-tag.cci.io/<custom-tag-key>: custom-tag-value
    spec:
      containers:
        - name: deploy-example
          image: swr.region-id.domain.com/org-name/image-name:tag
          env:
            - name: ENV1
              value: 'false'
            - name: ENV2
```

```
              value: xxx
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 500m
              memory: 1Gi
      dnsPolicy: Default
      imagePullSecrets:
        - name: imagepull-secret
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 100%
```

Step 8 In the top navigation bar, choose **Billing** > **Cost Center**. In the navigation pane, choose **Overview**. In the **Current Month Cost Breakdown** area, select **Cost Tags** and then the created tag to view the cost details.

Figure 7-2 Current Month Cost Breakdown



What's New New Function [Mar.] One-click access to AI service cost analyses. Learn more

Month-to-Date Costs [redacted]

Forecasted Month-End Costs [redacted]

Actual Overruns/Forecasted Overruns: No budgets. Create now

Cost Anomalies (Last 30 Days): 0

Estimated Monthly Savings [redacted]

Group By: Cost Tags / newtag

[redacted]

No recommended actions.

You can choose a specific method to view cost allocation details.

----End

8 Auditing

8.1 CCI Operations Supported by CTS

Cloud Trace Service (CTS) records operations on cloud service resources, allowing you to query, audit, and backtrack the resource operation requests initiated from the CCI console or open APIs as well as responses to the requests.

Table 8-1 CCI operations that can be recorded by CTS

Operation	Trace Name
Creating a Service	createService
Deleting a Service	deleteService
Replacing a Service	replaceService
Updating a Service	updateService
Creating a Deployment	createDeployment
Deleting a Deployment	deleteDeployment
Replacing a Deployment in a specified namespace	replaceDeployment
Updating a Deployment in a specified namespace	updateDeployment
Creating a namespace	createNamespace
Deleting a namespace	deleteNamespace
Creating a pod	createPod
Updating a pod	updatePod
Replacing a pod	replacePod
Deleting a pod	deletePod

Operation	Trace Name
Replacing the ObservabilityConfigura-tion	replaceObservabilityconfiguration
Creating a ConfigMap	createConfigmap
Updating a ConfigMap	updateConfigmap
Replacing a ConfigMap	replaceConfigmap
Deleting a ConfigMap	deleteConfigmap
Creating a secret	createSecret
Updating a secret	updateSecret
Replacing a secret	replaceSecret
Deleting a secret	deleteSecret
Deleting a network	deleteNetwork
Creating a network	createNetwork
Updating a network	updateNetwork
Replacing a network	replaceNetwork
Creating a PVC	createPersistentvolumeclaim
Replacing a PVC	replacePersistentvolumeclaim
Updating a PVC	updatePersistentvolumeclaim
Deleting a PVC	deletePersistentvolumeclaim
Creating an image snapshot	createImagesnapshot
Deleting an image snapshot	deleteImagesnapshot
Querying an image snapshot	getImagesnapshot
Querying all image snapshots in a specified namespace	listImagesnapshots
Creating a PV	createPV
Updating a PV	updatePV
Replacing a PV	replacePV
Deleting a PV	deletePV
Querying a PV	getPV
Listing all PVs in a specified namespace	listPVs
Querying a ReplicaSet	getReplicaset

Operation	Trace Name
Listing all ReplicaSets in a specified namespace	listReplicasets
Creating an HPA policy	createHPA
Deleting an HPA policy	deleteHPA
Updating an HPA policy	updateHPA
Replacing an HPA policy	replaceHPA
Querying an HPA policy	getHPA
Listing all HPA policies in a specified namespace	listHPAs
Creating a PoolBinding	createPoolbinding
Deleting a PoolBinding	deletePoolbinding
Querying a PoolBinding	getPoolbinding
Listing all PoolBindings in a specified namespace	listPoolbindings

8.2 Viewing a Trace

Scenarios

Once enabled, CTS starts recording operations on CCI resources. Operation records of the last seven days can be viewed on the CTS console.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner to select the desired region.

Step 3 Click **Service List** and choose **Management & Governance** > **Cloud Trace Service**.

Step 4 In the navigation pane, choose **Trace List**.

Step 5 Specify the filters used for querying traces. You can query traces using a combination of the following filters:

- **Trace Type**, **Trace Source**, **Resource Type**, and **Search By**

  Select the desired filters from the drop-down lists. Select **CCI** from the **Trace Source** drop-down list.

  If you select **Trace name** for **Search By**, you need to select a trace name.

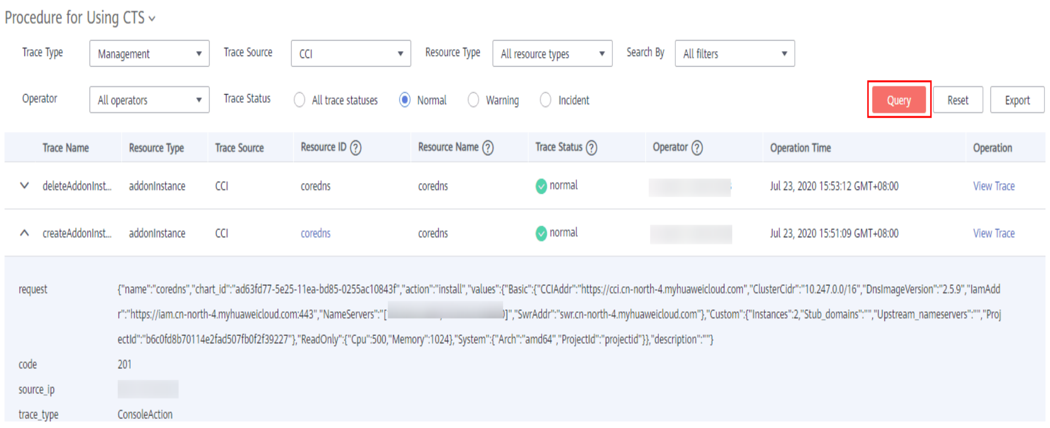
  If you select **Resource ID** for **Search By**, you need to select or enter a resource ID.

If you select **Resource name**, you need to select a resource name.

- **Operator**: Select an operator (a user not a tenant).
- **Trace Status**: Available options include **All trace statuses**, **Normal**, **Warning**, and **Incident**. You can select only one of them.
- Start time and end time: You can specify the time period in which to query traces.

Step 6 Click on the left of a trace to expand its details.

Figure 8-1 Expanding trace details



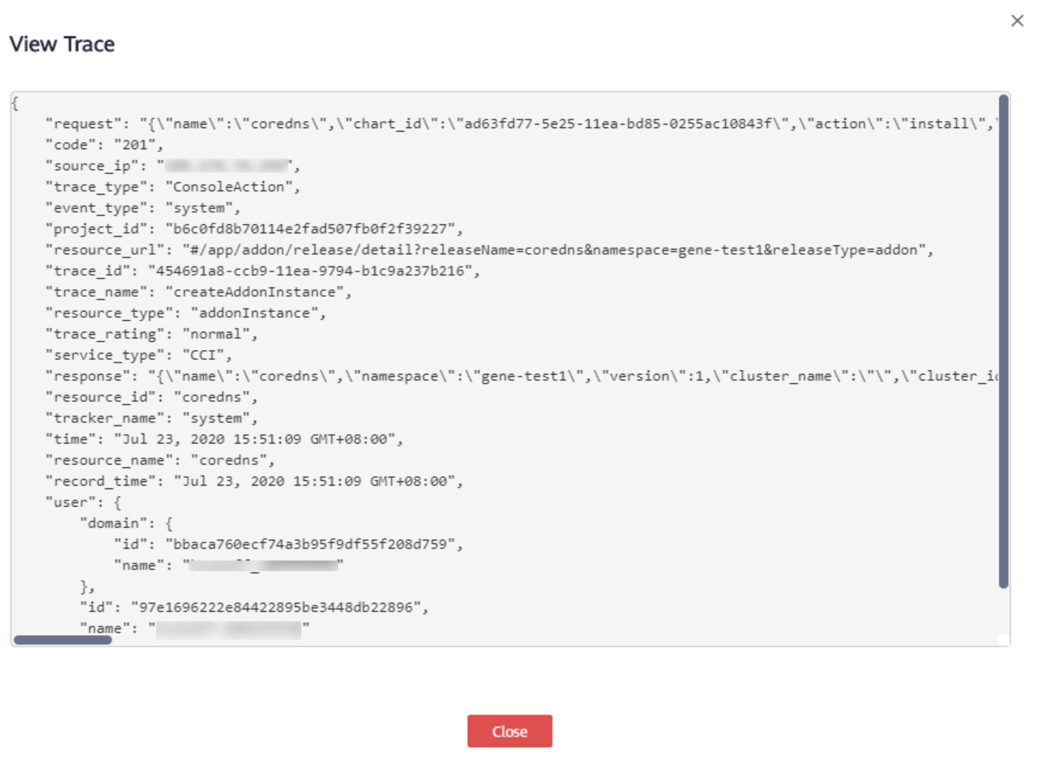
Trace Type: Management
Trace Source: CCI
Resource Type: All resource types
Search By: All filters
Operator: All operators
Trace Status: All trace statuses [✓ Normal] Warning Incident

	Trace Name	Resource Type	Trace Source	Resource ID	Resource Name	Trace Status	Operator	Operation Time	Operation
∨	deleteAddonInst...	addonInstance	CCI	coredns	coredns	normal	[redacted]	Jul 23, 2020 15:53:12 GMT+08:00	View Trace
∧	createAddonInst...	addonInstance	CCI	coredns	coredns	normal	[redacted]	Jul 23, 2020 15:51:09 GMT+08:00	View Trace

```
request     {"name":"coredns","chart_id":"ad63fd77-5e25-11ea-bd85-0255ac10843f","action":"install","values":{"Basic":{"CCIAddr":"https://cci.cn-north-4.myhuaweicloud.com","ClusterCidr":"10.247.0.0/16","DnsImageVersion":"2.5.9","IamAdd
            r":"https://iam.cn-north-4.myhuaweicloud.com:443","NameServers":"[[redacted])]","SwrAddr":"swr.cn-north-4.myhuaweicloud.com"},"Custom":{"Instances":2,"Stub_domains":"","Upstream_nameservers":"","Proj
            ectId":"b6c0fd8b70114e2fad507fb0f2f39227"},"ReadOnly":{"Cpu":500,"Memory":1024},"System":{"Arch":"amd64","ProjectId":"projectid"}},"description":""}
code        201
source_ip   [redacted]
trace_type  ConsoleAction
```

Step 7 Click **View Trace** in the **Operation** column. In the displayed dialog box shown in **Figure 8-2**, the trace structure details are displayed.

Figure 8-2 Viewing a trace



```
{
    "request": "{\"name\":\"coredns\",\"chart_id\":\"ad63fd77-5e25-11ea-bd85-0255ac10843f\",\"action\":\"install\",\
    "code": "201",
    "source_ip": "[redacted]",
    "trace_type": "ConsoleAction",
    "event_type": "system",
    "project_id": "b6c0fd8b70114e2fad507fb0f2f39227",
    "resource_url": "#/app/addon/release/detail?releaseName=coredns&namespace=gene-test1&releaseType=addon",
    "trace_id": "454691a8-ccb9-11ea-9794-b1c9a237b216",
    "trace_name": "createAddonInstance",
    "resource_type": "addonInstance",
    "trace_rating": "normal",
    "service_type": "CCI",
    "response": "{\"name\":\"coredns\",\"namespace\":\"gene-test1\",\"version\":1,\"cluster_name\":\"\",\"cluster_i
    "resource_id": "coredns",
    "tracker_name": "system",
    "time": "Jul 23, 2020 15:51:09 GMT+08:00",
    "resource_name": "coredns",
    "record_time": "Jul 23, 2020 15:51:09 GMT+08:00",
    "user": {
        "domain": {
            "id": "bbaca760ecf74a3b95f9df55f208d759",
            "name": "[redacted]"
        },
        "id": "97e1696222e84422895be3448db22896",
        "name": "[redacted]"
```

----End