

# Cloud Container Instance

## FAQs

**Issue**            01  
**Date**             2024-11-05



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

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

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

---

# Contents

---

<b>1 Product Consulting</b>	<b>1</b>
1.1 What Are the Public IP Addresses Contained in the Open-Source Third-Party Dependencies of CCI?	1
<b>2 Basic Concept FAQs</b>	<b>2</b>
2.1 What Is CCI?	2
2.2 What Are the Differences Between Cloud Container Instance and Cloud Container Engine?	3
2.3 What Is an Environment Variable?	6
2.4 What Is a Service?	6
2.5 What Is Mcore?	6
2.6 What Are the Relationships Between Images, Containers, and Workloads?	6
2.7 What Are Kata Containers?	7
2.8 Can kubectl Be Used to Manage Container Instances?	7
2.9 What Are Core-Hours in CCI Resource Packages?	7
<b>3 Workload Abnormalities</b>	<b>9</b>
3.1 Locating Method	9
3.2 What Do I Do If an Event Indicating That the Image Failed to Be Pulled Occurs?	9
3.3 What Do I Do If an Event Indicating That the Container Failed to Be Restarted Occurs?	11
<b>4 Container Workload FAQs</b>	<b>15</b>
4.1 Why Service Performance Does Not Meet the Expectation?	15
4.2 How Do I Set the Quantity of Instances (Pods)?	16
4.3 How Do I Check My Resource Quotas?	16
4.4 How Do I Set Probes for a Workload?	17
4.5 How Do I Configure an Auto Scaling Policy?	17
4.6 What Do I Do If the Workload Created from the sample Image Fails to Run?	17
4.7 How Do I View Pods After I Call the API to Delete a Deployment?	18
4.8 Why an Error Is Reported When a GPU-Related Operation Is Performed on the Container Entered by Using exec?	18
4.9 Can I Start a Container in Privileged Mode When Running the systemctl Command in a Container in a CCI Cluster?	19
4.10 Why Does the Intel oneAPI Toolkit Fail to Run VASP Tasks Occasionally?	19
4.11 Why Are Pods Evicted?	19
4.12 Why Is the Workload Web-Terminal Not Displayed on the Console?	19
4.13 Why Are Fees Continuously Deducted After I Delete a Workload?	19

<b>5 Image Repository FAQs.....</b>	<b>21</b>
5.1 Can I Export Public Images?.....	21
5.2 How Do I Create a Container Image?.....	21
5.3 How Do I Upload Images?.....	22
5.4 Does CCI Provide Base Container Images for Download?.....	22
5.5 Does CCI Administrator Have the Permission to Upload Image Packages?.....	22
5.6 What Permissions Are Required for Uploading Image Packages for CCI?.....	22
5.7 What Do I Do If Authentication Is Required During Image Push?.....	23
<b>6 Network Management FAQs.....</b>	<b>24</b>
6.1 How Do I View the VPC CIDR Block?.....	24
6.2 Does CCI Support Load Balancing?.....	24
6.3 How Do I Configure the DNS Service on CCI?.....	25
6.4 Does CCI Support InfiniBand (IB) Networks?.....	25
6.5 How Do I Access a Container from a Public Network?.....	26
6.6 How Do I Access a Public Network from a Container?.....	26
6.7 What Do I Do If Access to a Workload from a Public Network Fails?.....	26
6.8 What Do I Do If Error 504 Is Reported When I Access a Workload?.....	27
6.9 What Do I Do If the Connection Timed Out?.....	28
<b>7 Storage Management FAQs.....</b>	<b>29</b>
7.1 Which Cloud Storage Types Are Supported and Which Types Require Backup?.....	29
7.2 How Do I Use Cloud Storage?.....	29
7.3 Where Is the Data Generated by a Container Stored If No Cloud Storage Is Mounted?.....	29
7.4 Why Volume Mounting Failure Is Reported When a Job Pod Completes Its Task?.....	30
<b>8 Log Collection.....</b>	<b>31</b>
8.1 Why are logs duplicated or lost?.....	31
<b>9 Account.....</b>	<b>32</b>
9.1 Why Is a Message Displayed Indicating That My Account Is In Arrears Even If My Account Has Balance?.....	32
9.2 Why Resources Cannot Be Deleted?.....	32

# 1 Product Consulting

---

## 1.1 What Are the Public IP Addresses Contained in the Open-Source Third-Party Dependencies of CCI?

CCI provides a serverless container engine, allowing you to run containers without creating or managing clusters. CCI has two open-source third-party dependencies: k8s.io/kubernetes and go.etcd.io/etcd. In these dependencies, there are public IP addresses like "http://metadata.google.internal.", "https://www.googleapis.com" and "https://storage.googleapis.com". CCI does not use or exchange data with these public IP addresses.

k8s.io/kubernetes open-source community: <https://github.com/kubernetes/kubernetes>

go.etcd.io/etcd open-source community: <https://github.com/etcd-io/etcd>

# 2 Basic Concept FAQs

---

## 2.1 What Is CCI?

Cloud Container Instance (CCI) is a serverless container engine that allows you to run containers without creating or managing server clusters.

With the serverless architecture, you can focus on building and operating applications without having to create or manage servers, or worrying about server health. All you have to do is to specify resource requirements (such as the required CPU cores and memory space). This gives you a more focused approach to business needs and helps you reduce management and maintenance costs. Traditionally, to run containerized workloads using Kubernetes, you need to create a Kubernetes server cluster first. That is not the case with CCI. Under the serverless architecture, CCI allows you to directly create and use containerized workloads by using the console, kubectl, or Kubernetes APIs, and pay only for the resources consumed by these workloads.

CCI provides the following functions:

- Automated continuous delivery (CD)  
Verifies every container image change by running the new container images in just a few clicks.
- Full hosting of workloads during runtime  
Provides hosting for Deployments to ensure stable running.
- Fast auto scaling  
Allows you to customize auto scaling policies and scales resources automatically within 1s.
- High Availability (HA) assurance for applications  
Allows concurrent running of multiple instances and provides global load balancing.
- Monitoring of container status  
Provides health checks for containers and monitors container metrics in real time.
- Persistent data storage

Supports mounting of network storage volumes to persist service data.

## 2.2 What Are the Differences Between Cloud Container Instance and Cloud Container Engine?

Huawei Cloud provides two CNCF-certified Kubernetes services that feature high performance, high availability, and high security: Cloud Container Engine (CCE) and Cloud Container Instance (CCI).

**CCE** provides highly scalable, high-performance, enterprise-class Kubernetes clusters to run containers. CCE is a one-stop container platform that provides full-stack container services from Kubernetes cluster management, lifecycle management of containerized applications, application service mesh, and Helm charts to add-on management, application scheduling, and monitoring and O&M. With CCE, you can easily deploy, manage, and scale containerized applications on Huawei Cloud.

For details, see [What Is Cloud Container Engine?](#)

**CCI** is a serverless container engine that allows you to run containers without creating and managing servers or clusters. With CCI, you only need to manage containerized services running on Kubernetes. You can quickly create and run containerized workloads on CCI without managing clusters and servers.

With the serverless architecture, you can focus on building and operating applications without having to create or manage servers, not to mention the issues caused by abnormal server running. All you have to do is to specify resource requirements (on CPU and memory, for example). This gives you a more focused approach to business needs and helps you reduce management and maintenance costs. Traditionally, to run containerized workloads using Kubernetes, you need to create a Kubernetes cluster first.

For details, see [What Is CCI?](#)

## Description

**Table 2-1** Introduction to CCE and CCI

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
<p>CCE provides highly scalable, high-performance, enterprise-class Kubernetes clusters and supports Docker containers. CCE is a one-stop container platform that provides full-stack container services from Kubernetes cluster management, lifecycle management of containerized applications, application service mesh, and Helm charts to add-on management, application scheduling, and monitoring and O&amp;M. With CCE, you can easily deploy, manage, and scale containerized applications on Huawei Cloud.</p> <p>For details, see <a href="#">What Is Cloud Container Engine?</a></p>	<p>CCI is a serverless container engine that allows you to run containers without creating and managing server clusters. With CCI, you only need to manage containerized services running on Kubernetes. You can quickly create and run containerized workloads on CCI without managing clusters and servers.</p> <p>With the serverless architecture, you can focus on building and operating applications without having to create or manage servers, not to mention the issues caused by abnormal server running. All you have to do is to specify resource requirements (on CPU and memory, for example). This gives you a more focused approach to business needs and helps you reduce management and maintenance costs. Traditionally, to run containerized workloads using Kubernetes, you need to create a Kubernetes cluster first.</p>

## Creation Mode

**Table 2-2** Different creation modes

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
<p>CCE is a hosted Kubernetes service for container management. It allows you to create native Kubernetes clusters with just a few clicks.</p> <p>You need to create clusters and nodes on the console, but you do not need to manage master nodes.</p>	<p>CCI provides a serverless container engine. When deploying containers on Huawei Cloud, you do not need to purchase and manage ECSs, eliminating the need for O&amp;M and management.</p> <p>You can start applications without the need to create clusters, master nodes, or worker nodes.</p>



## Billing

**Table 2-3** Different billing modes

Aspect	Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Pricing	Resources are charged.	CPU (core-hour) and memory (GiB-hour)
Billing mode	Yearly/Monthly and pay-per-use	Pay-per-use
Minimum pricing unit	Hour	Second

## Application Scenarios

**Table 2-4** Different application scenarios

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Applicable to all scenarios. Generally, CCE runs large-scale and long-term stable applications, such as e-commerce, service middle-end, and IT system.	Applicable to batch computing, high-performance computing, burst scale-out, and CI/CD tests.

## Cluster Creation

**Table 2-5** Different cluster creation modes

Cloud Container Engine (CCE)	Cloud Container Instance (CCI)
Configure basic information (name, Region, networking, and compute) > Create a worker node > Configure the cluster > Create a workload.	Configure basic information (name, Region, networking, and compute) > Create a workload.

## Cooperation Between CCE and CCI

By installing the virtual-kubelet add-on, you can use CCI to deploy pods for your Deployments, StatefulSets, and jobs on CCE when service spikes occur, which can reduce consumption caused by cluster scaling.

### Functions:

- Creates pods automatically in seconds. When CCE cluster resources are insufficient, you do not need to add nodes to the cluster. virtual-kubelet automatically creates pods on CCI, eliminating the overhead of resizing the CCE cluster.
- Seamlessly works with Huawei Cloud SWR for you to use public and private images.
- Supports event synchronization, monitoring, logging, remote command execution, and status query for CCI pods.
- Allows you to view the capacity information about virtual elastic nodes.
- Supports connectivity between CCE and CCI pods through Services.

For details, see [Elastic Scaling of CCE Pods to CCI](#).

## 2.3 What Is an Environment Variable?

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

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

## 2.4 What Is a Service?

A service defines a set of instances and a means for accessing them, such as a single stable IP address and corresponding DNS name.

CCI uses service names instead of IP addresses to address communication between components. A service name is specified during workload creation.

## 2.5 What Is Mcore?

One CPU core is equal to 1000 mcores. Generally, the CPU usage of a containerized workload is measured in mcores.

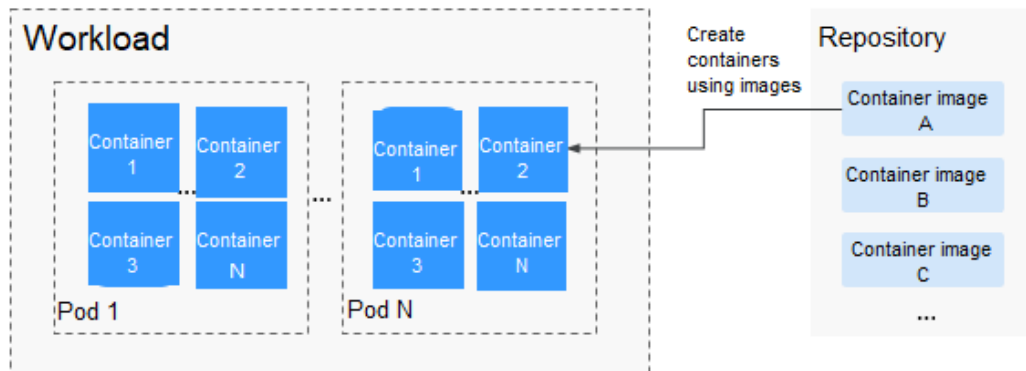
## 2.6 What Are the Relationships Between Images, Containers, and Workloads?

- Image: A container image is a special file system that includes all the programs, libraries, resources, and configuration files for running containers. It also includes some required configuration parameters, such as anonymous volumes, environment variables, and users. An image does not contain any dynamic data, and its content remains unchanged after being built.
- Container: A container is a runtime instance of an image. The relationship between an image and a container is similar to that between a class and an instance in the object-oriented program design. A container can be created, started, stopped, deleted, or suspended.

- A workload is an application running on Kubernetes and consists of one or more **Pods**. A pod consists of one or more containers. Each container is created from a container image.

The following figure shows the relationships between images, containers, and workloads.

**Figure 2-1** Relationships between images, containers, and workloads



## 2.7 What Are Kata Containers?

Kata containers are distinguished from common containers in a few aspects.

Each Kata container or rather pod runs on an independent micro-VM with an independent OS kernel, and is securely isolated from other pods at the virtualization layer. As CCI uses shared multi-tenant clusters, security isolation of containers is of higher requirements than that in the scenarios where users have independent, private Kubernetes clusters. With Kata, containers, kernels, compute resources, storage resources, and networks can be isolated between different tenants, protecting users' resources and data from being preempted or stolen.

## 2.8 Can kubectl Be Used to Manage Container Instances?

Yes. For details, see [Using Native kubectl](#).

## 2.9 What Are Core-Hours in CCI Resource Packages?

1 core-hour = 1 x 3600 core-seconds

A core-hour refers to the amount of resources used by an application when it runs one CPU core for one hour.

A core-second refers to the amount of resources used by an application when it runs one CPU core for one second.

Case 1:

If a Deployment uses 2.5 cores for two consecutive hours, the consumed resources are as follows:  $2.5 \times 2 = 5$  core-hours =  $5 \times 3600$  core-seconds.

Case 2:


If you purchase a package with 730 core-hours, you can allow containers to run 730 cores for one hour or run one core for 730 hours.

For details, see [Billing](#).

# 3 Workload Abnormalities

## 3.1 Locating Method

If a workload is faulty, check events first.

On the CCI console, click **Workloads** in the navigation pane and click the name of the faulty workload. In the **Pod List** area of the workload details page, click  at the left of the faulty pod (workload instance) to show more details about the pod. The **Events** tab page will display pod events.

**Figure 3-1** Viewing events

Pod List Enter an instance name.

Instance (Pod)	Status	Requested CPU (Cores)	Requested Memory (...)	Running Time	Price (¥/s)	Operation
ccf-deployment-201962...	Instance abnormal	2.00	4.00	0 days 0 hours 3 minutes 51s	0.000178	<a href="#">View Logs</a> <a href="#">Delete</a>
ccf-deployment-201962...	Instance abnormal	2.00	4.00	0 days 0 hours 6 minutes 50s	0.000178	<a href="#">View Logs</a> <a href="#">Delete</a>

Monitoring **Events** Container CLI

Events generated by pod instances during container creation and deletion over the past ... Jun 23, 2019 14:52 – Jun 24, 2019 14:52  Enter a Kubernetes event name.

Event Type	Event Name	Kubernetes Event	Occur...	First Occurred	Last Occurred
Abnormal	Failed to Restart C...	the failed container exited with ExitCode: 1	23	Jun 24, 2019 14:46:07 GMT...	Jun 24, 2019 14:50:41 GMT...
Abnormal	InstancesSynchro...	Error syncing pod failed to "StartContainer" for "container-1" with Crash...	1	Jun 24, 2019 14:46:19 GMT...	Jun 24, 2019 14:46:19 GMT...
Abnormal	Failed to Restart C...	Back-off restarting failed container	2	Jun 24, 2019 14:46:07 GMT...	Jun 24, 2019 14:46:19 GMT...
Available	InstancesContaine...	Started container	3	Jun 24, 2019 14:46:02 GMT...	Jun 24, 2019 14:46:17 GMT...
Available	InstancesCreated.	Created container	3	Jun 24, 2019 14:46:01 GMT...	Jun 24, 2019 14:46:17 GMT...
Available	Image Pulled Succ...	Successfully pulled image "100.125.0.198:20202/jorgensen/frontend.f.2"	3	Jun 24, 2019 14:46:01 GMT...	Jun 24, 2019 14:46:16 GMT...

## 3.2 What Do I Do If an Event Indicating That the Image Failed to Be Pulled Occurs?

If the details page of a workload shows an event indicating that the image fails to be pulled, perform the following operations to locate the fault:

## Check Item 1: imagePullSecret (This Is Required When You Use kubectl to Create a Workload)

In the following example, a Deployment named **nginx** is created. Check whether the `.yaml` file contains the **imagePullSecrets** field (bold field in the following information), which indicates the name of the secret used for pulling the image.

To pull an image from the [SoftWare Repository for Container \(SWR\)](#), set this field to **imagepull-secret**.

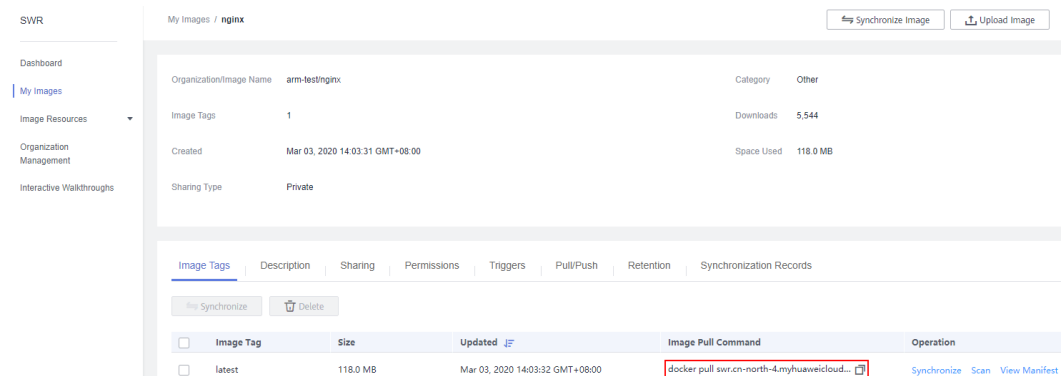
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx:alpine
          imagePullPolicy: Always
          name: nginx
imagePullSecrets:
  - name: imagepull-secret
```

## Check Item 2: Image Address

CCI allows you to create workloads using images pulled from [SWR](#).

SWR images can be obtained by using **Image Pull Command**. After an image is pushed, you can obtain its address.

Figure 3-2 Template address



## Check Item3: IAM Users' Permissions to Download Images

If you have enabled the Enterprise Management service, you need to use your account to grant IAM users with permissions to access SWR so that the IAM users can download private images of the account.

You can grant permissions to IAM users in either of the following ways:

- Grant permissions on the image details page. After permissions granted, IAM users can read, edit, and manage the image. For more details, see [Granting Permissions of a Specific Image](#).
- Grant permissions on the organization details page. After permissions granted, IAM users can read, edit, and manage all images in the organization. For more details, see [Granting Permissions of an Organization](#).

## Check Item 4: Docker Version Used for Image Packaging

The workload fails to pull the image, and the following error information is displayed:

```
failed to pull and unpack image "*****": failed to unpack image on snapshotter devmapper: failed to extract layer
```

```
sha256:xxxxxx: failed to get reader from content store: content digest sha256:xxxxxx: not found
```

The error message is as follows:

**Figure 3-3** Error message

Event Type	Event Name	Kubernetes Event	Occurrences
Abnormal	InstancesSynchronization fai...	Error syncing pod failed to "StartContainer" for "container-0" with ErrImagePull: "rpc error: code = NotFound desc = failed to pull and unpack image "docker.io/library/euleros:2.2": failed to unpack image on snapshotter devmapper: failed to extract layer sha256:1067b21d8ccd1d218d05b245f2b697bf0bfc6a7e7f536df05262444a4a55c98: failed to get reader from content store: content digest sha256:a6a7ec99af33b4ec09de9fe82298fa9e8ab4bae8501e00bcaaf4ab8ae721f22b: not found"	3
Abnormal	InstancesCreation failed.	Error: ImagePullBackOff	3
Abnormal	Failed to Repull Image	Back-off pulling image	3
Abnormal	InstancesSynchronization fai...	Error syncing pod failed to "StartContainer" for "container-0" with ErrImagePull: "rpc error: code = NotFound desc = failed to pull and unpack image "docker.io/library/euleros:2.2": failed to unpack image on snapshotter devmapper: failed to extract layer sha256:1067b21d8ccd1d218d05b245f2b697bf0bfc6a7e7f536df05262444a4a55c98: failed to get reader from content store: content digest sha256:a6a7ec99af33b4ec09de9fe82298fa9e8ab4bae8501e00bcaaf4ab8ae721f22b: not found"	3
Abnormal	InstancesCreation failed.	Error: ErrImagePull	3
Abnormal	InstancesFailed to repull the...	Failed to pull image "swr.cn-east-3.myhuaweicloud.com/dev-container/nginx:wt": rpc error: code = NotFound desc = failed to pull and unpack image "docker.io/library/euleros:2.2": failed to unpack image on snapshotter devmapper: failed to extract layer sha256:1067b21d8ccd1d218d05b245f2b697bf0bfc6a7e7f536df05262444a4a55c98: failed to get reader from content store: content digest sha256:a6a7ec99af33b4ec09de9fe82298fa9e8ab4bae8501e00bcaaf4ab8ae721f22b: not found"	3
Available	Pulling Image	Pulling image "swr.cn-east-3.myhuaweicloud.com/dev-container/nginx:wt"	3

Cause: The Docker version used to build the image is outdated (earlier than v1.10). Some image packaging standards are no longer supported by the community.

Solution: Use Docker v1.11 or later to rebuild the image and push it to SWR, upgrade the workload image tag and pull it again.

## 3.3 What Do I Do If an Event Indicating That the Container Failed to Be Restarted Occurs?

If the details page of a workload shows an event indicating that the container fails to be restarted, perform the following operations to locate the fault:

## Check Item 1: Port Conflict

- Step 1** Configure the kubectl. For details, see [Using Native kubectl](#).
- Step 2** On the CCI console, click the name of the workload whose container failed to restart. In the **Pod List** area of the workload details page, obtain the pod name.
- Step 3** View the name of the failed container.

```
kubectl describe pod $name -n $namespace | grep "Error syncing pod failed to"
```

**Figure 3-4** Viewing the name of the failed container

```
[root@master-1 ~]# kubectl describe pod -n cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2 | grep "Error syncing pod failed to"
Warning FailedSync 7m kubelet, 1ce9a08e-dd39-e911-alfb-9835ed8fc6a9 Error syncing pod failed to "StartContainer" for "container-1" with CrashLoopBackOff: "Back-off 10s restarting failed container=container-1 pod=port-conflict-7d4fc654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
Warning FailedSync 6m kubelet, 1ce9a08e-dd39-e911-alfb-9835ed8fc6a9 Error syncing pod failed to "StartContainer" for "container-1" with CrashLoopBackOff: "Back-off 20s restarting failed container=container-1 pod=port-conflict-7d4fc654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
```

- Step 4** View the logs of the container.

```
kubectl logs $podName -n $namespace -c $containerName
```

```
[root@master-1 ~]# kubectl logs port-conflict-7d4fc654bb-tc9gw -n cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2 -c container-1
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: still could not bind()
```

Solution: Re-create a workload and configure correct ports to avoid port conflicts.

----End

## Check Item 2: Workload Bugs

Check whether the workload startup command is correctly executed or whether the workload has a bug.

- Step 1** Configure the kubectl. For details, see [Using Native kubectl](#).
- Step 2** On the CCI console, click the name of the workload whose container failed to restart. In the **Pod List** area of the workload details page, obtain the pod name.
- Step 3** View the name of the failed container.

```
kubectl describe pod $name -n $namespace | grep "Error syncing pod failed to"
```

**Figure 3-5** Viewing the name of the failed container

```
[root@master-1 ~]# kubectl describe pod -n cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2 | grep "Error syncing pod failed to"
Warning FailedSync 7m kubelet, 1ce9a08e-dd39-e911-alfb-9835ed8fc6a9 Error syncing pod failed to "StartContainer" for "container-1" with CrashLoopBackOff: "Back-off 10s restarting failed container=container-1 pod=port-conflict-7d4fc654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
Warning FailedSync 6m kubelet, 1ce9a08e-dd39-e911-alfb-9835ed8fc6a9 Error syncing pod failed to "StartContainer" for "container-1" with CrashLoopBackOff: "Back-off 20s restarting failed container=container-1 pod=port-conflict-7d4fc654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
```

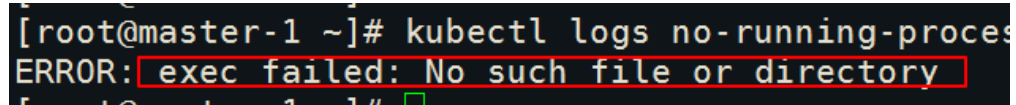


**Step 4** View the logs of the container.

```
kubectl logs $podName -n $namespace -c $containerName
```

Fix the workload bugs indicated in logs.

**Figure 3-6** Incorrect startup command configuration of the container



```
[root@master-1 ~]# kubectl logs no-running-proces  
ERROR: exec failed: No such file or directory
```

Solution: Re-create a workload and configure a correct startup command.

----End

### Check Item 3: Workload Health Check

If the liveness probing (a type of health check) is enabled for the workload and the number of health check failures exceeds the threshold, the workload details page shows the Kubernetes event **Liveness probe failed: .....** and containers in the workload instance will be restarted. In this case, reconfigure the health check policies.

### Check Item 4: Resource Type of the Namespace

Check whether the resource type selected during namespace creation is correct. General-computing and GPU-accelerated resources support x86 images.

**Step 1** Log in to the CCI console and click the name of the workload whose container failed to restart. The workload details page is displayed.

**Step 2** In **Pod List**, click **View Logs** in the **Operation** column of the abnormal pod.

**Step 3** View the displayed error messages.

```
ERROR: exec failed: Exec format error
```

```
ERROR: hyper send process initiated event: error
```

----End

### Other Check Items

A workload fault may be caused by the failure to start the workload inside the container. In this case, you can manually run the startup command inside the pod to which the container belongs and rectify the fault based on the error message. The specific procedure is as follows:

1. Configure the following startup command for the workload. In this way, the application will not be started after the pod is started and no operation will be performed.

Advanced Settings

Storage: Persistent volumes can be attached to containers for persistent data storage. ▼

Log Collection: Specify the container log path and the log storage space for application logs. You also need to configure policies to prevent logs from being over-sized. [Learn how](#) to use the Log Collection. ▼

Environment Variables: Environment variables are set in the container runtime environment, and can be modified after application deployment. ▼

Health Check: Health check regularly checks the health status of containers or applications. [Learn how](#) to configure a health check. ▼

Lifecycle: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. [Learn how](#) to configure a lifecycle script. ▼

Startup Commands: Startup commands correspond to Docker ENTRYPOINT commands. [Learn how](#) to configure startup commands. ▲

Executable Command

Parameters ⇌ Switch to single line input mode

Delete

⊕ Add

**Example**

Binary Mode Bash Mode

Executable

Command

Parameters

**NOTE**

- Before running the startup command, ensure that the **/bin/bash** command in the image is available.
- After the pod is started, run the **kubectl exec** command to enter the inside of the pod. Then, manually run the startup command inside the pod, and rectify the fault based on the error message.

# 4 Container Workload FAQs

---

## 4.1 Why Service Performance Does Not Meet the Expectation?

The underlying resources of CCI are shared by multiple tenants. To ensure service stability, the underlying layer of CCI has traffic control on disk I/O. For containers, the read and write operations on the **rootfs** directory and the standard output of workload logs are limited. If the service performance does not meet your expectations, locate the fault based on the following causes:

**Cause 1: Service containers print a large number of log files to stdout streams.**

The underlying layer of CCI limits the forwarding traffic of standard output. If the service log traffic is greater than 1 MB/s, you are advised to use log volumes to report logs to Application Operations Management (AOM). For details, see [Log Management](#). Alternatively, you can output logs to FlexVolume or persistent EVS or SFS volumes, and use a sidecar to run an open-source component such as Fluent Bit to report logs to your self-built log centers. If a large number of logs are printed to stdout streams, service performance may be affected due to the forwarding traffic limits.

**Cause 2: Service containers have high I/O read and write operations on the rootfs disk.**

CCI limits the I/O traffic of the container system disk (**rootfs**). If a process may have high disk I/O operations (bandwidth > 6 Mbit/s, IOPS > 1,000) during the service running or is sensitive to the disk I/O performance, do not execute it in **rootfs**. For example, frequently printing logs to **rootfs** may cause frequent read and write operations on it. You can place service-related configuration files or files that are not frequently read and written in **rootfs**. For high I/O file operations, select log volumes or FlexVolume (created or deleted with the pod), or persistent EVS or SFS volumes based on your service requirements. Executing high I/O operations in the **rootfs** disk may affect service performance due to disk traffic limiting.

If the preceding problems occur, adjust configurations based on the description to prevent service performance from being affected.

## 4.2 How Do I Set the Quantity of Instances (Pods)?


### When Creating a Workload

Click the plus and minus signs or enter a value to set the quantity of instances (pods).

**Figure 4-1** Setting the quantity of instances (pods)

The screenshot shows a form for creating a workload. The 'Workload Name' field contains 'nginx'. The 'Namespace' dropdown is set to 'im-test'. Below the namespace, it shows 'Available General-computing | vpc-mogujie 192.168.0.0/16'. The 'Description' field is empty. At the bottom, the 'Pods' field is highlighted with a red box and contains the number '2' with minus and plus buttons on either side.

### After a Workload Is Created

Click the workload whose instance quantity needs to be adjusted. In the **Scaling** area on the workload details page, click **Manual Scaling** and then , and set the instance quantity.

**Figure 4-2** Manual scaling

The screenshot shows the 'Scaling' section of a workload details page. It includes a 'Time Window (s)' field set to 30. Below this are two tabs: 'Manual Scaling' (selected) and 'Auto Scaling'. Under the 'Manual Scaling' tab, the 'Instances' field is highlighted with a red box and contains the number '1' with a pencil icon to its right.


**Figure 4-3** Adjusting the instance quantity

This screenshot is similar to Figure 4-2, showing the 'Manual Scaling' tab. The 'Instances' field is highlighted with a red box and shows a value of '1'. The field has minus and plus buttons on either side, and a dropdown arrow on the right.

## 4.3 How Do I Check My Resource Quotas?

CCI imposes a quota on the number and capacity of resources that can be used by a user. For details, see [Quotas](#).

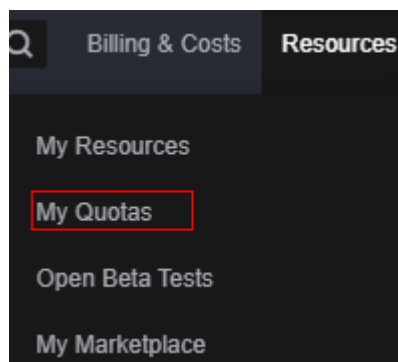
**Step 1** Log in to the Huawei Cloud console.

**Step 2** Click  in the upper left corner and select the desired region and project.

**Step 3** Choose **Resources > My Quotas** in the upper right corner of the page.

The **Quotas** page is displayed.

Figure 4-4 My quotas



**Step 4** View the used quota and total quota of resources on the **Quotas** page.

Figure 4-5 CCI resource quotas

Instances per namespace	0	80
Services per namespace	0	40
gpu_beta_t4	0	256
gpu_beta_q4	0	256
gpu_beta_100_32GB	0	256
gpu_beta_100_16GB	0	256
Ingresses per namespace	0	40
Networks per namespace	0	6
Statefulsets per namespace	0	40
Deployments per namespace	0	40
Cronjobs per namespace	0	80
Jobs per namespace	0	80
Namespaces	0	5
Volumes	0	80
Secrets/SSL_Cert per namespace	0	40
ConfigMaps per namespace	0	40
Memory per namespace(MB)	0	8,192,000
CPU per namespace(mCore)	0	1,000,000

----End

## 4.4 How Do I Set Probes for a Workload?

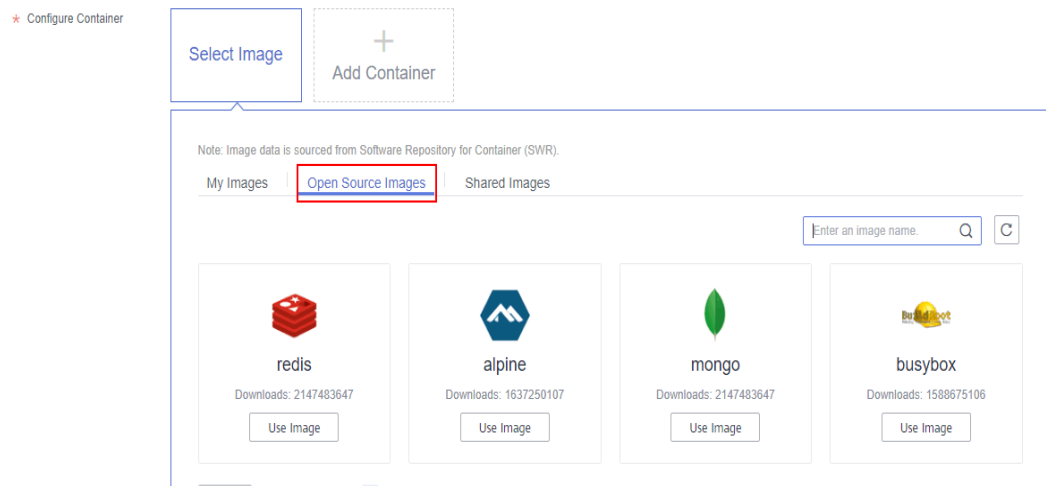
CCI supports Kubernetes-based liveness probing and readiness probing. You can set them when creating a workload. For details, see [Setting Health Check Parameters](#).

## 4.5 How Do I Configure an Auto Scaling Policy?

CCI supports three types of auto scaling policies: metric-based, scheduled, and periodic. For details, see [Scaling a Workload](#).

## 4.6 What Do I Do If the Workload Created from the sample Image Fails to Run?

If you have used SWR but have not pushed any image to SWR, SWR will build an image named **sample** for you. However, this image cannot run. You are advised to use an image on the **Open Source Images** tab to create a workload.



## 4.7 How Do I View Pods After I Call the API to Delete a Deployment?

The `propagationPolicy` field in a Deployment deletion API request indicates whether to delete pods along with the Deployment. This field can be set to **Orphan**, **Foreground**, or **Background**. For details about this field, see [Deleting a Deployment](#).

## 4.8 Why an Error Is Reported When a GPU-Related Operation Is Performed on the Container Entered by Using exec?

### Symptom

After I enter a container using `exec` and perform a GPU-related operation (such as using `nvidia-smi` or running a GPU training task using TensorFlow), the error message "cannot open shared object file: No such file or directory" is displayed.

```
root@cci-deployment-20196201-7fcb6477-khj8d:/workspace/examples/image-classification# python train_image.py
Traceback (most recent call last):
  File "train_imagenet.py", line 22, in <module>
    from common import find_mxnet, data, dali, fit
  File "/workspace/examples/image-classification/common/find_mxnet.py", line 20, in <module>
    import mxnet as mx
  File "/opt/mxnet/python/mxnet/_init_.py", line 24, in <module>
    from .context import Context, current_context, cpu, gpu, cpu_pinned
  File "/opt/mxnet/python/mxnet/context.py", line 24, in <module>
    from .base import classproperty, with_metaclass, _MXClassPropertyMetaClass
  File "/opt/mxnet/python/mxnet/base.py", line 212, in <module>
    _LIB = _load_lib()
  File "/opt/mxnet/python/mxnet/base.py", line 203, in _load_lib
    lib = ctypes.CDLL(lib_path[0], ctypes.RTLD_LOCAL)
  File "/usr/lib/python3.5/ctypes/_init_.py", line 347, in _init_
    self.handle = dlopen(self.name, mode)
OSError: libcuda.so.1: cannot open shared object file: No such file or directory
```

### Possible Cause

The CUDA library in a container is located in `/usr/local/nvidia/lib64`. This directory must be added to `LD_LIBRARY_PATH` to ensure that the CUDA library can be found.

### Solution

Log in to the GPU-accelerated container by using `kubectl exec` or console, run the `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/nvidia/lib64` command, and then perform other GPU-related operations.

## 4.9 Can I Start a Container in Privileged Mode When Running the `systemctl` Command in a Container in a CCI Cluster?

Currently, CCI does not support the privileged mode.

The privileged mode is enabled only when a container is allowed to access all devices on the host. CCI adopts the Kata hypervisor to isolate containers. Therefore, resources on the host are completely isolated from containers.

In other scenarios, you are advised to use the `securityContext` of Kubernetes to implement fine-grained permissions control, thereby ensuring the security and reliability of the container running environment.

## 4.10 Why Does the Intel oneAPI Toolkit Fail to Run VASP Tasks Occasionally?

Vienna Ab initio Simulation Package (VASP) tasks (used for electronic structure computing and quantum mechanics-molecular dynamics simulation) running on the Intel oneAPI Toolkit (Intel parallel computing platform) depends on the physical CPU version. If pod specifications are not high, VASP tasks may fail to run. If this happens, change the oneAPI version or upgrade pod specifications (at least four cores).

## 4.11 Why Are Pods Evicted?

If a node is overloaded, the system automatically evicts pods and clears some jobs. You are advised to apply for resources based on the actual usage.

## 4.12 Why Is the Workload Web-Terminal Not Displayed on the Console?

If the workload web-terminal cannot be displayed, check whether the account is in arrears.

## 4.13 Why Are Fees Continuously Deducted After I Delete a Workload?

After a workload is deleted, it still exists on the **My Resources** page and fees are continuously deducted. In this case, check whether there are pods running in the

namespace where the workload belongs. If there are pods running and you do not need them anymore, delete the pods. The billing will be stopped.



# 5 Image Repository FAQs

---

## 5.1 Can I Export Public Images?

You cannot export or download images uploaded by other users.

## 5.2 How Do I Create a Container Image?

This section describes how to use Dockerfile to create a container image for a simple web workload.

### Context

After you create a containerized workload using an official Nginx image, the default Nginx welcome page is displayed. The following describes how to customize an image to change the default welcome message to **Hello, CCI!**.

### Procedure

**Step 1** Log in to the VM running the container engine as the **root** user.

**Step 2** Run the following commands to create a file named **Dockerfile**:

```
mkdir mynginx  
cd mynginx  
touch Dockerfile
```

**Step 3** Run the following command to edit the **Dockerfile** file:

```
vi Dockerfile
```

File content:

```
FROM nginx  
RUN echo '<h1>Hello,CCI!</h1>' > /usr/share/nginx/html/index.html
```

Where:

- FROM statement: specifies that an Nginx image is used as a base image.
- RUN statement: indicates that the **echo** command is executed to display "Hello, CCI!"

**Step 4** Create a container image.

```
docker build -t nginx:v3
```

**Step 5** Run the following command to check the created image. The command output shows that the nginx image has been created with a tag of v3.

```
docker images
```

```
----End
```

## 5.3 How Do I Upload Images?

Software Repository for Container (SWR) manages images. It provides two methods to upload images:

- [Uploading an Image Through a Container Engine Client](#)
- [Uploading an Image Through the SWR Console](#)

## 5.4 Does CCI Provide Base Container Images for Download?

The image repository of CCI is provided by SWR, which provides base container images for download.

## 5.5 Does CCI Administrator Have the Permission to Upload Image Packages?

To upload images for CCI, you need to use SWR.

You also need to assign the **Tenant Administrator** permission to your account.

## 5.6 What Permissions Are Required for Uploading Image Packages for CCI?

To upload images for CCI, you need to use SWR.

You need to assign the **Tenant Administrator** permission to your account. For details, see [SWR permissions](#).

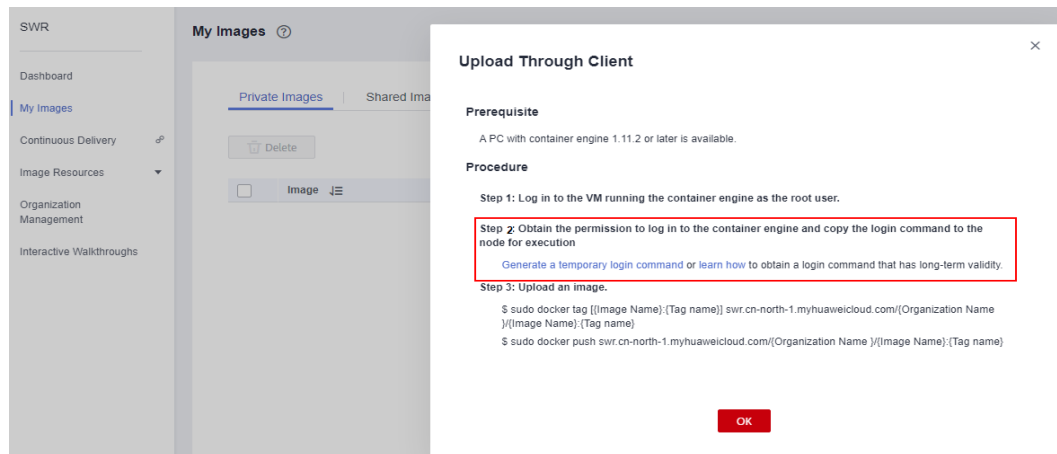
For details about how to upload an image, see [Uploading an Image Through a Container Engine Client](#) or [Uploading an Image Through SWR Console](#).

## 5.7 What Do I Do If Authentication Is Required During Image Push?

To upload images for CCI, you need to use SWR.

To upload images using SWR, first obtain the permission to access SWR. For details about how to upload images, see [Uploading an Image Through a Container Engine Client](#).

Figure 5-1 Uploading an image

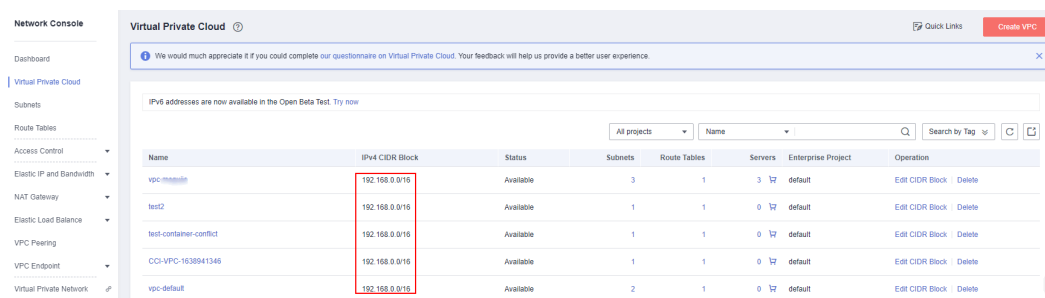


# 6 Network Management FAQs

## 6.1 How Do I View the VPC CIDR Block?

On the home page of the VPC console, view **Name** and **IPv4 CIDR Block** of VPCs. You can modify the CIDR block of a VPC or re-create a VPC.

Figure 6-1 Viewing the VPC CIDR blocks



The screenshot shows the 'Virtual Private Cloud' console interface. A table lists several VPCs with their respective IPv4 CIDR blocks highlighted in red. The table has columns for Name, IPv4 CIDR Block, Status, Subnets, Route Tables, Servers, Enterprise Project, and Operation.

Name	IPv4 CIDR Block	Status	Subnets	Route Tables	Servers	Enterprise Project	Operation
vpc-main	192.168.0.0/16	Available	3	1	3	default	Edit CIDR Block Delete
test2	192.168.0.0/16	Available	1	1	0	default	Edit CIDR Block Delete
test-container-conflict	192.168.0.0/16	Available	1	1	0	default	Edit CIDR Block Delete
CCI-VPC-1638941346	192.168.0.0/16	Available	1	1	0	default	Edit CIDR Block Delete
vpc-default	192.168.0.0/16	Available	2	1	0	default	Edit CIDR Block Delete

## 6.2 Does CCI Support Load Balancing?

CCI supports load balancing. On the **Configure Access Settings** page for creating a workload on the CCI console, the access mode can be set to **ELB** for both intranet access and Internet access. For details about ELB-based intranet access, see [Workload Access Through a Private Network Load Balancer](#).

Generally, load balancing refers to public network load balancing. CCI connects to Elastic Load Balance (ELB) to implement load balancing.

When creating a workload on the CCI console, you can choose **Intranet access** or **Internet access** and configure a load balancer on the **Configure Access Settings** page.

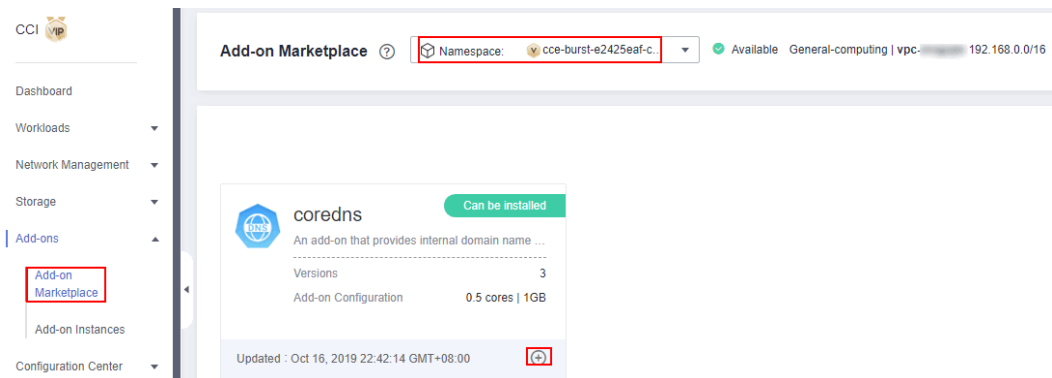
- For details about public network load balancing, see [Public Network Access](#).
- For details about private network load balancing, see [Private Network Access](#).

## 6.3 How Do I Configure the DNS Service on CCI?

If workloads require the internal domain name resolution provided by Kubernetes, the `coredns` add-on must be installed. In this case, `dnsPolicy` of pods must be set to **ClusterFirst**.

On the **Add-on Marketplace** page, click  on the card of the `coredns` add-on to install it under the specified namespace.

**Figure 6-2** Installing an add-on



If workloads do not require the internal domain name resolution provided by Kubernetes but require DNS, `dnsPolicy` of pods must be set to **Default**.

In addition, you can set `dnsPolicy` of pods to **None** to use the custom DNS service. The following is a YAML example:

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0
```

## 6.4 Does CCI Support InfiniBand (IB) Networks?

No.

## 6.5 How Do I Access a Container from a Public Network?

You can bind a load balancer to a workload and use the IP address of the load balancer to access the workload from public networks. For details, see [Public Network Access](#).

If you use the kubectl, you can create the service and ingress objects and bind load balancers to them, enabling container access from the public network. For details, see [Service](#) and [Ingress](#).

## 6.6 How Do I Access a Public Network from a Container?

You can use the NAT Gateway service available on the public cloud platform. This service offers NAT for containers in a VPC, allowing these containers to access the Internet using an EIP. The NAT Gateway service supports a large number of concurrent connections, which makes it suitable for applications involving a large number of requests and connections. For details, see [Accessing Public Networks from a Container](#).

## 6.7 What Do I Do If Access to a Workload from a Public Network Fails?

1. A workload must be in running state before it can be accessed from a public network. If your workload is abnormal or not ready, it cannot be accessed properly from a public network.
2. The workload cannot be accessed properly from the public network until the network route is configured, which takes 1 to 3 minutes.
3. If the workload cannot be accessed 3 minutes after being created, click the workload on the CCI console. On the details page that is displayed, choose **Access Settings** to check whether any alarm events are reported. The container cannot be accessed from the public network if either of the following events is displayed:
  - Listener port is repeated: This event occurs when you delete a workload for which a load balancer port is configured, and immediately after that, create a workload using the same load balancer port. It takes some time for a load balancer port to be deleted. Wait for 5 to 10 minutes, and then the workload can be accessed from the public network.
  - Create listener failed: This event occurs usually because the listener quota is exceeded. Select another load balancer with a sufficient quota.
4. The workload is still inaccessible after it has been created for 3 minutes, and there is no alarm event. The possible reason is that no corresponding process is actually listening to the user-configured container port. Currently, CCI cannot detect this type of exception. You need to check whether the image is listening to this container port. If the container port is properly listened to, the

access failure may be caused by the load balancer. In this case, check the status of the load balancer.

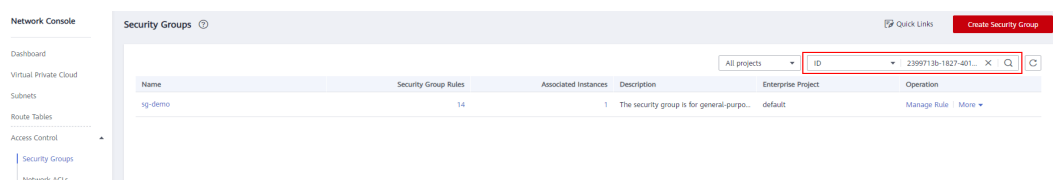
## 6.8 What Do I Do If Error 504 Is Reported When I Access a Workload?

Generally, error 504 is reported when the security group is not configured to allow the traffic from the load balancer port to CCI workload pods. Check the security group used by the CCI workload pods and ensure that the security group rule allows the traffic from the port bound to the load balancer.

You can obtain the security group bound to a pod by calling the [API for reading the network](#) of the workload. In a response, **network.alpha.kubernetes.io/default-security-group** in **metadata.annotations** is the security group ID.

```
{
  "kind": "Network",
  "apiVersion": "networking.cci.io/v1beta1",
  "metadata": {
    "name": "namespace-test-dc1-default-network",
    "namespace": "namespace-test",
    "selfLink": "/apis/networking.cci.io/v1beta1/namespaces/namespace-test/networks/namespace-test-dc1-default-network",
    "uid": "6fb85414-af6b-11e8-b6ef-f898ef6c78b4",
    "resourceVersion": "5016899",
    "creationTimestamp": "2018-09-03T11:21:00Z",
    "annotations": {
      "network.alpha.kubernetes.io/project-id": "51bf52609f2a49c68bfda3398817b376",
      "network.alpha.kubernetes.io/default-security-group": "19c5d024-aed5-4856-b958-c0f65ce70855",
      "network.alpha.kubernetes.io/domain-id": "aadb43c0b14c4cafbccfff483d075987"
    },
    "enable": true
  },
  "spec": {
    "cidr": "192.168.244.0/23",
    "attachedVPC": "0d4080e5-546a-46c4-86fe-f3e26d685177",
    "networkType": "underlay_neutron",
    "physicalNetwork": "phy_net0",
    "networkID": "0022e356-f730-4226-802e-9cdaa6e7da17",
    "subnetID": "1ffd839d-e534-4fa8-a59d-42356335bf74",
    "availableZone": "cnnorth1a"
  },
  "status": {
    "state": "Active"
  }
}
```










Log in to the [Network Console](#), and search for the security group based on the obtained security group ID.



Click the security group name, and add the rules shown in the following figure on the **Inbound Rules** tab page.

**NOTICE**

If you access the workload from the public network through UDP, an ICMP rule must be added, which will be used during health check.

<input type="checkbox"/> Protocol & Port  	Type	Source 
<input type="checkbox"/> All	IPv4	sg-test 
<input type="checkbox"/> ICMP : Echo	IPv4	0.0.0.0/0 
<input type="checkbox"/> TCP : 22	IPv4	0.0.0.0/0 
<input type="checkbox"/> TCP : 3389	IPv4	0.0.0.0/0 
<input type="checkbox"/> TCP : 1-65535	IPv4	0.0.0.0/0 
<input type="checkbox"/> UDP : 1-65535	IPv4	0.0.0.0/0 

## 6.9 What Do I Do If the Connection Timed Out?

### Symptom

Pods can be created, but the message "[Errno 110] Connection timed out" is displayed when the Python Django SMTP service is used to send emails.

### Possible Cause

- You have purchased only ELB but not NAT Gateway. Therefore, containers can be accessed only from the external network. Containers can access the external network after you purchase a NAT Gateway.
- Port 25 is prohibited from sending messages to secure the network environment.

### Solution

- Solution 1: Use the [NAT Gateway](#) service to enable container instances (pods) in a VPC to access public networks. The NAT Gateway service provides source network address translation (SNAT), which translates private IP addresses to a public IP address by binding an elastic IP address (EIP) to the gateway, providing secure and efficient access to the Internet. For details, see [Accessing Public Networks from a Container](#).
- Solution 2: Contact technical support to allow port 25 of your new EIP.



# 7 Storage Management FAQs

---

## 7.1 Which Cloud Storage Types Are Supported and Which Types Require Backup?

Currently, CCI supports the following cloud storage types:

- EVS disks
- SFS Turbo file systems

If an EVS disk is used, you need to manually configure a backup policy for it. For details, see [VBS Backup Management](#).

## 7.2 How Do I Use Cloud Storage?

You can create cloud storage on the CCI console. For details, see [Overview](#).

## 7.3 Where Is the Data Generated by a Container Stored If No Cloud Storage Is Mounted?

If no EVS disk or other cloud storage is mounted, the application data is stored in the physical machine disks of the container. Each pod can be allocated with a maximum of 20 GB disk of the CPU host or 20-GB disk of the GPU host for data storage. If the node is a dedicated node, the storage space can be adjusted based on customer requirements.

### NOTE

To ensure data security, the container engine obtains a virtual disk from devicemapper when creating a container. Other container engines cannot access the virtual disk of this container.

## 7.4 Why Volume Mounting Failure Is Reported When a Job Pod Completes Its Task?

### Symptom

When a job pod completes its task, there are still events showing that a volume failed to be mounted to the pod.

### Possible Cause

When a pod of a Deployment, StatefulSet, job, or cron job is started on a node:

- The kubelet creates a podWorker (an independent goroutine) to check volume mounting into the pod at a regular interval of 0.3 seconds. If the volume mounting is not complete within 0.3 seconds or the timeout duration (123 seconds) expires, the event "Unable to mount volumes for pod ..." is reported.
- The VolumeManager (an independent goroutine) in the kubelet is responsible for mounting volumes into the pod.

For a long running pod of a Deployment or StatefulSet: If the pod is started without experiencing failures to pull images, mount volumes, assign container CIDR block, or provide node CPU or memory resources that match the pod's needs, the volume mounting will eventually be successful after a few check cycles.

For a short-lived pod of a job or cron job: The pod is very likely to exit gracefully soon after it is started. For example, the GCS Demo job in question runs only a few echo and ls commands and exits in less than 1 second. If the pod exits gracefully during the interval between two consecutive volume mounting checks, a false event notification will be generated but services are not affected. In practice, pod jobs always run longer than 1 second.

The kubelet's capability to check volume mounting is already considered in the volume mounting framework provided by the community.

### Solution

A pod of a job or cron job usually runs for a short period of time. If the pod completes its task much earlier than the timeout duration of volume mounting, the volume mounting timeout event is a false event notification and can be ignored. The event has no impact on the service run by the pod.

# 8 Log Collection

---

## 8.1 Why are logs duplicated or lost?

### Logs collected repeatedly

Cause 1: The log file is dumped, and the dumped file is still matched.

Description: If the file name of the configured log path contains a wildcard character, for example, `/tmp/*.log`, after the `/tmp/test.log` file is dumped to `/tmp/test.001.log`, it is still matched by the wildcard rule and is collected again as a new file.

### Log loss

Cause 1: The log file exists for less than 5 seconds.

Description: CCI detects log files at an interval of 5 seconds. If the duration from the time when a log file is created to the time when it is deleted or renamed is less than 5 seconds, it may not be collected.

# 9 Account

---

## 9.1 Why Is a Message Displayed Indicating That My Account Is In Arrears Even If My Account Has Balance?

### Symptom

When the account is reset and has balance, a message is displayed indicating that the account is in arrears during namespace creation.

### Solution

Log out and log in again, or clear the browser cache. Then, try to create the namespace again.

## 9.2 Why Resources Cannot Be Deleted?

### Symptom

Resources cannot be deleted.

### Solution

Some resource permissions are restricted because your account is in arrears. Top up the account and then delete the resources.