Copyright © Huawei Technologies Co., Ltd. 2019. 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 Technologies Co., Ltd.

Trademarks and Permissions

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

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

1 Environment Configuration ........................................................................................................ 1
  1.1 Logging In to the CCI Console .......................................................................................... 1
  1.2 (Optional) Uploading Images ....................................................................................... 1
  1.3 Creating an ELB (Optional) ............................................................................................ 1
  1.4 Preparing the SSL Certificate (Optional) ........................................................................ 2

2 Namespace .............................................................................................................................. 3

3 Workload ........................................................................................................................................ 6
  3.1 Pod ................................................................................................................................ 6
  3.2 Deployment .................................................................................................................... 8
  3.3 Job ................................................................................................................................. 12
  3.4 Cron Job ....................................................................................................................... 15
  3.5 Viewing Resource Usage .............................................................................................. 18
  3.6 Setting Container Startup Command .............................................................................. 18
  3.7 Container Lifecycle Hook ............................................................................................. 20
  3.8 Health Check ................................................................................................................ 20
  3.9 Web-Terminal ............................................................................................................... 22
  3.10 Upgrading a Workload ............................................................................................... 22
  3.11 Scaling a Workload .................................................................................................... 25

4 Load Network Access ............................................................................................................... 31
  4.1 Network Access Overview ............................................................................................ 31
  4.2 Private Network Access ............................................................................................... 32
  4.3 Public Network Access ................................................................................................. 35
  4.4 Accessing Public Networks from a Container ............................................................... 39

5 Storage Management ............................................................................................................... 44
  5.1 Overview ...................................................................................................................... 44
  5.2 EVS Disk ...................................................................................................................... 45
  5.3 SFS File System ........................................................................................................... 47
  5.4 SFS Turbo File System ............................................................................................... 49
  5.5 OBS ............................................................................................................................. 50

6 Configuration Management .................................................................................................. 52
  6.1 Using a ConfigMap ....................................................................................................... 52
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2 Using a Secret</td>
<td>54</td>
</tr>
<tr>
<td>6.3 SSL Certificate</td>
<td>57</td>
</tr>
<tr>
<td>7 Log Management</td>
<td>60</td>
</tr>
<tr>
<td>8 Add-on Management</td>
<td>62</td>
</tr>
</tbody>
</table>
1 Environment Configuration

1.1 Logging In to the CCI Console

Log in to the Cloud Container Instance (CCI) console and grant CCI the permission to access other cloud services.

Procedure

Step 1  Log in to the management console.

Step 2  Click in the upper left corner and select a region.

Step 3  In the All Services area, choose Computing > Cloud Container Instance.

Step 4  At the first time you log in to the CCI console, a message will be displayed, asking you to grant CCI the permission to access other cloud services. Click Agree.

When the permission is successfully granted, an agency named cci_admin_trust will be created. You can view the agency on the IAM console.

----End

1.2 (Optional) Uploading Images

HUAWEI CLOUD provides the Software Repository for Container (SWR) service for you to upload Docker images to the image repository. You can easily import these images when creating workloads on CCI. For details about how to upload images, see Uploading an Image to SWR.

1.3 Creating an ELB (Optional)

You can access a container workload from an external network through an ELB. For details about how to create an ELB, choose Help Center > Elastic Load Balance > Quick Start > Creating an Enhanced Load Balancer.
Step 1  Log in to the management console.

Step 2  Choose Service List > Network > Elastic Load Balance.

Step 3  On the Elastic Load Balance page, click Buy Enhanced Load Balancer to create an ELB.

Specify the parameters to create the ELB.

**NOTE**

The ELB can be a private or a public network ELB. Set the parameter type to Public network or Private network.

---End

1.4 Preparing the SSL Certificate (Optional)

CCI allows you to access loads using HTTPS. When creating a load, you can use your own
SSL certificate.

SSL certificates are divided into authoritative certificates and self-signed certificates.
Authoritative certificates are issued by authoritative digital certificate certification authorities,
which can be purchased from third-party certificate agents. The client trusts websites that use
authoritative certificates by default. Self-signed certificates are issued by users themselves.
Generally, they can be generated by using OpenSSL and are untrusted by the client by default.
The browser will display an alarm message during access, and you can continue normal
access by ignoring the alarm.

For details about the SSL certification, see Section 6.3 SSL Certificate.
Namespaces are a way to divide cluster resources among multiple users. Namespaces are suited for scenarios where multiple users spread across multiple teams or projects.

Currently, CCI provides general-computing and GPU-accelerated namespaces. You need to select the resource type when creating a namespace, so that the container in the created workload runs on this type of clusters.

- General-computing: Supports creation of container instances with CPU resources. This namespace type is suitable for general computing scenarios.
- CPU-accelerated: Supports creation of container instances with GPU resources. This namespace type is suitable for scenarios such as deep learning, scientific computing, and video processing.

Relationship Between Namespaces and Networks

A namespace corresponds to a subnet in a VPC, as shown in Figure 2-1. When a namespace is created, it will be associated with an existing VPC or a newly created VPC, and a subnet will be created under the VPC. Containers and other resources created under this namespace will be in the corresponding VPC and subnet.

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

Figure 2-1 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 console. In the navigation pane, choose Namespaces.

Step 2  On the page displayed on the right, click Create for the target namespace type.

**NOTE**
If you click Quick Creation, a namespace will be created with a random name and associated with an existing VPC and subnet in random. If no VPCs are available, a VPC will be automatically created with a subnet allocated to each AZ.

Step 3  Enter a name for the namespace.

**NOTE**
The namespace name must be globally unique in CCI.

Step 4  Configure a VPC.

You can use an existing VPC or create a VPC. If you create a VPC, it is recommended that the VPC CIDR block be set to 10.0.0.0/8 – 24, 172.16.0.0/12 – 24, or 192.168.0.0/16 – 24.

---

**NOTICE**
The VPC CIDR block and subnet CIDR block cannot be set to 10.247.0.0/16, because this CIDR block is reserved by CCI for containerized workloads. If you use this CIDR block, IP address conflicts may occur, which may result in workload creation failures or service unavailability. If you do not need to access pods through workloads, you can allocate this CIDR block to a VPC.

Step 5  Configure a subnet CIDR block.

Ensure that there are sufficient available IP addresses. If the number of IP addresses are insufficient, workloads will fail to be created.

**Figure 2-2** Configuring a subnet

**Subnet Settings**

<table>
<thead>
<tr>
<th>Select Subnet</th>
<th>Existing subnet</th>
<th>New subnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Name</td>
<td>cci-cnorth1a-1315423911</td>
<td></td>
</tr>
<tr>
<td>Subnet CIDR Block</td>
<td>192.168.128.0 / 19</td>
<td></td>
</tr>
</tbody>
</table>

**Available CIDR Blocks**
192.168.0.0/16

**Available IP Addresses**
8186
Step 6 Click Create.

After the creation is complete, you can view the VPC and subnet information on the namespace details page.

---End

Deleting a Namespace

NOTICE

Deleting a namespace will remove all data resources (workloads, ConfigMaps, secrets, and SSL certificates) related to the namespace.

Step 1 Log in to the CCI console. In the navigation pane, choose Namespaces. On the page displayed on the right, click the namespace to be deleted.

Step 2 In the upper right corner, click Delete. In the dialog box that is displayed, enter DELETE and click Yes.

NOTE

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

---End

Creating a Namespace Through kubectl

For details, see Namespace and Network.
3 Workload

3.1 Pod

What Is Pod?

Pod is the minimum unit that Kubernetes uses to create or deploy resources. A pod encapsulates one or more containers, volumes, an independent network IP address, and policies for controlling container running.

A pod can be used in the following two modes:

- A pod runs a container. This is the most common mode in Kubernetes. You can consider a pod as a single encapsulated container. Kubernetes directly manages the pod instead of the container.

- A pod runs multiple containers that need to be coupled to work and need to share resources. In this scenario, an application contains a main container and several sidecar containers, as shown in **Figure 3-1**. For example, the main container is a web server that provides file services from a fixed directory, and the sidecar container periodically downloads files to the directory.
In Kubernetes, pods are rarely created directly. Instead, controllers such as Deployments and jobs, are used to manage pods. Controllers can create and manage multiple pods, and provide replica management, rolling upgrade, and self-healing capabilities. A controller generally uses a pod template to create corresponding pods.

**Viewing Pods**

Sometimes you may create pods by calling the API or running the kubectl command. As these pods are not created under a workload or job, they cannot be conveniently managed on the console. To solve this problem, CCI provides pod management, which allows you to filter pods by source.

**Figure 3-2 Selecting a pod source**

You can view details about all pods, including basic information, container composition, monitoring data, and events. You can use the web-terminal to access pods. In addition, you can delete pods and view pod logs.

**Figure 3-3 Pod details**
Creating a Pod Through `kubectl`

For details, see Pod.

3.2 Deployment

A Deployment is a service-oriented encapsulation of pods. A Deployment may contain one or more pods. Each pod has the same role; therefore, the system automatically distributes requests to the pods of the Deployment. All pods in a Deployment share the same storage volume.

As described in 3.1 Pod, a pod is the minimum unit that Kubernetes uses to create or deploy resources. It is designed to be an ephemeral, one-off entity. A pod can be evicted when node resources are insufficient and disappears along with a cluster node failure. Kubernetes 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. Each pod replica has the same role. Therefore, the system automatically distributes requests to multiple pod replicas of a Deployment.

A Deployment integrates a lot of functions, including online deployment, rolling upgrade, replica creation, and restoration of online jobs. To some extent, Deployments can be used to realize unattended rollout, which greatly reduces communication difficulties and operation risks in the rollout process.

**Figure 3-4 Deployment**

![Diagram of Deployment]

Creating a Deployment

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

**Step 2** Configure basic information.

- **Workload Name**
Enter 1 to 63 characters starting and ending with a letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Consecutive periods are not allowed, and a period cannot follow or be followed by a hyphen.

- **Namespace**
  Select a namespace. If a namespace is not available, create one by following the procedure provided in 2 Namespace.

- **Description**
  Enter a description, which cannot exceed 250 characters.

- **Pods**
  Specify the number of pods. A workload can have one or more pods. Each pod consists of one or more containers with the same specifications. Configuring multiple pods for a workload ensures high reliability. If one pod is faulty, the workload can still run properly.

- **Pod Specifications**
  You can select GPU-accelerated and allocate GPUs to the workload only if the namespace is of the GPU-accelerated type.

Currently, three types of pods are provided, including general-computing (used in general-computing namespaces), RDMA-accelerated, and GPU-accelerated (used in GPU-accelerated namespaces).

At present, GPU-accelerated pods support the following GPUs: NVIDIA Tesla V100 32GB, NVIDIA Tesla V100 16GB, and NVIDIA Tesla P4 8GB.

- Specifications of NVIDIA Tesla V100 32GB are as follows:
  - NVIDIA Tesla V100 32GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 32GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 32GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 32GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla V100 16GB are as follows:
  - NVIDIA Tesla V100 16GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 16GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 16GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 16GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla P4 8GB are as follows:
  - NVIDIA Tesla P4 8GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla P4 8GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla P4 8GBx3, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla P4 8GBx4, 32 CPU cores, 256 GB memory

CCI supports NVIDIA GPU drivers 396.26 and 410.104. The CUDA toolkit used in your application must meet the requirements listed in Table 3-1. For details about the compatibility between CUDA toolkits and drivers, see CUDA Compatibility at https://www.nvidia.com.
Table 3-1 Compatibility between NVIDIA GPU drivers and CUDA toolkits

<table>
<thead>
<tr>
<th>NVIDIA GPU Driver Version</th>
<th>CUDA Toolkit Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>396.26</td>
<td>CUDA 9.2 (9.2.88) or earlier</td>
</tr>
<tr>
<td>410.104</td>
<td>CUDA 10.0 (10.0.130) or earlier</td>
</tr>
</tbody>
</table>

If the pod type is not GPU-accelerated, the container specifications you select must meet the following requirements:

- **Configure Container**
  
  A pod generally contains only one container, but it can also contain multiple containers that use different images. If your application requires multiple containers in a pod, click **Add Container** and then select an image.

---

**NOTICE**

If different containers in a pod listen to the same port, a port conflict will occur and the pod may fail to start. For example, if an Nginx container (which listens to port 80) has been added to a pod, a port conflict will occur when another HTTP container in the pod tries to listen to port 80.

- **My Images**: images you have uploaded to SWR
- **Official Docker Hub Images**: public images on Docker Hub
- **Shared Images**: images shared by others through SWR

When an image is selected, you need to set the image version, container name, and number of CPU and memory resources that can be used by the container. (The minimum configuration of a container is 0.25 CPU cores and 0.2 GB memory.)

In a GPU-accelerated pod (available only in GPU-accelerated namespaces), only one container can use GPUs. If there are multiple containers in your pod, you can specify the container to use GPUs by enabling the GPU option.

You can also configure the following advanced settings for a container:

- **Storage**: You can mount persistent volumes to containers to persist data files. Currently, Elastic Volume Service (EVS) volumes, Scalable File Service (SFS) volumes, and SFS Turbo volumes are supported. Click the **EVS Volumes**, **SFS Volumes**, or **SFS Turbo Volumes** tab, and set the volume name, capacity, container path, and disk type. After the workload is created, you can manage storage volumes. For details, see **5.2 EVS Disk, 5.3 SFS File System**, or **5.4 SFS Turbo File System**.

- **Log collection**: Workload logs can be collected to the path you set and the size of log files can be limited. Click **Add Log Storage**, enter a container path for storing logs, and set the upper limit of log file size. After the workload is created, you can view workload logs on the AOM console. For details, see **7 Log Management**.

- **Environment variables**: You can manually set environment variables or add variable references. Environment variables add flexibility to workload configuration.
environment variables for which you have assigned values during container creation will take effect when the container is running. 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 create a secret reference, see 6.2 Using a Secret.

- Health check: Container health can be checked regularly during container running. For details about how to configure health checks, see 3.8 Health Check.
- Lifecycle: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. For details about how to configure the scripts, see 3.7 Container Lifecycle Hook.
- Startup commands: You can set the commands to be executed immediately after the container is started. Startup commands correspond to Docker's ENTRYPOINT startup instructions. For details, see 3.6 Setting Container Startup Command.
- Configuration management: You can add ConfigMaps and secrets to a container. For details about how to create ConfigMaps and secrets, see 6.1 Using a ConfigMap and 6.2 Using a Secret.

Step 3 Click Next to configure access information.

Three options are available:

- **Do not use**: No entry is provided to allow access from other workloads. This mode is suited for scenarios where custom service discovery is used or where access entry is not required.
- **Intranet access**: A domain name or internal domain name/virtual IP address is configured for the current workload so that this workload can provide services for other workloads in an internal network. Two internal network access modes are available: Service and ELB. For details about the internal network access, see 4.2 Private Network Access.
- **Internet access**: An entry is provided to allow access from the Internet. HTTP, HTTPS, TCP, and UDP are supported. For details about the public network access, see 4.3 Public Network Access.

Step 4 After configuration is complete, click Submit and then Back to Deployment List.

In the workload list, if the workload status is Running, the workload is created successfully. You can click the workload name to view workload details and press F5 to view the real-time workload status.

To access the workload, click the Access Settings tab to view the access address.

---End

**Deleting a Pod**

After the workload is created, you can manually delete pods. As pods are controlled by a controller, a 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.

In the pod list, click Delete for the target pod, as shown in Figure 3-5.
3.3 Job

A job is responsible for batch processing of short lived one-off tasks, that is, tasks that are executed only once. It ensures that one or more pods are successfully completed.

A job is a resource object that Kubernetes uses to control batch tasks. Batch jobs are different from long-term servo jobs (such as Deployment). The former can be started and terminated at specific time, while the latter runs uneasingly unless it is terminated. The pods managed by a job will be automatically removed after successfully completing tasks based on user configurations.

This run-to-completion feature of jobs is especially suitable for one-off tasks, such as continuous integration (CI). It works with the per-second billing of CCI to implement pay-per-use in real sense.

Creating a Job

Step 1  Log in to the CCI console. In the navigation pane, choose Workloads > Jobs. On the page displayed on the right, click Create Job.

Step 2  Configure basic information.

- Job Name
  Enter 1 to 63 characters starting and ending with a letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Consecutive periods are not allowed, and a period cannot follow or be followed by a hyphen.

- Namespace
  Select a namespace. If a namespace is not available, create one by referring to 2 Namespace.

- Description
Pod Specifications

You can select **GPU-accelerated** and allocate GPUs to the workload only if the namespace is of the GPU-accelerated type.

Currently, three types of pods are provided, including general-computing (used in general-computing namespaces), **RDMA-accelerated**, and GPU-accelerated (used in GPU-accelerated namespaces).

At present, GPU-accelerated pods support the following GPUs: NVIDIA Tesla V100 32GB, NVIDIA Tesla V100 16GB, and NVIDIA Tesla P4 8GB.

- Specifications of NVIDIA Tesla V100 32GB are as follows:
  - NVIDIA Tesla V100 32GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 32GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 32GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 32GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla V100 16GB are as follows:
  - NVIDIA Tesla V100 16GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 16GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 16GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 16GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla P4 8GB are as follows:
  - NVIDIA Tesla P4 8GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla P4 8GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla P4 8GBx3, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla P4 8GBx4, 32 CPU cores, 256 GB memory

CCI supports NVIDIA GPU drivers **396.26** and **410.104**. The CUDA toolkit used in your application must meet the requirements listed in Table 3-2. For details about the compatibility between CUDA toolkits and drivers, see CUDA Compatibility at https://www.nvidia.com.

<table>
<thead>
<tr>
<th>NVIDIA GPU Driver Version</th>
<th>CUDA Toolkit Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>396.26</td>
<td>CUDA 9.2 (9.2.88) or earlier</td>
</tr>
<tr>
<td>410.104</td>
<td>CUDA 10.0 (10.0.130) or earlier</td>
</tr>
</tbody>
</table>

If the pod type is not GPU-accelerated, the container specifications you select must meet the following requirements:

- **Configure Container**
A pod generally contains only one container, but it can also contain multiple containers that use different images. If your application requires multiple containers in a pod, click **Add Container** and then select an image.

**NOTICE**

If different containers in a pod listen to the same port, a port conflict will occur and the pod may fail to start. For example, if an Nginx container (which listens to port 80) has been added to a pod, a port conflict will occur when another HTTP container in the pod tries to listen to port 80.

- **My Images**: images you have uploaded to SWR
- **Official Docker Hub Images**: public images on Docker Hub
- **Shared Images**: images shared by others through SWR

When an image is selected, you need to set the image version, container name, and number of CPU and memory resources that can be used by the container. (The minimum configuration of a container is 0.25 CPU cores and 0.2 GB memory.)

In a GPU-accelerated pod (available only in GPU-accelerated namespaces), only one container can use GPUs. If there are multiple containers in your pod, you can specify the container to use GPUs by enabling the GPU option.

You can also configure the following advanced settings for a container:

- **Storage**: You can mount persistent volumes to containers to implement persistent storage of data files. Currently, EVS disks, SFS file systems, and SFS Turbo file systems are supported on the console. Click the **EVS Volumes**, **SFS Volumes**, or **SFS Turbo Volumes** tab, and set the volume name, capacity, container path, and disk type. After the workload is created, you can manage storage volumes. For details, see **5.2 EVS Disk**, **5.3 SFS File System**, or **5.4 SFS Turbo File System**.

- **Log collection**: Workload logs can be collected to the path you set and the size of log files can be limited. Click **Add Log Storage**, enter a container path for storing logs, and set the upper limit of log file size. After the workload is created, you can view workload logs on the AOM console. For details, see **7 Log Management**.

- **Environment variables**: You can manually set environment variables or add variable references. Environment variables add flexibility to workload configuration. The environment variables for which you have assigned values during container creation will take effect when the container is running. 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 create a secret reference, see **6.2 Using a Secret**.

- **Liveness probe**: You can configure a liveness probe for customized health checking of the container. If the container fails the check, the CCI will stop the container and determine whether to restart the container based on the restart policy. For details about how to configure a liveness probe, see **3.8 Health Check**.

- **Lifecycle**: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. For details about how to configure the scripts, see **3.7 Container Lifecycle Hook**.
- Startup commands: You can set the commands to be executed immediately after the container is started. Startup commands correspond to Docker's ENTRYPOINT startup instructions. For details, see 3.6 Setting Container Startup Command.
- Configuration management: You can add ConfigMaps and secrets to a container. For details about how to create ConfigMaps and secrets, see 6.1 Using a ConfigMap and 6.2 Using a Secret.

Step 3 Configure advanced job settings.

Jobs can be classified into one-off jobs and custom jobs.

- One-off job: A one-off job creates one pod each time. The job is completed when the pod is successfully executed.
- Custom job: You can set the number of executions and the number of concurrent executions for a custom job. Times Executed specifies the number of pods that need to be successfully executed until the job is completed. Parallel Jobs specifies the maximum number of pods that can run concurrently during the execution of the job. The number of parallel jobs should be less than the times executed.

You can set the timeout period for the job. When the job execution duration exceeds the timeout period, the job will be identified as failed, and all pods under this job will be deleted. If this parameter is left blank, the job will never time out.

Step 4 Click Next, click Submit, and then click Back to Job List.

If the job status is Running, the job is created successfully. You can click the job name to view job details and press F5 to view the real-time job status.

---End

Creating a Job Through kubectl

For details, see Creating a Job.

3.4 Cron Job

A Cron job runs a job on a specified schedule. A Cron job object is similar to a line of a crontab file in Linux.

Creating a Cron Job

Step 1 Log in to the CCI console. In the navigation pane, choose Workloads > Cron Jobs. On the page displayed on the right, click Create Cron Job.

Step 2 Configure basic information.

- **Job Name**
  Enter 1 to 63 characters starting and ending with a letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Consecutive periods are not allowed, and a period cannot follow or be followed by a hyphen.

- **Namespace**
  Select a namespace. If a namespace is not available, create one by following the procedure provided in 2 Namespace.
Description
Enter a description, which cannot exceed 250 characters.

Pod Specifications
You can select **GPU-accelerated** and allocate GPUs to the workload only if the namespace is of the GPU-accelerated type.

Currently, three types of pods are provided, including general-computing (used in general-computing namespaces), **RDMA**-accelerated, and GPU-accelerated (used in GPU-accelerated namespaces).

At present, GPU-accelerated pods support the following GPUs: NVIDIA Tesla V100 32GB, NVIDIA Tesla V100 16GB, and NVIDIA Tesla P4 8GB.

- Specifications of NVIDIA Tesla V100 32GB are as follows:
  - NVIDIA Tesla V100 32GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 32GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 32GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 32GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla V100 16GB are as follows:
  - NVIDIA Tesla V100 16GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 16GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 16GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 16GBx8, 32 CPU cores, 256 GB memory

- Specifications of NVIDIA Tesla P4 8GB are as follows:
  - NVIDIA Tesla P4 8GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla P4 8GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla P4 8GBx3, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla P4 8GBx4, 32 CPU cores, 256 GB memory

CCI supports NVIDIA GPU drivers 396.26 and 410.104. The CUDA toolkit used in your application must meet the requirements listed in Table 3-3. For details about the compatibility between CUDA toolkits and drivers, see CUDA Compatibility at https://www.nvidia.com.

<table>
<thead>
<tr>
<th>NVIDIA GPU Driver Version</th>
<th>CUDA Toolkit Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>396.26</td>
<td>CUDA 9.2 (9.2.88) or earlier</td>
</tr>
<tr>
<td>410.104</td>
<td>CUDA 10.0 (10.0.130) or earlier</td>
</tr>
</tbody>
</table>

If the pod type is not GPU-accelerated, the container specifications you select must meet the following requirements:

- Configure Container
A pod generally contains only one container, but it can also contain multiple containers that use different images. If your application requires multiple containers in a pod, click **Add Container** and then select an image.

**NOTICE**

If different containers in a pod listen to the same port, a port conflict will occur and the pod may fail to start. For example, if an Nginx container (which listens to port 80) has been added to a pod, a port conflict will occur when another HTTP container in the pod tries to listen to port 80.

- **My Images**: images you have uploaded to SWR
- **Official Docker Hub Images**: public images on Docker Hub
- **Shared Images**: images shared by others through SWR

When an image is selected, you need to set the image version, container name, and number of CPU and memory resources that can be used by the container. (The minimum configuration of a container is 0.25 CPU cores and 0.2 GB memory.)

In a GPU-accelerated pod (available only in GPU-accelerated namespaces), only one container can use GPUs. If there are multiple containers in your pod, you can specify the container to use GPUs by enabling the GPU option.

You can also configure the following advanced settings for a container:

- **Log collection**: Workload logs can be collected to the path you set and the size of log files can be limited. Click **Add Log Storage**, enter a container path for storing logs, and set the upper limit of log file size. After the workload is created, you can view workload logs on the AOM console. For details, see **7 Log Management**.
- **Environment variables**: You can manually set environment variables or add variable references. Environment variables add flexibility to workload configuration. The environment variables for which you have assigned values during container creation will take effect when the container is running. 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 create a secret reference, see **6.2 Using a Secret**.

- **Liveness probe**: You can configure a liveness probe for customized health checking of the container. If the container fails the check, the CCI will stop the container and determine whether to restart the container based on the restart policy. For details about how to configure a liveness probe, see **3.8 Health Check**.

- **Lifecycle**: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. For details about how to configure the scripts, see **3.7 Container Lifecycle Hook**.

- **Startup commands**: You can set the commands to be executed immediately after the container is started. Startup commands correspond to Docker's **ENTRYPOINT** startup instructions. For details, see **3.6 Setting Container Startup Command**.

- **Configuration management**: You can add ConfigMaps and secrets to a container. For details about how to create ConfigMaps and secrets, see **6.1 Using a ConfigMap** and **6.2 Using a Secret**.
Step 3 Configure advanced job settings.

- **Concurrency Policy**
  - **Forbid**: A new job cannot be created before the previous job is completed.
  - **Allow**: New jobs can be created continuously.
  - **Replace**: A new job replaces the previous job when it is time to create a job but the previous job is not completed.

- **Timing Rule**: Set the schedule based on which the job is executed.

- **Job Record**: Set the number of records to be retained for successful jobs and failed jobs.

Step 4 Click Next, click Submit, and then click **Back to Cron Job List**.

If the job status is **Running**, the Cron job is created successfully. You can click the job name to view job details and press **F5** to view the real-time job status.

---End

Creating a Cron Job Through kubectl

For details, see **Creating a Cron Job**.

### 3.5 Viewing Resource Usage

After you have created a workload, you may want to know the resource usage rates of each pod.

CCI allows you to monitor the CPU or CPU usage and memory usage of each pod. Go to the details page of a Deployment, job, or cron job, and click Expand for a pod in the pod list. On the Monitoring tab page, view the resource usage rates, as shown in **Figure 3-7**. You can also view the resource usage rates of each pod by choosing **Workload Management > Pod** in the navigation pane.

**Figure 3-7** Viewing monitoring data

![Figure 3-7](image)

### 3.6 Setting Container 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 may configure or initialize MySQL databases before running MySQL servers. You can set **ENTRYPOINT** or **CMD** in the...
Dockerfile when creating 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 fails to be started.

In CCI, 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 creating a workload, as shown in the following figure.

![Figure 3-8 Startup command](image)

When Docker runs, only one ENTRYPOINT command is supported. The startup command set in CCI will overwrite the ENTRYPOINT and CMD commands set in Dockerfile during image creation. The following table lists the rules.

<table>
<thead>
<tr>
<th>Image Entrypoint</th>
<th>Image CMD</th>
<th>Command for Running a Container</th>
<th>Parameter for Running a Container</th>
<th>Command Executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>[touch]</td>
<td>[/root/test]</td>
<td>Not set</td>
<td>Not set</td>
<td>[touch /root/test]</td>
</tr>
<tr>
<td>[touch]</td>
<td>[/root/test]</td>
<td>[mkdir]</td>
<td>Not set</td>
<td>[mkdir]</td>
</tr>
<tr>
<td>[touch]</td>
<td>[/root/test]</td>
<td>Not set</td>
<td>[/opt/test]</td>
<td>[touch /opt/test]</td>
</tr>
<tr>
<td>[touch]</td>
<td>[/root/test]</td>
<td>[mkdir]</td>
<td>[/opt/test]</td>
<td>[mkdir /opt/test]</td>
</tr>
</tbody>
</table>
3.7 Container Lifecycle Hook

Setting a Container Lifecycle Hook

Based on Kubernetes, CCI provides containers with lifecycle hooks. The hooks enable containers to run code triggered by events during their management 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:

- **Post-Start Processing**: triggered immediately after the container is started
- **Pre-Stop Processing**: triggered immediately before the container is stopped

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

During workload creation, expand the **Advanced Settings** area, and click the **Post-Start Processing** or **Pre-Stop Processing** tab in the **Lifecycle** area.

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

**Figure 3-9** Command settings

![Command settings](image)

**Setting a Container Lifecycle Hook Through kubectl**

For details, see **Lifecycle Management**.

3.8 Health Check

Container health can be checked regularly during container running.

CCI provides two health check methods based on Kubernetes:

- **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
containerized application fails the check, the container will be restarted. If the containerized application passes the check, no operation will be performed.

- Readiness probe: checks whether a containerized application is ready to start handling requests. An application may take a long time to start up and provide services, for example because 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 comes in.

**Health Check Modes**

- **HTTP GET 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.

- **Command Line Script**
  The probe runs a command in the container and checks the exit status code. If the exit status code is 0, the probe is healthy.

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

  ![Figure 3-10 Command setting](image)

  **Common Parameter Description**

  **Table 3-4 Health check parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Window</td>
<td>Delay time of the check (unit: second). For example, if this parameter is set to 10, the probe starts 10 seconds after the container is started.</td>
</tr>
<tr>
<td>Timeout Time</td>
<td>Timeout period of the check (unit: second). For example, if this parameter is set to 10, the timeout period of a health check is 10 seconds. If the timeout period is exceeded, the health check is regarded as a failure. If this parameter is set to 0 or left empty, the timeout period will be 1 second.</td>
</tr>
</tbody>
</table>
Setting a Health Check Through kubectl

- For details about how to set the liveness probe, see Liveness Probe.
- For details about how to set the readiness probe, see Readiness Probe.

3.9 Web-Terminal

The web-terminal provides the container connection function to help you quickly debug the container.

Constraints and Restrictions

- The web-terminal logs in to the container by using sh shell by default. Therefore, the container must support sh shell.
- Only running containers can be logged in to by using the web-terminal.
- You need to enter exit in the web-terminal during exit; otherwise, the sh process will remain.

Connecting to the Container by Using the Web-terminal

**Step 1** Log in to the CCI console. In the navigation pane, choose Workloads > Deployments. On the page displayed on the right, click the workload to be accessed.

**Step 2** In the Pod List area of the workload details page, click the arrow icon at the left of the pod and then click the CLI tab.

When # is displayed, you have logged in to the container.

![Container CLI](image)

---End

3.10 Upgrading a Workload

A workload can be updated and upgraded after being created.

Currently, rolling upgrade is supported. Rolling upgrade will gradually replace the workload of the old version with that of the new version. During the upgrade process, service traffic
will be balanced and distributed to the new and old workloads at the same time, so the service will not be interrupted.

Upgrading a Workload

**Step 1** Log in to the CCI console. In the navigation pane, choose Workloads > Deployments. On the page displayed on the right, click the name of the workload to be upgraded. Then, click Upgrade in the upper right corner of the workload details page.

**Step 2** Modify pod specifications.

You can select **GPU-accelerated** and allocate GPUs to the workload only if the namespace is of the GPU-accelerated type.

Currently, three types of pods are provided, including general-computing (used in general-computing namespaces), RDMA-accelerated, and GPU-accelerated (used in GPU-accelerated namespaces).

At present, GPU-accelerated pods support the following GPUs: NVIDIA Tesla V100 32GB, NVIDIA Tesla V100 16GB, and NVIDIA Tesla P4 8GB.

- **Specifications of NVIDIA Tesla V100 32GB are as follows:**
  - NVIDIA Tesla V100 32GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 32GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 32GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 32GBx8, 32 CPU cores, 256 GB memory

- **Specifications of NVIDIA Tesla V100 16GB are as follows:**
  - NVIDIA Tesla V100 16GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla V100 16GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla V100 16GBx4, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla V100 16GBx8, 32 CPU cores, 256 GB memory

- **Specifications of NVIDIA Tesla P4 8GB are as follows:**
  - NVIDIA Tesla P4 8GBx1, 4 CPU cores, 32 GB memory
  - NVIDIA Tesla P4 8GBx2, 8 CPU cores, 64 GB memory
  - NVIDIA Tesla P4 8GBx3, 16 CPU cores, 128 GB memory
  - NVIDIA Tesla P4 8GBx4, 32 CPU cores, 256 GB memory

If the pod type is not GPU-accelerated, the container specifications you select must meet the following requirements:

**Step 3** Modify container settings.

1. Click **Change Image** to select a new image.
When an image is selected, you need to set the image version, container name, and number of CPU and memory resources that can be used by the container. (The minimum configuration of a container is 0.25 CPU cores and 0.2 GB memory.) Only one container in a pod can use GPUs. If your pod has multiple containers, you can specify the container that can use GPUs by enabling the GPU option.

You can also configure the following advanced settings for a container:

- **My Images**: images you have uploaded to SWR
- **Official Docker Hub Images**: public images on Docker Hub
- **Shared Images**: images shared by others through SWR

When an image is selected, you need to set the image version, container name, and number of CPU and memory resources that can be used by the container. (The minimum configuration of a container is 0.25 CPU cores and 0.2 GB memory.) Only one container in a pod can use GPUs. If your pod has multiple containers, you can specify the container that can use GPUs by enabling the GPU option.

You can also configure the following advanced settings for a container:

- **Storage**: You can mount persistent volumes to containers to implement persistent storage of data files. Currently, EVS disks, SFS file systems, and SFS Turbo file systems are supported on the console. Click the **EVS Volumes**, **SFS Volumes**, or **SFS Turbo Volumes** tab, and set the volume name, capacity, container path, and disk type. After the workload is created, the storage volumes can be managed. For details, see **5.2 EVS Disk**, **5.3 SFS File System**, or **5.4 SFS Turbo File System**.
- **Log collection**: Workload logs can be collected to the path you set and the size of log files can be limited. Click **Add Log Storage**, enter a container path for storing logs, and set the upper limit of log file size. After the workload is created, you can view workload logs on the AOM console. For details, see **7 Log Management**.
- **Environment variables**: You can manually set environment variables or add variable references. Environment variables add flexibility to workload configuration. The environment variables for which you have assigned values during container creation
will take effect when the container is running. 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 create a secret reference, see 6.2 Using a Secret.

- Health check: Container health can be checked regularly during container running. For details about how to configure health checks, see 3.8 Health Check.
- Lifecycle: Lifecycle scripts specify actions that applications take when a lifecycle event occurs. For details about how to configure the scripts, see 3.7 Container Lifecycle Hook.
- Startup commands: You can set the commands to be executed immediately after the container is started. Startup commands correspond to Docker's ENTRYPOINT startup instructions. For details, see 3.6 Setting Container Startup Command.
- Configuration management: You can add ConfigMaps and secrets to a container. For details about how to create ConfigMaps and secrets, see 6.1 Using a ConfigMap and 6.2 Using a Secret.

**Step 4** Click Next.

**Step 5** Click Submit to upgrade the workload.

----End

**Upgrading a Workload Through kubectl**

For details about how to use kubectl to upgrade a workload, see "Upgrading a Deployment" in Deployment.

**3.11 Scaling a Workload**

This section describes two workload scaling methods: auto scaling and manual scaling. You can select a scaling method as required.

- Auto scaling: Supports metric-based, scheduled, and periodic policies. When the configuration is complete, pods can be automatically added or deleted based on resource changes or a specified schedule.
- Manual scaling: Scales the number of pods immediately after the configuration is complete.

---

**NOTICE**

If an EVS disk is mounted to a pod, the EVS disk will not be deleted when the pod is deleted. If a new pod with the same name as the deleted pod is created, no EVS disks can be mounted to the new pod.
Auto Scaling

**NOTE**
Currently, auto scaling is supported only for Deployments.

A properly configured auto scaling policy eliminates the need to manually adjust resources in response to service changes and traffic peaks, helping you reduce manpower and resource consumption. Currently, CCI supports the following types of auto scaling policies:

**Metric-based**: Scales the workload based on CPU/memory usage. You can specify a CPU/memory usage threshold. If the usage is higher or lower than the threshold, instances can be automatically added or deleted.

**Scheduled**: Scales the workload at a specified time. A scheduled scaling policy is suited for scenarios such as flash sales and anniversary promotions.

**Periodic**: Scales the workload on a daily, weekly, or monthly basis. A periodic scaling policy is suited for applications that have periodic traffic changes.

- **Configure a metric-based auto scaling policy.**
  a. Log in to the CCI console. In the navigation pane, choose **Workloads > Deployments**. On the page displayed on the right, click the name of the target Deployment.
  b. In the **Scaling** area of the Deployment details page, click **Auto Scaling** and then click **Add Scaling Policy**.

**Figure 3-13 Adding a metric-based auto scaling policy**

### Adding Scaling Policy

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter 1 to 64 characters starting with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.</td>
<td></td>
</tr>
<tr>
<td>Policy Type</td>
<td>Metric-based policy</td>
</tr>
<tr>
<td>Trigger Condition</td>
<td>CPU usage</td>
</tr>
<tr>
<td>Duration</td>
<td>60s</td>
</tr>
<tr>
<td>Consecutive Times</td>
<td>3</td>
</tr>
<tr>
<td>Policy Action</td>
<td>Add</td>
</tr>
</tbody>
</table>

**Figure 3-13 Adding a metric-based auto scaling policy**
Table 3-5 Parameters of a metric-based auto scaling policy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>Name of a policy.</td>
</tr>
<tr>
<td>Policy Type</td>
<td>Select <strong>Metric-based policy</strong>.</td>
</tr>
<tr>
<td>Trigger Condition</td>
<td>Select <strong>CPU usage</strong> or <strong>Memory usage</strong>. If you set the trigger condition to average memory usage &gt; 70%, the scaling policy will be triggered when the average memory usage exceeds 70%.</td>
</tr>
<tr>
<td>Duration</td>
<td>Statistical period. Select a value from the drop-down list. If the value is set to 60, metric statistics are collected every 60 seconds.</td>
</tr>
<tr>
<td>Consecutive Times</td>
<td>If this parameter is set to 3, the configured action will be triggered when the threshold is reached for 3 consecutive statistical periods.</td>
</tr>
<tr>
<td>Policy Action</td>
<td>Action to be executed when the policy is triggered. The action can be to add or reduce the number of instances.</td>
</tr>
</tbody>
</table>

c. Click **Confirm**. The policy is added to the policy list, and its status is **Enabled**.

**Figure 3-14 Policy enabled**

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

- Configure a scheduled auto scaling policy.
  a. In the **Scaling** area, click **Auto Scaling** and then click **Add Scaling Policy**.
**Figure 3-15** Adding a scheduled auto scaling policy

**Adding Scaling Policy**

- **Policy Name**: Enter a policy name. Enter 1 to 64 characters starting with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.
- **Policy Type**: Select **Scheduled Policy**.
- **Triggered**: 2018-08-29 14:21
  - The time interval between scheduled or periodic scaling policies must be longer than 1 minute.
- **Policy Action**: Add

**Table 3-6** Parameters of a scheduled auto scaling policy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>Name of a policy.</td>
</tr>
<tr>
<td>Policy Type</td>
<td>Select <strong>Scheduled Policy</strong>.</td>
</tr>
<tr>
<td>Triggered</td>
<td>Time when the policy is triggered.</td>
</tr>
<tr>
<td>Policy Action</td>
<td>Action to be executed when the policy is triggered. The action can be to add or reduce the number of instances.</td>
</tr>
</tbody>
</table>

b. Click **Confirm**.

The policy is added to the policy list, and its status is **Enabled**.

Configure a periodic auto scaling policy.

a. In the **Scaling** area, click **Auto Scaling** and then click **Add Scaling Policy**.
Figure 3-16 Adding a periodic auto scaling policy

Adding Scaling Policy

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Enter a policy name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Type</td>
<td>Metric-based policy, Scheduled policy, Periodic policy</td>
</tr>
</tbody>
</table>

Table 3-7 Parameters of a periodic auto scaling policy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>Name of a policy.</td>
</tr>
<tr>
<td>Policy Type</td>
<td>Select Periodic Policy.</td>
</tr>
<tr>
<td>Select Time</td>
<td>Time range and specific time when the policy is triggered.</td>
</tr>
<tr>
<td>Policy Action</td>
<td>Action to be executed when the policy is triggered.</td>
</tr>
</tbody>
</table>

b. Click Confirm.
   The policy is added to the policy list, and its status is Enabled.

Manual Scaling

Step 1  Log in to the CCI console. In the navigation pane, choose Workloads > Deployments. On the page displayed on the right, click the name of the target Deployment.

Step 2  In the Scaling area, click Manual Scaling. Click ✍️ and change the number of instances (for example, change the value to 3), and then click Save. The scaling takes effect immediately.

CCI provides a time window for executing pre-stop processing commands before an application is deleted. If a command process is still running when the time window expires, the application will be forcibly deleted.
Step 3  In the pod list, you can see that new instances are being created. When the status of all added instances becomes **Running**, the scaling is completed successfully.

---End
4 Load Network Access

4.1 Network Access Overview

Workload access scenarios can be categorized as follows:

- **4.2 Private Network Access**: Access to internal HUAWEI CLOUD resources
  - **Service**: This mode is used when workloads in the same namespace need to access each other.
  - **ELB** (private network): This mode is used when non-CCI resources (such as ECSs) in the same VPC as the current workload need to access each other. In addition, this mode can be used when workloads in the same VPC but different namespaces need to access each other. In this mode, the current workload can be accessed by other workloads in the internal network using *Internal domain name* or *ELB IP address:ELB port*. The HTTP/HTTPS and TCP/UDP protocols are supported. If non-CCI resources are in a VPC different from the current workload, you can also create a VPC **peer connection** to achieve communication.

- **4.3 Public Network Access**: Bind an ELB to the workload (the ELB must be in the same VPC as the workload) and access the workload from the public network through the ELB.

- **4.4 Accessing Public Networks from a Container**: Configure SNAT rules through the **NAT Gateway** service so that containers can access public networks.
4.2 Private Network Access

The access scenarios are divided as follows:

- **Workload Access Through a Service**: This mode is used when workloads in the same CCI namespace need to access each other.

- **Workload Access Through a Private Network Load Balancer**: This mode is used when non-CCI resources (such as ECSs) in the same VPC as the current workload need to access each other. In addition, this mode can be used when workloads in the same VPC but different namespaces need to access each other. In this mode, the current workload can be accessed by other workloads in the internal network using `Internal domain name` or `ELB IP address:ELB port`. The HTTP/HTTPS and TCP/UDP protocols are supported. If non-CCI resources are in a VPC different from the current workload, you can also create a VPC **peer connection** to achieve communication.

**Workload Access Through a Service**

The smallest resource unit in the workload is the pod, and accessing a workload is to access the pods in the workload. Pods in a workload can be dynamically created and destroyed, for example, during capacity scaling or rolling upgrade. In this case, the pod addresses will change, which makes it inconvenient to access pods.

To solve this problem, CCI provides the coredns (internal domain name resolution) add-on. Changes of pods are managed by the workload and not perceived externally.

The workload can be accessed using `Service name:Workload access port`, where the workload access port is mapped to the container port. As shown in the following figure, if the pod in the frontend needs to access the pod in the backend, the former only needs to access `nginx:8080`. 

---

**Figure 4-1 Network access diagram**

1. A workload accesses another workload in the same namespace using a service.
2. A workload accesses another workload in the same VPC using a private network load balancer.
3. A workload accesses another workload in a peer VPC using a private network load balancer.
4. An external load balancer accesses a workload using a public network load balancer.
5. A workload accesses the Internet using a NAT gateway.
To enable the Service name:Workload access port access mode, configure the following parameters when creating the workload:

- **Service Name**: Specifies the name of the service, which is an object for managing pod access. For details about services, see Service.
- **Install coredns**: Specifies whether to install the coredns add-on. Install the coredns add-on so that it can provide internal domain name resolution for other workloads.
- **Workload Port Settings**:
  - **Protocol**: Specifies the communication protocol used to access the workload. Select TCP or UDP.
  - **Workload Access Port**: Specifies the port for accessing the workload.
  - **Container Port**: Specifies the port to which the container listens. The workload access port will be mapped to the container port.
Figure 4-3 Configuring service access parameters

Creating a Service Through `kubectl`

For details, see Service.

Workload Access Through a Private Network Load Balancer

For non-CCI resources, if they are in a namespace different from the current workload and need to access the workload, bind an enhanced load balancer of the Private network type to the workload, and use the virtual IP address of the load balancer to access the workload.

Select the load balancer when configuring the access mode. For configuration details, see 4.3 Public Network Access.

Figure 4-4 Workload access through a private network load balancer
4.3 Public Network Access

Public network access means accessing a workload over the Internet. You can bind an enhanced load balancer to the workload (the load balancer must be in the same VPC as the workload) so that you can access the workload through the load balancer. Currently, layer 4 and layer 7 public network access are both supported.

- TCP and UDP are supported for layer 4 public network access. After configuration is complete, the workload can be accessed using Public network IP address of the load balancer:Load balancer port.

- HTTP and HTTPS are supported for layer 7 public network access. After configuration is complete, the workload can be accessed using http://Public network domain name or public network IP address of the load balancer:Load balancer port/Mapping path.

Services forward requests based on layer 4 TCP and UDP protocols. Ingresses forward requests based on layer 7 HTTP and HTTPS protocols. Domain names and paths can be used to achieve finer granularities, as shown in the following figure.

**Figure 4-5 Ingress-Service**

The following figure shows an example of accessing a workload over the HTTP protocol.
Figure 4-6 Public network access

Configuring Public Network Access

On the **Access Settings** page for creating a workload, select the **Internet Access** mode and configure the following parameters:

- **Service Name**: Specifies the name of the service, which is an object for managing pod access. For details about services, see **Service**.
- **Load Balancer**: Select an enhanced load balancer. If no enhanced load balancer is available, create one.

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The enhanced load balancer to be created must be in the same VPC as the workload.</td>
</tr>
</tbody>
</table>

- **ELB Protocol**: Communication protocol for public network access, which can be **HTTP/HTTPS** or **TCP/UDP**.
- **Ingress Name**: Specifies the name of the ingress, which is an object for managing Layer 7 access. If this parameter is not set, the workload name will be used as the ingress name by default. For details about ingresses, see Ingress.

- **Public Domain Name** (configurable when the HTTP/HTTPS protocol is used): To access the workload using a domain name, you need to purchase a public domain name and map the domain name to the EIP of the selected load balancer.

- **Certificate** (mandatory when the HTTPS protocol is selected): For details about how to import an SSL certificate, see 6.3 SSL Certificate.

- **ELB Port**: Select the protocol and port for accessing the workload using the load balancer.

- **Workload Port Protocol**: Communication protocol for accessing the workload, which can be TCP or UDP. If the ELB protocol is set to HTTP/HTTPS, the workload port protocol will be TCP.

- **Workload Port Settings**:
  - **Workload Access Port**: Port for accessing the workload.
  - **Container Port**: Port to which the container listens. The workload access port will be mapped to the container port.

- **HTTP Route Settings**:
  - **Mapping Path**: path to be accessed. It must start with a slash (/). For example, /api/web. It can also be the root path /.
  - **Workload Access Port**: Previously configured workload access port.

As shown in Figure 4-7, if the IP address of the load balancer is 10.10.10.10, you can access the workload by visiting http://10.10.10.10:6071/.
Figure 4-7 Configuring public network access parameters

Troubleshooting the Problem of Failing to Access a Workload from the Public Network

1. The premise of normal access from the public network is that the workload is in running state. If your workload is abnormal or not ready, it cannot be accessed properly from the 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 details page that is displayed, choose Access Settings to check whether any alarm events are reported. The following are two common events:
   - 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. You are advised to delete the workload and create it again or wait for 5 – 10 minutes until the Internet access can be normally used.
   - Create listener failed: This event occurs usually because that the listener quota is exceeded. Select another load balancer with a sufficient quota.

4. The workload is inaccessible 3 minutes after it is created, 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 lie in the load balancer. In this case, check the status of the load balancer.

**Enabling Public Network Access Through kubectl**

To enable the access to a workload from the public network, two Kubernetes objects (that is, service and ingress) are required. For details, see Service and Ingress.

**4.4 Accessing Public Networks from a Container**

You can use the NAT Gateway service to enable container instances 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. Figure 4-8 shows the SNAT architecture. The SNAT function allows the container instances in a VPC to access the Internet without being bound to an EIP. SNAT supports a large number of concurrent connections, which makes it suitable for applications involving a large number of requests and connections.
To enable a container instance to access the Internet, perform the following steps:

**Step 1** Buy an EIP.
1. Log in to the management console.
2. Select the desired region and project.
3. Choose **Service List > Network > Virtual Private Cloud**.
4. In the navigation pane, choose **Elastic IP**.
5. On the **Elastic IP** page, click **Buy EIP**.
6. Set parameters as required.

**NOTE**
Set **Region** to the region where container instances are located.
Figure 4-9 Buying an EIP

Step 2 Buy a NAT gateway. For details, see Buying a NAT Gateway.

1. Log in to the management console.
2. Select the desired region and project.
3. Choose Service List > Network > NAT Gateway.
4. On the displayed page, click Buy NAT Gateway.
5. Set parameters as required.

**NOTE**

Select the VPC and subnet that you have configured for the namespace of container instances.

Figure 4-10 Buying a NAT gateway

Step 3 Configure an SNAT rule and bind the EIP to the subnet. For details, see Adding an SNAT Rule.
1. Log in to the management console.
2. Select the desired region and project.
3. Choose Service List > Network > NAT Gateway.
4. On the displayed page, click the name of the NAT gateway for which you want to add the SNAT rule.
5. On the SNAT Rules tab page, click Add SNAT Rule.
6. Set parameters as required.

**NOTE**
Select the subnet that you have configured for the namespace of container instances.

**Figure 4-11** Adding an SNAT rule

After the SNAT rule is configured, workloads can access public networks from the container. As shown in the following figure, public networks can be pinged from the container.
Figure 4-12 Accessing public networks from a container

---End
5 Storage Management

5.1 Overview

CCI supports multiple types of persistent storage to meet your requirements in different scenarios. You can use the following types of storage when creating a workload:

- Elastic Volume Service (EVS) disk
  You can mount an EVS disk to a container path. When the container is migrated, the mounted EVS disk is migrated together. This storage type is suited for the scenario where data needs to be stored persistently. For details, see 5.2 EVS Disk.

- Scalable File Service (SFS) file system
  You can create an SFS file system and mount it to a container path. The file systems created by the underlying SFS service can also be used. SFS file systems are suited for workload scenarios where data needs to be stored persistently and read by and written to multiple nodes. Such scenarios include media processing, content management, big data analysis, and workload analysis. For details, see 5.3 SFS File System.

- SFS Turbo file system
  You can create an SFS turbo file system and mount it to a container path. SFS Turbo file systems are fast, on-demand, and scalable. They are suitable for DevOps, containerized microservices, and enterprise office applications. For details, see 5.4 SFS Turbo File System.

- Object Storage Service (OBS) bucket
  CCI allows you to use OBS through an SDK. You can configure OBS usage when defining an application using an SDK, package the application into an image, and then use the image to deploy a workload on CCI. For details, see 5.5 OBS.

PersistentVolumeClaim (PVC)

CCI uses PVCs to apply for and manage persistent storage. With PVCs, you only need to specify the type and capacity of storage resources without concerning about how to create and release underlying storage resources.

In practice, you can bind a PVC to the volume in a pod and use persistent storage through the PVC, as shown in Figure 5-1.
On the CCI console, you can import existing EVS disks, SFS file systems, and SFS Turbo file systems. When importing such a storage resource, CCI creates a PVC for the resource.

You can also purchase EVS disks and SFS file systems on the CCI console. After these storage resources are purchased, CCI will create PVCs for them and import them to the local.

5.2 EVS Disk

To meet data persistency requirements, CCI allows you to mount EVS disks to containers. By using an EVS disk, you can mount the remote file directory of a storage system to a container. Data in the data volume is saved permanently. Even if the container is deleted, only the mounted data volume is deleted, and the data is retained in the storage system.

The EVS supports three specifications: common I/O, high I/O, and ultra-high I/O.

- **Common I/O**: The back-end storage is provided by the SATA storage media. It is suitable for high-capacity application scenarios with low read/write rate requirements and less transaction processing, such as development testing and enterprise office applications.
- **High I/O**: The back-end storage is provided by the SAS storage medium. It is suitable for application scenarios with relatively high performance, high read/write rate requirements, and real-time data storage requirements, such as creating file systems and distributed file sharing.
- **Ultra-high I/O**: The back-end storage is provided by the SSD storage medium. It is suitable for application scenarios with high performance, high read/write rate requirements, and data-intensive requirements, such as NoSQL, relational database, and data warehouses (such as Oracle RAC and SAP HANA).

**Constraints**

- Only EVS disks that meet the following conditions in AZ 1, Beijing can be imported to CCI: (1) Can be shared and be of the SCSI type; (2) Not encrypted or exclusive; (3) Not billed on a yearly/monthly basis, or bound to ECS for sales.
- An EVS volume can be used only as a new disk. The content in the EVS volume that has never been mounted to CCI is invisible to the container.
- If an imported EVS disk is deleted from the EVS console, it cannot be perceived by CCI. You are advised to delete the EVS disk after confirming that it is not used by any workload.
- An EVS volume can be mounted to only one instance.

**Adding EVS Disks**

**Step 1** Log in to the CCI console. In the navigation pane, choose Storage > EVS.
If you have purchased EVS disks on the EVS console, go to Step 2.
If you have not purchased any EVS disk, go to Step 3.

Step 2 Click Import. On the Import EVS Disk page, select one or more EVS disks that you want to import and click Import.

**NOTE**

An EVS disk can be imported to only one namespace. If an EVS disk has been imported into a namespace, it is invisible in other namespaces and cannot be imported again.

After the EVS disk is imported, you can see the corresponding volume.

**Figure 5-2 Import result**

<table>
<thead>
<tr>
<th>PVC Name</th>
<th>Disk Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvc-test</td>
<td>pvc-09711a8b-f775-11e8-a40b-501...</td>
<td>🟢 EVS disk c...</td>
</tr>
</tbody>
</table>

Step 3 Click Buy EVS Volume. On the Buy EVS Volume page, set parameters, click Next, confirm the specifications, and click Submit.

- **PVC name**: name of a PVC.
- **Namespaces**: namespace to which the PVC belongs.
- **Type**: disk type, which can be common I/O, high I/O, or ultra-high I/O.
- **Capacity**: disk capacity, which ranges from 10 to 1000 GB.
- **AZ**: availability zone to which the disk belongs.
Using EVS Volumes

After selecting a container image during workload creation, choose Storage > EVS Volumes and click Add EVS Volume in the Advanced Settings area and mount the EVS volume to the container.

**Figure 5-3 Configuring EVS volume parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Capacity (GB)</th>
<th>Container Path</th>
<th>Disk type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc-evs-vol04th</td>
<td>10</td>
<td>/mnt/data</td>
<td>Ultra-high I/O</td>
</tr>
</tbody>
</table>

**NOTE**

For a Deployment, only one EVS volume will be created, which is shared by all pods in the Deployment.

After a workload is created, you can view the relationship between the EVS disk and the workload by choosing Storage > EVS.

**Figure 5-4 Managing EVS Volumes**

<table>
<thead>
<tr>
<th>PVC Name</th>
<th>Disk Name</th>
<th>Status</th>
<th>Workload</th>
<th>Capacity(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvc-test</td>
<td>pvc-09711a8b-775-11e6-a40b-501...</td>
<td></td>
<td>nginx</td>
<td>1Gi</td>
</tr>
</tbody>
</table>

Creating EVS Volumes Through kubectl

For details, see Using Persistent Storage.

5.3 SFS File System

CCI enables you to mount SFS file systems to containers. Currently, SFS file systems of only the Network File System (NFS) type are supported. SFS file systems are applicable to a wide range of scenarios, including media processing, content management, big data, and application analysis.

**Constraints**

- If an SFS file system is in use, do not modify the VPC configuration associated with the SFS file system. Otherwise, the containers in CCI cannot access the SFS file system.
- Exercise caution when deleting an SFS file system. After an SFS file system is deleted, containers in CCI are unavailable.

**Importing SFS File Systems**

CCI allows you to import existing SFS file systems.
Step 1  Log in to the CCI console. In the navigation pane, choose Storage > SFS.
- If you have created SFS file systems on the SFS console, go to Step 2.
- If you have not created any SFS file system, go to Step 3.

Step 2  Click Import. On the Import SFS File System page, select one or more file systems that you want to import and click Import.

Step 3  Click Create SFS Volume, set parameters, and click Create.

- PVC name: name of a PVC.
- Namespaces: namespace to which the PVC belongs.
- Type: type of the SFS file system. Currently, only NFS is supported.
- Total Capacity: required capacity.
- Access Mode: mode to access to the SFS volume. Currently, only ReadWriteMany is supported. To be specific, the SFS volume can be read by and written to multiple nodes.

---End

Using SFS Volumes

After selecting a container image during Deployment creation (3.2 Deployment) or job creation (Creating a Job), choose Storage > SFS Volumes and click Add SFS Volume in the Advanced Settings area and mount the SFS volume to the container.

Figure 5-5 Configuring SFS volume parameters

You can select automatically-created volumes or existing volumes. Before using existing volumes, import SFS file systems in advance. For details, see Importing SFS File Systems.
Do not mount SFS volumes to a system directory, such as / or /var/run. Otherwise, the container becomes abnormal. You are advised to mount SFS volumes to an empty directory. If the directory is not empty, ensure that there are no files affecting container startup in the directory. Otherwise, such files will be replaced, resulting in failures to start the container and create the workload.

If SFS volumes need to be mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container. Otherwise, high-risk files on the host machine may be damaged.

Creating SFS Volumes Through kubectl

For details, see Using Persistent Storage.

5.4 SFS Turbo File System

You can create an SFS Turbo file system and mount it to a container path. SFS Turbo file systems are fast, on-demand, and scalable. They are suitable for DevOps, containerized microservices, and enterprise office applications.

Importing SFS Turbo File Systems

CCI allows you to import existing SFS Turbo file systems.

Step 1  Log in to the CCI console. In the navigation pane, choose Storage > SFS Turbo. On the page displayed on the right, select a namespace and click Import.

Step 2  Select one or more SFS Turbo file systems that you want to import.

Step 3  Click Import.

Using SFS Turbo Volumes

After selecting a container image during Deployment creation (3.2 Deployment) or job creation (Creating a Job), choose Storage > SFS Turbo Volumes and click Add SFS Turbo Volume in the Advanced Settings area and mount the SFS Turbo volume to the container.
Adding an SFS Turbo volume

**Figure 5-6 Adding an SFS Turbo volume**

**NOTE**
Currently, only existing SFS Turbo volumes can be used.

**Unbinding SFS Turbo Volumes**

If an imported SFS Turbo volume is no longer required, you can unbind it from the SFS Turbo file system. After being unbound, the SFS Turbo file system cannot be used for your workloads.

**NOTE**
If an SFS Turbo volume has been mounted to a workload, it cannot be unbound from the SFS Turbo file system.

**Step 1** Log in to the CCI console. In the navigation pane, choose Storage > SFS Turbo. In the file system list, click **Unbind** in the row where the target SFS Turbo volume resides.

**Step 2** Read the message that is displayed and click **Yes**.

---End

### 5.5 OBS

The CCI currently supports the use of OBS directly in SDK mode.

OBS is an object-based mass storage service that provides customers with massive, secure, highly reliable, and low-cost data storage capabilities. For details about the OBS, see OBS.

#### Application Scenarios

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Application Scenario</th>
<th>Description</th>
</tr>
</thead>
</table>
### Storage Type | Application Scenario | Description
--- | --- | ---
Native cloud application data | The OBS provides high-performance, high-availability storage services so that mobile and Internet applications can use storage services featuring high scalability, low costs, and high efficiency. Applications can use any amount of data and access content from anywhere by using the OBS.

Infrequent access storage | Static website hosting | Website operators or personal website publishers use OBS as a static website hosting resource pool. After static web pages are uploaded to the OBS, the entire static website is hosted by configuring website functions, and data is quickly distributed by using the CDN.

Backup/Active archive | The OBS provides a highly persistent, highly scalable and secure solution to back up and archive critical data for users. Users can use OBS's version control feature to provide further protection for stored data. The highly persistent and secure infrastructure designed to provide a robust disaster recovery solution for advanced data protection supports third-party backup and archiving software such as CommVault and NBU.

Other service scenarios | Services such as the image service IMS, EVS backup service VBS, and RDS database service use OBS as their storage resource pools or backup storage.

Archival storage | Deep/Cold archive | Archival storage is ideal for data that users need to back up for long periods of time (such as months, or even years) and are not sensitive to latency. Archive products can be considered if users can tolerate less than 12 hours of time delay when extracting data.

### Notes
Currently only access to buckets in North China is supported. Data can be accessed by using the domain names obs.myhwclouds.com and obs.cn-north-1.myhwclouds.com.

### Using the OBS

**Step 1**  Use OBS in the application by using the SDK. For details about the downloading and usage of the OBS SDK, visit [https://developer.huaweicloud.com/sdk?OBS](https://developer.huaweicloud.com/sdk?OBS).

**Step 2**  Package the application into a Docker image and upload it to the software repository for container. For details, see the [Software Repository for Container User Guide](https://developer.huaweicloud.com/repo).

**Step 3**  Use the image. For details, see Section 3.2 Deployment.

----End
6 Configuration Management

6.1 Using a ConfigMap

ConfigMaps are a type of resource used to store the configurations required by applications. After a ConfigMap is created, it can be used as a file in a containerized application.

Creating a ConfigMap

**Step 1** Log in to the CCI console. In the navigation pane, choose Configuration Center > ConfigMaps. On the page displayed on the right, select a namespace and click Create ConfigMap.

**Step 2** CCI allows you to create a ConfigMap by manually specifying parameters or uploading a file.

- Method 1: manually specifying parameters. Configure parameters by referring to Table 6-1. Parameters marked with an asterisk (*) are mandatory. After parameters are configured, click Create.

<table>
<thead>
<tr>
<th>Table 6-1 Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>Basic information</td>
</tr>
<tr>
<td>*Name</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Data</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Label</td>
</tr>
</tbody>
</table>

- Method 2: uploading a file.

**NOTE**

The file must be in JSON or YAML format, and the file size must be less than 1 MB. For details, see ConfigMap File Format.

Click Add File, select the created ConfigMap resource file, and click Open.

**Step 3** After the configuration is complete, click Create.

---End

**Using a ConfigMap**

After a ConfigMap is created, mount it to the specified directory of the container during workload creation. As shown in the following figure, mount ConfigMap cci-configmap01 to the /tmp/configmap1 directory.

**Figure 6-1 Using a ConfigMap**

<table>
<thead>
<tr>
<th>ConfigMap</th>
<th>Container Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>cci-configmap01</td>
<td>/tmp/configmap1</td>
</tr>
</tbody>
</table>

After the workload is created, a ConfigMap file will be created under /tmp/configmap1. The key of the ConfigMap indicates the file name, and the value indicates the file content.

**ConfigMap File Format**

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

An example of the `configmap.json` file is as follows:

```json
{
    "kind": "ConfigMap",
    "apiVersion": "v1",
    "metadata": {
        "name": "test-configmap",
        "labels": {
            "label-01": "value-01",
            "label-02": "value-02"
        },
        "annotations": {
            "description": "a test configmap"
        },
        "enable": true
    },
    "data": {
        "key-01": "value-01",
        "key-02": "value-02"
    }
}
```

**YAML format**

An example of the `configmap.yaml` file is as follows:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
labels:
  label-01: value-01
  label-02: value-02
annotations:
  description: "a test configmap"
  enable: true
data:
  key-01: value-01
  key-02: value-02
```

Creating a ConfigMap Through `kubectl`

For details, see `ConfigMap`.

### 6.2 Using a Secret

Secrets are Kubernetes objects for storing sensitive data such as passwords, tokens, certificates, and private keys. A secret can be loaded to a container as an environment variable or a file when the container is started.

**Note**

Secrets and SSL certificates share the same quota.

### Creating a Secret

**Step 1** Log in to the CCI console. In the navigation pane, choose `Configuration Center > Secrets`. On the page displayed on the right, select a namespace and click `Create Secret`.

**Step 2** CCI allows you to create a secret by manually specifying parameters or uploading a file.

- **Method 1**: manually specifying parameters. Configure parameters by referring to Table 6-2. Parameters marked with an asterisk (*) are mandatory. After parameters are configured, click `Create`.
**Table 6-2 Basic information**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic information</strong></td>
<td></td>
</tr>
<tr>
<td>*Name</td>
<td>Name of the secret. Enter 1 to 253 characters starting and ending with a letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Do not enter two consecutive periods or a period adjacent to a hyphen.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the secret.</td>
</tr>
</tbody>
</table>
| *Data | Secret data can be used in the container. **Key** indicates the file name and **Value** indicates the file content.  
1. Click **Add Data**.  
2. Enter a key and a value. If you select **Auto transcoding**, the value you entered will be automatically encoded using Base64. |
| Label | Labels are attached to various objects (such as applications, nodes, and services) in the form of key-value pairs. Labels define the identifiable properties of these objects and are used to manage and select them.  
1. Click **Add Label**.  
2. Enter the key and the value. |

- Method 2: uploading a file.

**NOTE**

The file must be in JSON or YAML format, and the file size must be less than 2 MB. For details, see **Secret File Format**.

Click **Add File**, select the created secret resource file, and click **Open**.

**Step 3** After the configuration is complete, click **Create**.

The newly created secret is displayed in the secret list.

--- End

**Using a Secret**

After a secret is created, it can be referenced as an environment variable or mounted to a container path during workload creation.
Figure 6-2 Referencing a secret as an environment variable

Figure 6-3 Mounting a secret to a container path

Secret File Format

- **secret.yaml** resource description file
  
  For example, to obtain the following key-value pairs and encrypt them for an application, you can use the secret.
  
  key1: value1
  key2: value2
  
  The defined secret file **secret.yaml** is as follows. Base64 encoding is required for the value. For details about the Base64 encoding method, see **Base64 Encoding**.

  ```yaml
  apiVersion: v1
  kind: Secret
  metadata:
    name: mysecret          #Secret name
  annotations:             #Secret name
    description: "test"
  labels:                  #Secret name
    label-01: value-01
    label-02: value-02
  data:
    key1: dmFsdWUx        #Base64 encoding required
    key2: dmFsdWUy        #Base64 encoding required
  type: Opaque            #Must be Opaque
  
- **secret.json** resource description file
  
  The defined secret file **secret.json** is as follows:

  ```json
  {
    "apiVersion": "v1",
  ```
Base64 Encoding

To perform Base64 encoding on a character string, run the `echo -n Content to be encoded | base64` command. The following is an example:

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

Creating a Secret Through kubectl

For details, see Secret.

6.3 SSL Certificate

SSL is a security protocol designed to provide security and data integrity protection for Internet communications.

CCI allows you to upload SSL certificates. When HTTPS is used for access, CCI automatically installs the SSL certificate to the layer-7 load balancer to implement data transmission encryption.

**NOTE**

Secrets and SSL certificates share the same quota.

SSL Certificate

An SSL certificate indicates compliance with the SSL protocol. An SSL certificate is issued to a server by a trusted digital certificate authority (CA) after the CA has verified the identity of the server. SSL certificates have the functions of server authentication and data transmission encryption. By installing an SSL certificate, a server can encrypt the data transmitted between clients and the server, preventing information leak. In addition, the SSL certificate verifies whether the websites visited by the server are authentic and reliable.

SSL certificates are divided into authoritative certificates and self-signed certificates. Authoritative certificates are issued by CAs. You can purchase authoritative certificates from third-party certificate agents. A client trusts websites that use authoritative certificates by default. Self-signed certificates are issued by users themselves, usually using OpenSSL. Self-signed certificates are untrusted by the client by default. The browser will display an alarm message when you access a website that uses a self-signed certificate, but you can continue the access by ignoring the alarm.
Application Scenarios

By installing an SSL certificate, a server can encrypt the data transmitted between clients and the server, preventing information leak. To enable secure public network access for a web application in CCI, set the workload access mode to Internet access and the ELB protocol to HTTPS, and then select the certificate for Internet access during workload creation.

Adding a Certificate

**Step 1** Log in to the CCI console. In the navigation pane, choose Configuration Center > SSL Certificates. On the page displayed on the right, select a namespace and click Add Certificate.

**Step 2** Specify the name and description information of the SSL certificate.

Certificate name: Enter 1 to 253 characters starting and ending with a letter or digit. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed. Do not enter two consecutive periods or a period adjacent to a hyphen.

**Step 3** Upload the certificate file and private key file.

- .crt and .cer certificate files are supported, and the file size cannot exceed 1 MB. The file content must comply with the corresponding CRT or CER protocol.
- .key and .pem private key files are supported, and the file size cannot exceed 1 MB. Private keys cannot be encrypted.

**Figure 6-4** Uploading SSL certificate files

**Certificate Data**

* Certificate File  No certificate file has been uploaded. [Please upload the file.]

Upload a file of less than 1 MB. Only files in CRT or CER format can be uploaded.

* Secret File  No secret file has been uploaded. [Please upload the file.]

Upload a file of less than 1 MB. Only files in KEY or PEM format can be uploaded.

**Step 4** Click Add.

----End

Using an SSL Certificate

When the service has public network access, you can use the SSL certificate and set the ELB protocol to the HTTPS protocol.

During workload creation, set the workload access mode to Internet access and the ELB protocol to HTTP/HTTPS, and select the SSL certificate. The SSL certificate will be automatically installed on the ELB to encrypt data for transmission.
After the workload is created, CCI will create a certificate for the load balancer and name the certificate after the workload. If a certificate with a name starting with `beethoven-cci-ingress` is created on CCI, do not delete or update it. Otherwise, an access exception may occur.

Updating and Deleting an SSL Certificate

- A certificate can be updated before it expires, and the workload using the certificate will update it synchronously.
- Do not delete a certificate that is being used by a workload. Otherwise, the workload may be inaccessible.
CCI supports the mounting of the log storage volume for log collection. To write logs to the log storage volume, you only need to add the log storage volume when you create a workload.

CCI is interconnected with Application Operations Management (AOM). AOM collects the .log files in container log storage and dumps them in AOM to facilitate viewing and retrieval.

Adding a Log Storage Volume

You can add a log storage volume for a container when creating a workload.

- Log path in the container: path for mounting the log storage to the container. The log output path of the application must be the same as this path so that logs can be written to the log storage volume.

**NOTICE**

1. After the log storage volume is mounted, the existing content in the log path will be overwritten. Ensure that the log path is an independent path; otherwise, the previous content will be invisible.

2. AOM collects only .log, .trace, and .out files in the log path.

3. In addition, AOM can collect a maximum of 20 log files. Therefore, your logs can be exported to a maximum of 20 files in the log path. Otherwise, the logs cannot be dumped to AOM.

4. AOM scans log files every minute. When a log file exceeds 50 MB, it is dumped immediately. A new .zip file is generated in the directory where the log file is located. AOM stores only the latest 20 .zip files. When the number of .zip files exceeds 20, earlier .zip files are deleted. After a log file is dumped, AOM clears the log file.

- Log storage space: Space of log storage.
After the workload is created, you can view the container logs. Click the created workload, and click **View Logs** in the row of the container instance.

You can view the logs of the corresponding container on the AOM console. For the log query method in AOM, see **Viewing Log Files**.
Except the necessary support components, other components of Kubernetes run as add-ons, such as Kubernetes DNS and Kubernetes Dashboard.

Add-ons are extension of the existing features. The current CCI provides the coredns add-on for users. You can directly install the add-on and use the functions provided by the add-on conveniently.

coredns

The coredns add-on provides internal domain name resolution services for your other workloads. You are advised not to delete or upgrade this workload; otherwise, the internal domain name resolution service may be unavailable.

Installing an Add-on

**Step 1** Log in to the CCI console. In the navigation pane, choose Add-ons. On the page displayed on the right, click Install Add-on.

**Step 2** Click Use This Add-on for the target add-on. Then, select a version from the Add-on Version drop-down list and click Submit to install the add-on.

After the installation is complete, you can see the installed add-on in the add-on list.
Follow-up Processing

After the add-on is installed and created, you can also perform the following operations on the add-on.

Table 8-1 Other operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrade</td>
<td>Click Upgrade, select the target version, specify the parameters related to the add-on, confirm the new configuration information, and click Submit.</td>
</tr>
<tr>
<td>Roll back</td>
<td>Click Rollback, select the version for rollback, and click Submit.</td>
</tr>
</tbody>
</table>
| Delete    | Click Delete, and then click Confirm.  

**NOTICE**  
The deletion is irreversible. Perform the deletion operation with caution.