Cloud Container Engine Autopilot

Quick Start

Issue 01

Date 2025-01-03

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

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

Contents

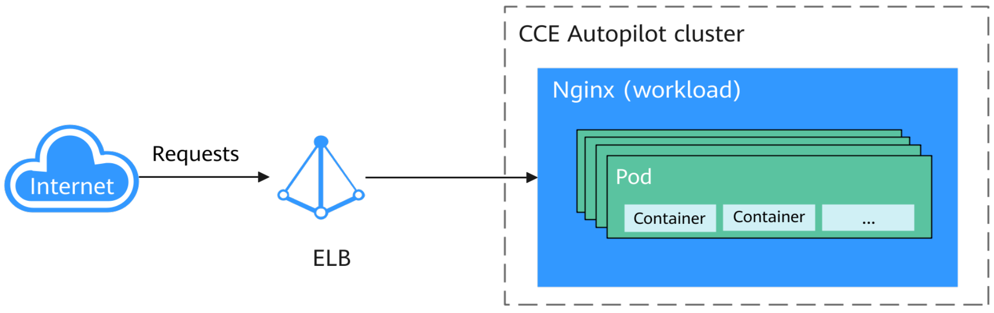
1 Deploying an Nginx Workload in a CCE Autopilot Cluster

CCE Autopilot clusters are serverless. They support intelligent version upgrade, automatic vulnerability fixing, and intelligent parameter tuning, allowing you to have a more intelligent experience. In addition, CCE Autopilot uses the same underlying resource pool. You do not need to manage and maintain the allocation and expansion of underlying resources, effectively reducing O&M costs. The underlying resource pool supports quick fault isolation and rectification. This ensures that applications can run continuously and stably and improves application reliability.

Nginx is a high-performance, open-source HTTP web server and a reverse proxy for handling concurrent connections, balancing traffic, and caching static content. An Nginx workload in a cluster can serve as a load balancer and reverse proxy to effectively distribute traffic and ensure high availability and fault tolerance of applications. It also simplifies traffic management, security control, and API gateway functions in the microservice architecture, improving system flexibility and scalability.

The following describes how to create a CCE Autopilot cluster and deploy an Nginx workload in the cluster. **Figure 1-1** shows the overall architecture.

Figure 1-1 Solution architecture



Procedure

Table 1-1 Nginx deployment process

Step	Description	Billing
Preparations	Sign up for a HUAWEI ID and make sure you have a valid payment method configured.	Billing is not involved.
Step 1: Enable CCE for the First Time and Perform Authorization	Obtain the required permissions for your account when you use CCE in the current region for the first time.	Billing is not involved.
Step 2: Create a CCE Autopilot Cluster	Create a CCE Autopilot cluster so that you can use Kubernetes in a more simple way.	Cluster management and VPC endpoints are billed. For details, see **Billing**.
Step 3: Deploy Nginx and Access It	Create an Nginx workload in the cluster and create a LoadBalancer Service so that the workload can be accessed from the public network.	Pods and load balancers are billed. For details, see **Cluster Billing** and **Billed Items (Dedicated Load Balancers)**.
Follow-up Operations: Releasing Resources	To avoid additional expenditures, release resources promptly if you no longer need them.	Billing is not involved.

Preparations

- Before you start, sign up for a HUAWEI ID and complete real-name authentication. For details, see **Signing Up for a HUAWEI ID and Enabling Huawei Cloud Services** and **Getting Authenticated**.

Step 1: Enable CCE for the First Time and Perform Authorization

CCE works closely with multiple cloud services to support computing, storage, networking, and monitoring functions. When you log in to the CCE console for the first time, CCE automatically requests permissions to access those cloud services in the region where you run your applications. If you have been authorized in the current region, skip this step.

Step 1 Log in to the **CCE console** using your HUAWEI ID.

Step 2 Click 📍 in the upper left corner of the console and select a region, for example, CN East-Shanghai1.

Step 3 Wait for the **Authorization Statement** dialog box to appear, carefully read the statement, and click **OK**.

After you agree to delegate the permissions, CCE creates an agency named **cce_admin_trust** in IAM to perform operations on other cloud resources and grants it the **Tenant Administrator** permissions. **Tenant Administrator** has the permissions on all cloud services except IAM. The permissions are used to call the cloud services on which CCE depends. The delegation takes effect only in the current region.

NOTICE

CCE depends on other cloud services. If you do not have the **Tenant Administrator** permissions, CCE may be unavailable due to insufficient permissions. For this reason, do not delete or modify the **cce_admin_trust** agency when using CCE.

----End

Step 2: Create a CCE Autopilot Cluster

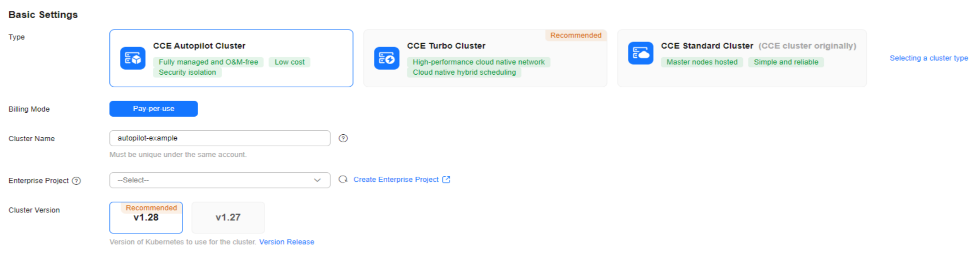
Create a CCE Autopilot cluster so that you can use Kubernetes in a more simple way. In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Buying a CCE Autopilot Cluster**.

Step 1 Log in to the **CCE console**.

- If there is no cluster in your account in the current region, click **Buy Cluster** or **Buy CCE Autopilot Cluster**.
- If there is already a cluster in your account in the current region, choose **Clusters** in the navigation pane and then click **Buy Cluster**.

Step 2 Configure basic cluster information. For details about the parameters, see **Figure 1-2** and **Table 1-2**.

Figure 1-2 Basic cluster information



Basic Settings

Type: CCE Autopilot Cluster (Fully managed and O&M-free, Low cost, Security isolation) | CCE Turbo Cluster (Recommended; High-performance cloud native network, Cloud native hybrid scheduling) | CCE Standard Cluster (CCE cluster originally) (Master nodes hosted, Simple and reliable)

Selecting a cluster type

Billing Mode: Pay-per-use

Cluster Name: autopilot-example

Must be unique under the same account.

Enterprise Project: --Select--

Create Enterprise Project

Cluster Version: v1.28 (Recommended) | v1.27

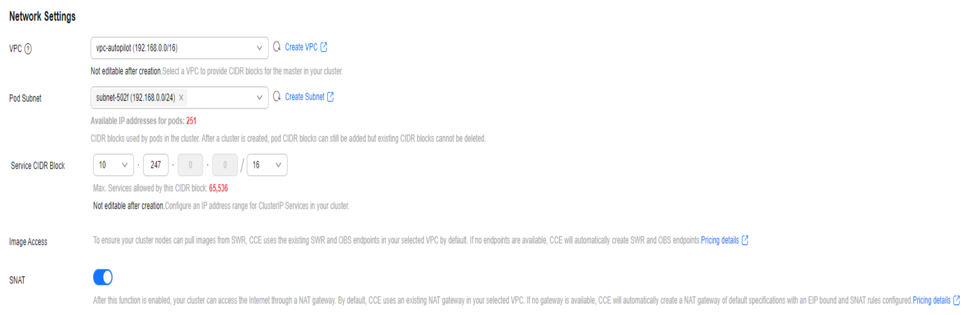
Version of Kubernetes to use for the cluster. Version Release

Table 1-2 Basic cluster information

Parameter	Example Value	Description
Type	CCE Autopilot Cluster	CCE allows you to create various types of clusters for diverse needs. ● CCE standard clusters provide highly reliable and secure containers for commercial use. ● CCE Turbo clusters use high-performance cloud native networks and provide cloud native hybrid scheduling. Such clusters have improved resource utilization and can be used in more scenarios. ● CCE Autopilot clusters are serverless, and you do not need to bother with server O&M. This greatly reduces O&M costs and improves application reliability and scalability. For more information about cluster types, see **Cluster Comparison**.
Cluster Name	autopilot-example	Enter a name for the cluster. Enter 4 to 128 characters. Start with a lowercase letter and do not end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
Enterprise Project	default	This parameter is displayed only for enterprise users who have enabled Enterprise Project. Enterprise projects are used for cross-region resource management and make it easy to centrally manage resources by department or project team. For more information, see **Project Management**. Select an enterprise project as needed. If there is no special requirement, you can select **default**.
Cluster Version	v1.28	Select the Kubernetes version for the cluster. You are advised to select the latest version.

Step 3 Configure cluster network information. For details about the parameters, see **Figure 1-3** and **Table 1-3**.

Figure 1-3 Cluster network information



Network Settings

VPC: vpc-autopilot (192.168.0.0/16) Create VPC
Not editable after creation.Select a VPC to provide CIDR blocks for the master in your cluster.

Pod Subnet: subnet-502f (192.168.0.0/24) Create Subnet
Available IP addresses for pods: 251
CIDR blocks used by pods in the cluster. After a cluster is created, pod CIDR blocks can still be added but existing CIDR blocks cannot be deleted.

Service CIDR Block: 10 · 247 · 0 · 0 / 16
Max. Services allowed by this CIDR block: 65,536
Not editable after creation.Configure an IP address range for ClusterIP Services in your cluster.

Image Access: To ensure your cluster nodes can pull images from SWR, CCE uses the existing SWR and OBS endpoints in your selected VPC by default. If no endpoints are available, CCE will automatically create SWR and OBS endpoints.Pricing details

SNAT: enabled
After this function is enabled, your cluster can access the Internet through a NAT gateway. By default, CCE uses an existing NAT gateway in your selected VPC. If no gateway is available, CCE will automatically create a NAT gateway of default specifications with an EIP bound and SNAT rules configured.Pricing details

Table 1-3 Cluster network information

Parameter	Example Value	Description
VPC	vpc-autopilot	Select a VPC where the cluster will be running. If no VPC is available, click **Create VPC** on the right to create one. For details, see **Creating a VPC and Subnet**. The VPC cannot be changed after the cluster is created.
Pod Subnet	subnet-502f	Select the subnet where the pods will be running. Each pod requires a unique IP address. The number of IP addresses in a subnet determines the maximum number of pods in a cluster and the maximum number of containers. After the cluster is created, you can add subnets. If no subnet is available, click **Create Subnet** on the right to create one. For details, see **Creating a VPC and Subnet**.
Service CIDR Block	10.247.0.0/16	Select a Service CIDR block, which will be used by containers in the cluster to access each other. This CIDR block determines the maximum number of Services. After the cluster is created, the Service CIDR block cannot be changed.
SNAT	Enabled	This option is enabled by default, and the cluster can access the Internet through a NAT gateway. By default, an existing NAT gateway in the selected VPC is used. If there are no NAT gateways, CCE Autopilot automatically creates a NAT gateway with default specifications, binds an EIP to the NAT gateway, and configures SNAT rules. The NAT gateway will be billed. For details, see **NAT Gateway Billing**.

Step 4 Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation.

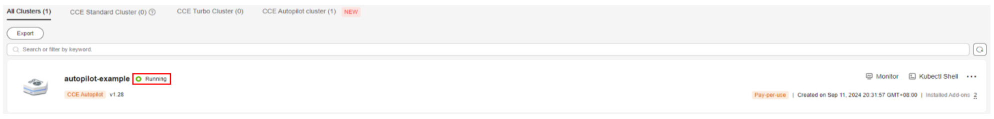
In this example, only the default add-ons, CoreDNS and Kubernetes Metrics Server, are installed.

Step 5 Click **Next: Add-on Configuration** to configure the selected add-ons. Mandatory add-ons cannot be configured.

Step 6 Click **Next: Confirm configuration**, check the displayed cluster resource list, and click **Submit**.

It takes about 5 to 10 minutes to create a cluster. After the cluster is created, the cluster is in the **Running** state.

Figure 1-4 A running cluster



autopilot-example Running
CCE Autopilot v1.28
Pay-per-use | Created on Sep 11, 2024 20:31:57 GMT+08:00 | Installed Add-ons 2

----End

Step 3: Deploy Nginx and Access It

You can create an Nginx workload in a cluster and deploy it in containers for high resource utilization and automatic management. You also need to create a LoadBalancer Service for the workload so that the workload can be accessed from the public network. You can use either of the following methods to deploy and access the Nginx workload.

Using the CCE Console

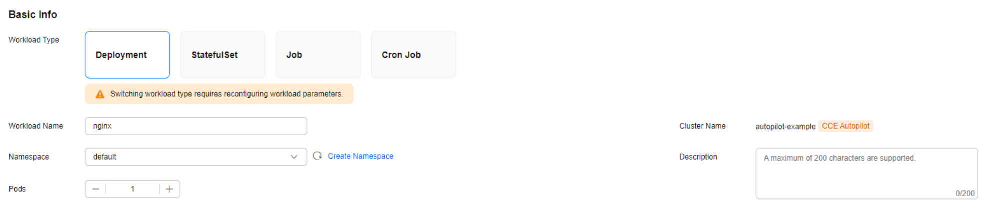
Step 1 Log in to the CCE console and click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**.

Step 3 Configure basic workload information. For details about the parameters, see **Figure 1-5** and **Table 1-4**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Creating a Workload**. You can select a reference document based on the workload type.

Figure 1-5 Basic workload information



Workload Type: Deployment
Switching workload type requires reconfiguring workload parameters.
Workload Name: nginx
Namespace: default
Pods: 1
Cluster Name: autopilot-example CCE Autopilot
Description: A maximum of 200 characters are supported. 0/200

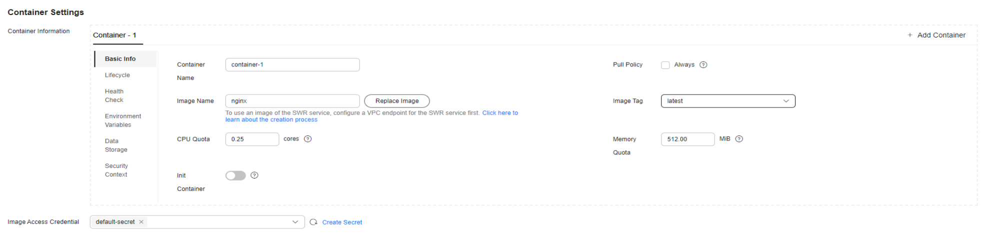
Table 1-4 Basic workload information

Parameter	Example Value	Description
Workload Type	Deployment	Select a workload type. A workload defines the creation, status, and lifecycle of pods. By creating a workload, you can manage and control the behavior of multiple pods, such as scaling, update, and restoration. • **Deployment**: runs a stateless application. It supports online deployment, rolling upgrade, replica creation, and restoration. • **StatefulSet**: runs a stateful application. Each pod for running the application has an independent state, and the data can be restored when the pod is restarted or migrated, ensuring application reliability and consistency. • **Job**: a one-off job. After the job is complete, the pods are automatically deleted. • **Cron Job**: a time-based job runs a specified job in a specified period. For more information about workloads, see **Workload Overview**. In this example, Nginx is deployed as a Deployment because Nginx is mainly used to forward requests, balance loads, and distribute static content. No persistent data needs to be stored locally.
Workload Name	nginx	Enter a name for the workload. Enter 1 to 63 characters, starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
Namespace	default	Select a namespace. A namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces. After a cluster is created, the **default** namespace is created by default. If there is no special requirement, select **default**.
Pods	1	Enter the number of pods. The policy for setting the number of pods is as follows: • High availability: To ensure high availability of the workload, at least 2 pods are required to prevent individual faults. • High performance: Set the number of pods based on the workload traffic and resource requirements to avoid overload or resource waste. In this example, set the number of pods set to **1**.

Step 4 Configure container information. For details about the parameters, see **Figure 1-6** and **Table 1-5**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Creating a Workload**. You can select a reference document based on the workload type.

Figure 1-6 Container settings



Container Name: container-1
Pull Policy: Always
Image Name: nginx
Image Tag: latest
To use an image of the SWR service, configure a VPC endpoint for the SWR service first. Click here to learn about the creation process
CPU Quota: 0.25 cores
Memory Quota: 512.00 MiB
Init Container
Image Access Credential: default-secret

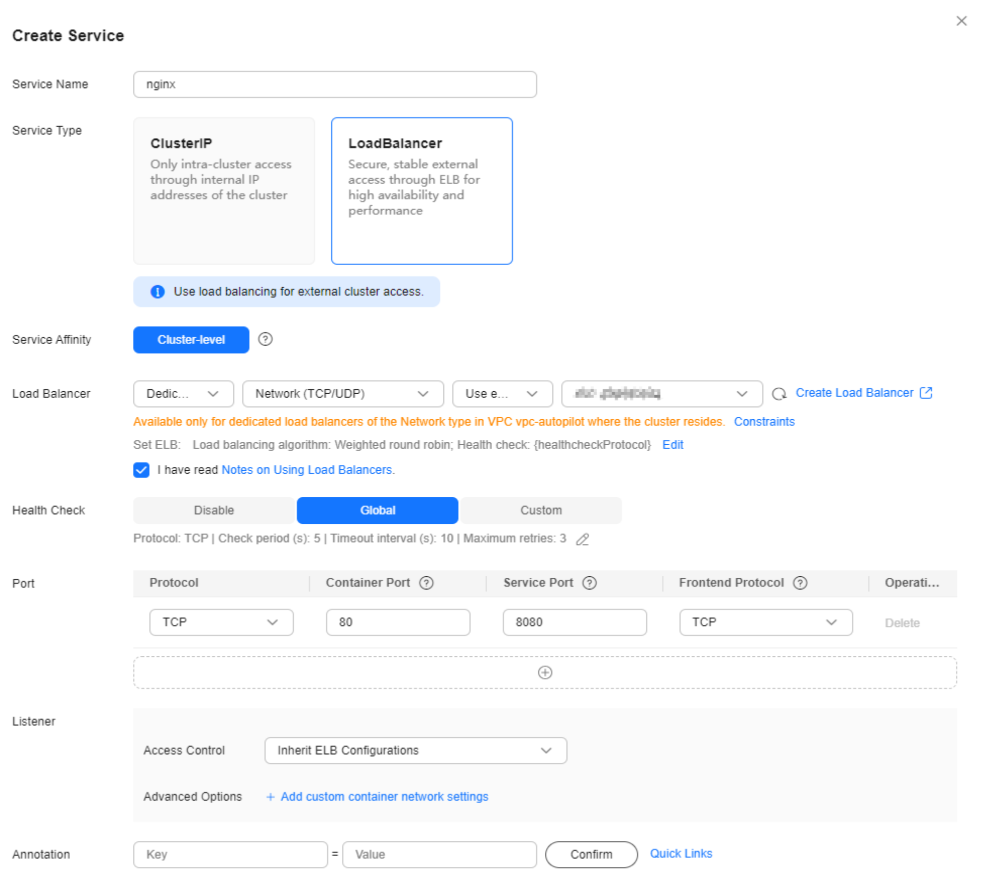
Table 1-5 Container settings

Parameter	Example Value	Description
Image Name	nginx	Click **Select Image**. In the displayed dialog box, click the **Open Source Images** tab and select a public image.
Image Tag	latest	Select the required image tag.
CPU Quota	0.25 cores	Specify the CPU limit, which defines the maximum number of CPU cores that can be used by a container. The default value is 0.25 cores.
Memory Quota	512 MiB	Specify the memory limit, which is the maximum memory available for a container. The default value is 512 MiB. If the memory exceeds 512 MiB, the container will be terminated.

Step 5 Click + under **Service Settings**. On the page that is displayed, configure the Service. For details about the parameters, see **Figure 1-7** and **Table 1-6**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Service** . Select a reference document based on the Service type.

Figure 1-7 Service settings



Create Service

Service Name: nginx

Service Type:
- ClusterIP: Only intra-cluster access through internal IP addresses of the cluster
- LoadBalancer: Secure, stable external access through ELB for high availability and performance

Use load balancing for external cluster access.

Service Affinity: Cluster-level

Load Balancer: Dedic... | Network (TCP/UDP) | Use e... | [redacted] | Create Load Balancer

Available only for dedicated load balancers of the Network type in VPC vpc-autopilot where the cluster resides. Constraints

Set ELB: Load balancing algorithm: Weighted round robin; Health check: {healthcheckProtocol} Edit

☑ I have read Notes on Using Load Balancers.

Health Check: Disable [✓ Global] Custom

Protocol: TCP | Check period (s): 5 | Timeout interval (s): 10 | Maximum retries: 3

Port:

Protocol	Container Port	Service Port	Frontend Protocol	Operati...
TCP	80	8080	TCP	Delete

Listener:

Access Control: Inherit ELB Configurations

Advanced Options: + Add custom container network settings

Annotation: Key = Value Confirm Quick Links

Table 1-6 Service settings

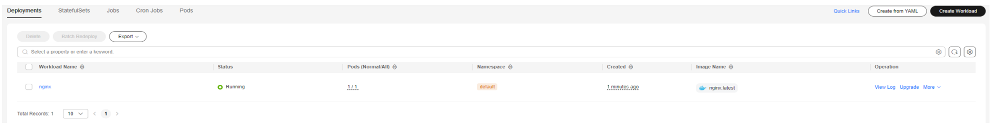
Paramete r	Example Value	Description
Service Name	nginx	Enter a name for the Service. Enter 1 to 63 characters, starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
Service Type	LoadBalanc er	Select a Service type, which determines how the workload is accessed. ● **ClusterIP**: The workload can only be accessed using IP addresses in the cluster. ● **LoadBalancer**: The workload can be accessed from the public network through a load balancer. In this example, external access to Nginx is required. So you need to create a Service of the LoadBalancer type. For more information about the Service types, see **Service**.

Parameter	Example Value	Description
Load Balancer	• Dedicated • Network load balancing (TCP/UDP) • Use existing • **elb-nginx**	• Select **Use existing** if there is a load balancer available. **NOTE** The load balancer must meet the following requirements: – It is in the same VPC as the cluster. – It is a dedicated load balancer. – It has a private IP address bound. • If there is no available load balancer, select **Auto create** to create one with an EIP bound. For details, see **Creating a LoadBalancer Service**.
Port	Protocol: TCP	**Protocol**: Select a protocol for the load balancer listener.
	Container Port: 80	**Container Port**: Enter the port on which the application listens. This port must be the same as the listening port provided by the application for external systems. When the **nginx** image is used, the container port must be set to 80 because Nginx uses port 80 to provide HTTP services by default.
	Service Port: 8080	**Service Port**: Enter a custom port. The load balancer will use this port as the listening port to provide an entry for external traffic.

Step 6 Click **Create Workload**.

After the Deployment is created, it is in the **Running** state in the Deployment list.

Figure 1-8 A running workload

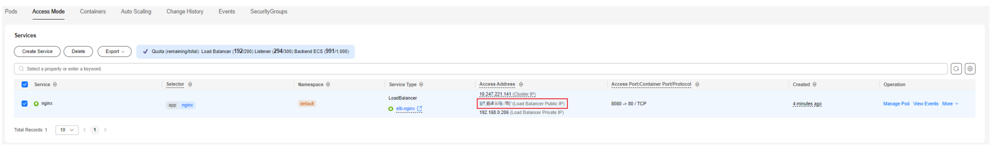


Workload Name	Status	Pods (Normal/All)	Namespace	Created	Image Name	Operation
nginx	Running	1 / 1	default	1 minutes ago	nginx:latest	View Log Upgrade More

Total Records: 1 10 1

Step 7 Click the workload name to go to the workload details page and obtain the external access address of Nginx. On the **Access Mode** tab, view the access address in the format of *{EIP bound to the load balancer}*:*{Access port}*. *{EIP bound to the load balancer}* indicates the public IP address of the load balancer, and *{Access port}* is the **Service Port** in **Step 5**.

Figure 1-9 Access address



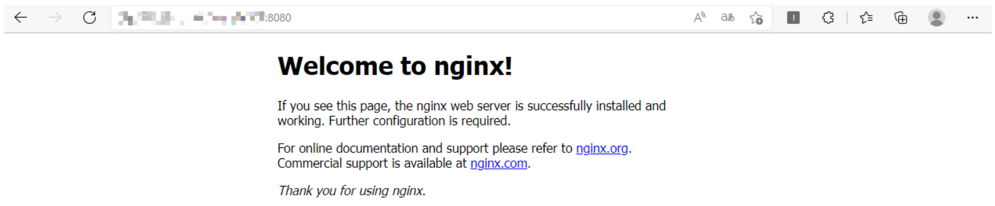
Quota (remaining/total): Load Balancer (192/200) Listener (294/300) Backend ECS (991/1,000)

Service	Selector	Namespace	Service Type	Access Address	Access Port:Container Port/Protocol	Created	Operation
nginx	app nginx	default	LoadBalancer elb-nginx	10.247.221.141 (Cluster IP) [redacted] (Load Balancer Public IP) 192.168.0.206 (Load Balancer Private IP)	8080 -> 80 / TCP	4 minutes ago	Manage Pod View Events More

Total Records: 1 10 1

Step 8 In the address box of your browser, enter *{EIP bound to the load balancer}*:*{Access port}* to access the application.

Figure 1-10 Accessing Nginx



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

----End

Using kubectl

Command line operations are required. You can perform related operations in either of the following ways:

- **Using CloudShell**: You need to ensure kubectl has been configured in CloudShell and the cluster has been connected using CloudShell. For details, see **Connecting to a Cluster Using CloudShell**.
- **Using an ECS**: You need to prepare a Linux ECS that is in the same VPC as the cluster and has an EIP bound. For details, see **Purchasing and Using a Linux ECS**. You can use an existing ECS or buy a new one. You also need to install kubectl and **connect to the cluster through kubectl**.

The following uses the first method as an example to describe how you can use kubectl to create an Nginx workload.

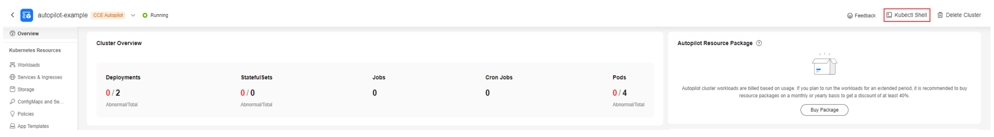
Step 1 Click the cluster name to access the cluster console.

Step 2 In the upper right corner, click **CLI tool** to access CloudShell.

NOTE

Using CloudShell to connect to a cluster is only available in some regions. For details, see the management console. If a region is not supported, using an ECS for operations.

Figure 1-11 CloudShell



Cluster Overview

Deployments 0 / 2 Abnormal/Total
StatefulSets 0 / 0 Abnormal/Total
Jobs 0
Cron Jobs 0
Pods 0 / 4 Abnormal/Total

Autopilot Resource Package

Autopilot cluster workloads are billed based on usage. If you plan to run the workloads for an extended period, it is recommended to buy resource packages on a monthly or yearly basis to get a discount of at least 40%.

Step 3 Create a YAML file for creating a workload. In this example, the file name is **nginx-deployment.yaml**. You can change it as needed.

NOTE

A Linux file name is case sensitive and can contain letters, digits, underscores (_), and hyphens (-), but cannot contain slashes (/) or null characters (\0). To improve compatibility, do not use special characters, such as spaces, question marks (?), and asterisks (*).

```
vim nginx-deployment.yaml
```

The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx     # Workload name
spec:
  replicas: 1     # Number of pods
  selector:
    matchLabels:  # Selector, which is used to select resources with specific labels. The value must be the same as that of the selector in the YAML file of the LoadBalancer Service in Step 6.
      app: nginx
  template:
    metadata:
      labels:     # Labels
        app: nginx
    spec:
      containers:
      - image: nginx:latest   # {Image name}:{Image tag}
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

Press **Esc** to exit editing mode and enter **:wq** to save the file.

Step 4 Run the following command to create the workload:

```
kubectl create -f nginx-deployment.yaml
```

If information similar to the following is displayed, the workload is being created:

```
deployment.apps/nginx created
```

Step 5 Run the following command to check the workload status:

```
kubectl get deployment
```

If the value of **READY** is **1/1**, the pod created for the workload is available. This means the workload has been created.

```
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           4m59s
```

The following table describes the parameters in the command output.

Table 1-7 Parameters in the command output

Parameter	Example Value	Description
NAME	nginx	Workload name.
READY	1/1	The number of available pods/desired pods for the workload.
UP-TO-DATE	1	The number of pods that have been updated for the workload.
AVAILABLE	1	The number of pods available for the workload.
AGE	4m59s	How long the workload has run.

Step 6 Run the following command to create the **nginx-elb-svc.yaml** file for configuring the LoadBalancer Service and associating the Service with the created workload. You can change the file name as needed.

An existing load balancer is used to create the Service. To automatically create a load balancer, see **Using kubectl to Create a Load Balancer**.

```
vim nginx-elb-svc.yaml
```

The file content is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx      # Service name
  annotations:
    kubernetes.io/elb.id: <your_elb_id>         # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance         # Load balancer type
spec:
  selector:        # The value must be the same as the value of matchLabels in the YAML file of the 
workload in Step 3.
     app: nginx
  ports:
  - name: service0
    port: 8080
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Press **Esc** to exit editing mode and enter **:wq** to save the file.

Table 1-8 Parameters for using an existing load balancer

Parameter	Example Value	Description
kubernetes.io/elb.id	405ef586-0397-45c3-bfc4-xxx	ID of an existing load balancer. **NOTE** The load balancer must meet the following requirements: ● It is in the same VPC as the cluster. ● It is a dedicated load balancer. ● It has a private IP address bound. **How to Obtain**: Log in to the **Network Console**. In the upper left corner, select the region where the cluster resides. In the navigation pane, choose **Elastic Load Balance** > **Load Balancers** and locate the load balancer. The ID is displayed below the load balancer name. Click the name of the load balancer. On the **Summary** tab, verify that the load balancer meets the preceding requirements.
kubernetes.io/elb.class	performance	Load balancer type. The value can only be **performance**, which means that only dedicated load balancers are supported.
selector	app: nginx	Selector, which is used by the Service to send traffic to pods with specific labels.
ports.port	8080	The port used by the load balancer as an entry for external traffic. You can use any port.

Parameter	Example Value	Description
ports.protocol	TCP	Protocol for the load balancer listener.
ports.targetPort	80	Port used by a Service to access the target container. This port is closely related to the applications running in the container. If the **nginx** image is used, set this port to **80**.

Step 7 Run the following command to create a Service:

```
kubectl create -f nginx-elb-svc.yaml
```

If information similar to the following is displayed, the Service has been created:

```
service/nginx created
```

Step 8 Run the following command to check the Service:

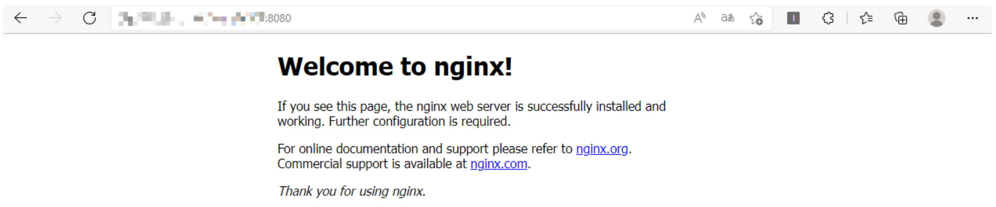
```
kubectl get svc
```

If information similar to the following is displayed, the workload access mode has been configured:

```
NAME         TYPE          CLUSTER-IP     EXTERNAL-IP               PORT(S)          AGE
kubernetes   ClusterIP     10.247.0.1     <none>                    443/TCP          18h
nginx        LoadBalancer  10.247.56.18   xx.xx.xx.xx,xx.xx.xx.xx   8080:30581/TCP   5m8s
```

Step 9 In the address box of your browser, enter *{External access address}:{Service port}* to access the workload. The external access address is the first IP address displayed for **EXTERNAL-IP**, and the Service port is 8080.

Figure 1-12 Accessing Nginx



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

----End

Follow-up Operations: Releasing Resources

To avoid additional expenditures, release resources promptly if you no longer need them.

NOTICE

- Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be recovered.
- VPC resources (such as endpoints, NAT gateways, and EIPs for SNAT) associated with a cluster are retained by default when the cluster is deleted. Before deleting a cluster, ensure that the resources are not used by other clusters.

Step 1 Log in to the **CCE console**. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster, click ··· in the upper right corner of the cluster card, and click **Delete Cluster**.

Step 3 In the displayed **Delete Cluster** dialog box, delete related resources as prompted.

Step 4 Enter **DELETE** and click **Yes** to start deleting the cluster.

It takes 1 to 3 minutes to delete a cluster. If the cluster name disappears from the cluster list, the cluster has been deleted.

----End

2 Scaling a Workload Using an HPA Policy

The traffic to applications fluctuates in peak and off-peak hours. If sufficient compute is always maintained to cope with peak traffic, the cost is high. To solve this problem, you can configure an auto scaling policy for the workload. This policy automatically adjusts the number of pods in the workload based on traffic changes or resource usage to reduce costs. You can set auto scaling policies for workloads in CCE standard, CCE Turbo, and CCE Autopilot clusters. For details about the differences, see **Table 2-1**.

This example describes how you can use HPA to adjust the scale of a workload running in a CCE Autopilot cluster. In CCE Autopilot clusters, you do not need to deploy, manage, or maintain nodes. You only need to set HPA policies to automatically adjust the number of pods on demand, simplifying resource management and O&M. In addition, CCE Autopilot prioritizes performance by setting up a serverless resource foundation in collaboration with underlying services. It uses multi-level resource pool pre-provisioning technology to meet diverse heterogeneous resource requirements and continuously improves performance through iterations.

Table 2-1 Comparison of workload auto scaling policies

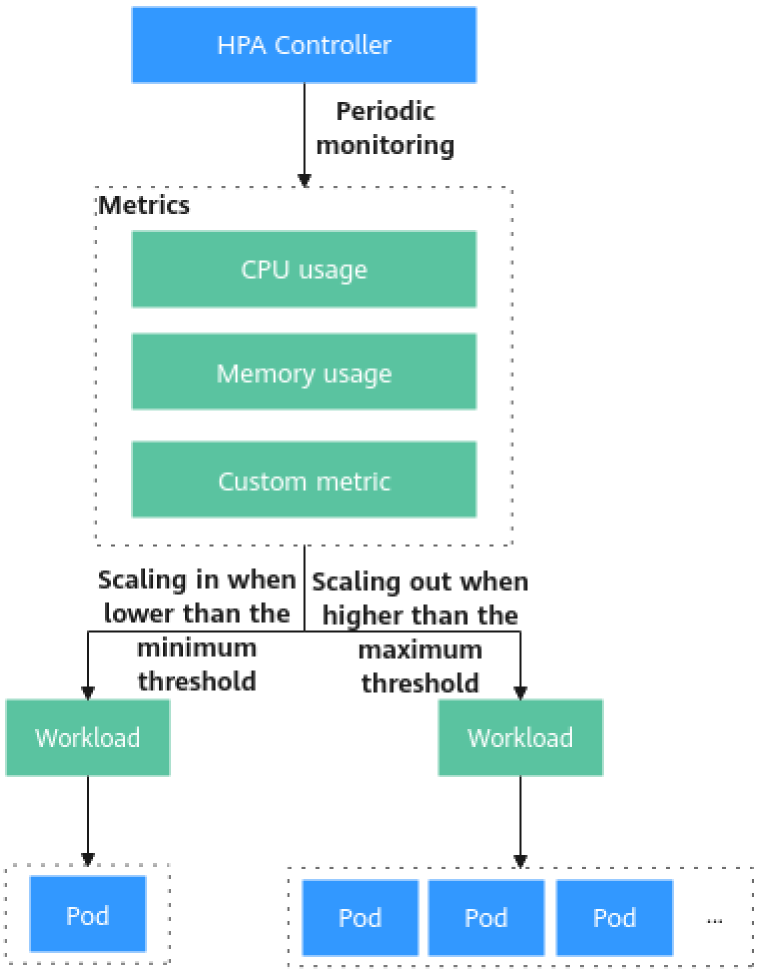
Cluster Type	Policy	Differences
CCE standard/CCE Turbo clusters	You need to configure a Cluster AutoScaling (CA) policy for nodes and Horizontal Pod Autoscaling (HPA) policy for the workload. • CA policy: scales nodes to prevent workload creation failures caused by insufficient resources. • HPA policy: adjusts the number of pods in a workload.	• Both CA and HPA policies are required. • Auto scaling can be implemented in minutes.

Cluster Type	Policy	Differences
CCE Autopilot clusters	You only need to create an HPA policy for workload scaling.	● Only an HPA policy is required. ● Auto scaling can be implemented in seconds.

In this example, you only need to configure an HPA policy. The following describes how this HPA policy works:

Like in **Figure 2-1**, the HPA policy periodically monitors specified metrics (such as CPU usage, memory usage, and other custom metrics) to dynamically adjust the number of pods in the workload. Take the CPU usage as an example. When the CPU usage is higher than the maximum threshold, the HPA policy increases the number of pods to reduce the compute pressure of the workload. When the CPU usage is lower than the minimum threshold, the HPA policy reduces the number of pods to save costs. For more information about HPA, see **How HPA Works**.

Figure 2-1 How HPA works



Procedure

Table 2-2 Procedure for configuring auto scaling

Step	Description	Billing
Preparations	Sign up for a HUAWEI ID and make sure you have a valid payment method configured.	Billing is not involved.
Step 1: Enable CCE for the First Time and Perform Authorization	Obtain the required permissions for your account when you use CCE in the current region for the first time.	Billing is not involved.
Step 2: Create a CCE Autopilot Cluster	Create a CCE Autopilot cluster so that you can use Kubernetes in a more simple way.	Cluster management and VPC endpoints are billed. For details, see **Billing**.
Step 3: Create a Compute-intensive Application and Upload Its Image to SWR	Create a compute-intensive application for the stress test. Create an image using the application and then push the image to SWR for deployment and management in the cluster.	An ECS is required. You will be billed for the ECS and the bound EIP. For details, see **Billing Overview**.
Step 4: Create a Workload Using the hpa-example Image	Use the image to create a workload and create a Service of the LoadBalancer type for the workload so that the workload can be accessed from the public network.	Pods and load balancers are billed. For details, see **Cluster Billing** and **Billed Items (Dedicated Load Balancers)**.
Step 5: Create an HPA Policy	Create an HPA policy for the workload to control the number of pods required by the workload.	Billing is not involved.
Step 6: Verify Auto Scaling	Verify that the HPA policy triggers auto scaling for the workload.	During auto scaling, the number of pods increases or decreases and the pod price changes.For details, see **Billing**.
Follow-up Operations: Releasing Resources	To avoid additional expenditures, release resources promptly if you no longer need them.	Billing is not involved.

Preparations

- Before you start, sign up for a HUAWEI ID and complete real-name authentication. For details, see **Signing Up for a HUAWEI ID and Enabling Huawei Cloud Services** and **Getting Authenticated**.

Step 1: Enable CCE for the First Time and Perform Authorization

CCE works closely with multiple cloud services to support computing, storage, networking, and monitoring functions. When you log in to the CCE console for the first time, CCE automatically requests permissions to access those cloud services in the region where you run your applications. If you have been authorized in the current region, skip this step.

Step 1 Log in to the **CCE console** using your HUAWEI ID.

Step 2 Click in the upper left corner of the console and select a region, for example, CN East-Shanghai1.

Step 3 Wait for the **Authorization Statement** dialog box to appear, carefully read the statement, and click **OK**.

After you agree to delegate the permissions, CCE creates an agency named **cce_admin_trust** in IAM to perform operations on other cloud resources and grants it the **Tenant Administrator** permissions. **Tenant Administrator** has the permissions on all cloud services except IAM. The permissions are used to call the cloud services on which CCE depends. The delegation takes effect only in the current region.

NOTICE

CCE depends on other cloud services. If you do not have the **Tenant Administrator** permissions, CCE may be unavailable due to insufficient permissions. For this reason, do not delete or modify the **cce_admin_trust** agency when using CCE.

----End

Step 2: Create a CCE Autopilot Cluster

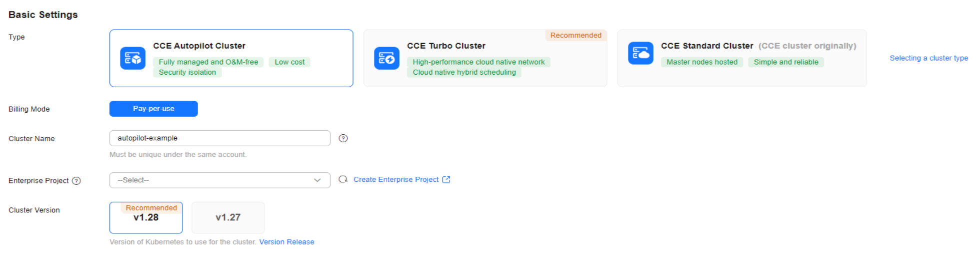
Create a CCE Autopilot cluster so that you can use Kubernetes in a more simple way. In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Buying a CCE Autopilot Cluster**.

Step 1 Log in to the **CCE console**.

- If there is no cluster in your account in the current region, click **Buy Cluster** or **Buy CCE Autopilot Cluster**.
- If there is already a cluster in your account in the current region, choose **Clusters** in the navigation pane and then click **Buy Cluster**.

Step 2 Configure basic cluster information. For details about the parameters, see **Figure 2-2** and **Table 2-3**.

Figure 2-2 Basic cluster information


Basic Settings

Type: CCE Autopilot Cluster (Fully managed and O&M-free, Low cost, Security isolation) | CCE Turbo Cluster (Recommended; High-performance cloud native network, Cloud native hybrid scheduling) | CCE Standard Cluster (CCE cluster originally) (Master nodes hosted, Simple and reliable) — Selecting a cluster type

Billing Mode: Pay-per-use

Cluster Name: autopilot-example
Must be unique under the same account.

Enterprise Project: --Select-- — Create Enterprise Project

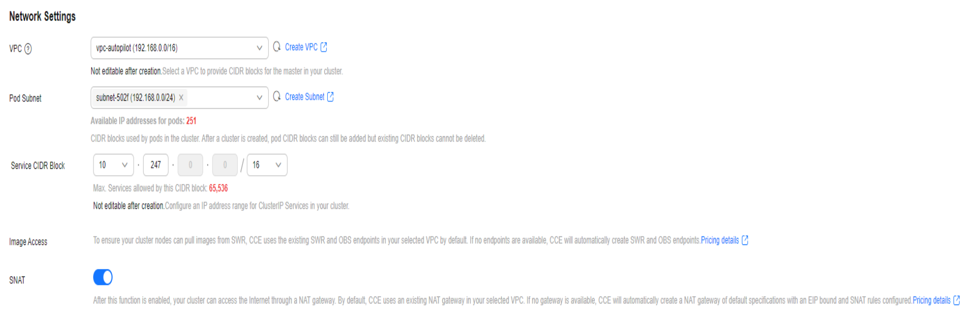
Cluster Version: Recommended v1.28 | v1.27
Version of Kubernetes to use for the cluster. Version Release

Table 2-3 Basic cluster information

Parameter	Example Value	Description
Type	CCE Autopilot Cluster	CCE allows you to create various types of clusters for diverse needs. ● CCE standard clusters provide highly reliable and secure containers for commercial use. ● CCE Turbo clusters use high-performance cloud native networks and provide cloud native hybrid scheduling. Such clusters have improved resource utilization and can be used in more scenarios. ● CCE Autopilot clusters are serverless, and you do not need to bother with server O&M. This greatly reduces O&M costs and improves application reliability and scalability. For more information about cluster types, see **Cluster Comparison**.
Cluster Name	autopilot-example	Enter a name for the cluster. Enter 4 to 128 characters. Start with a lowercase letter and do not end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
Enterprise Project	default	This parameter is displayed only for enterprise users who have enabled Enterprise Project. Enterprise projects are used for cross-region resource management and make it easy to centrally manage resources by department or project team. For more information, see **Project Management**. Select an enterprise project as needed. If there is no special requirement, you can select **default**.
Cluster Version	v1.28	Select the Kubernetes version for the cluster. You are advised to select the latest version.

Step 3 Configure cluster network information. For details about the parameters, see **Figure 2-3** and **Table 2-4**.

Figure 2-3 Cluster network information



Network Settings

VPC: vpc-autopilot (192.168.0.0/16) — Create VPC

Not editable after creation.Select a VPC to provide CIDR blocks for the master in your cluster.

Pod Subnet: subnet-502f (192.168.0.0/24) — Create Subnet

Available IP addresses for pods: 251

CIDR blocks used by pods in the cluster. After a cluster is created, pod CIDR blocks can still be added but existing CIDR blocks cannot be deleted.

Service CIDR Block: 10 . 247 . 0 . 0 / 16

Max. Services allowed by this CIDR block: 65,536

Not editable after creation.Configure an IP address range for ClusterIP Services in your cluster.

Image Access: To ensure your cluster nodes can pull images from SWR, CCE uses the existing SWR and OBS endpoints in your selected VPC by default. If no endpoints are available, CCE will automatically create SWR and OBS endpoints.Pricing details

SNAT: enabled

After this function is enabled, your cluster can access the Internet through a NAT gateway. By default, CCE uses an existing NAT gateway in your selected VPC. If no gateway is available, CCE will automatically create a NAT gateway of default specifications with an EIP bound and SNAT rules configured.Pricing details

Table 2-4 Cluster network information

Parameter	Example Value	Description
VPC	vpc-autopilot	Select a VPC where the cluster will be running. If no VPC is available, click **Create VPC** on the right to create one. For details, see **Creating a VPC and Subnet**. The VPC cannot be changed after the cluster is created.
Pod Subnet	subnet-502f	Select the subnet where the pods will be running. Each pod requires a unique IP address. The number of IP addresses in a subnet determines the maximum number of pods in a cluster and the maximum number of containers. After the cluster is created, you can add subnets. If no subnet is available, click **Create Subnet** on the right to create one. For details, see **Creating a VPC and Subnet**.
Service CIDR Block	10.247.0.0/16	Select a Service CIDR block, which will be used by containers in the cluster to access each other. This CIDR block determines the maximum number of Services. After the cluster is created, the Service CIDR block cannot be changed.
SNAT	Enabled	This option is enabled by default, and the cluster can access the Internet through a NAT gateway. By default, an existing NAT gateway in the selected VPC is used. If there are no NAT gateways, CCE Autopilot automatically creates a NAT gateway with default specifications, binds an EIP to the NAT gateway, and configures SNAT rules. The NAT gateway will be billed. For details, see **NAT Gateway Billing**.

Step 4 Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation.

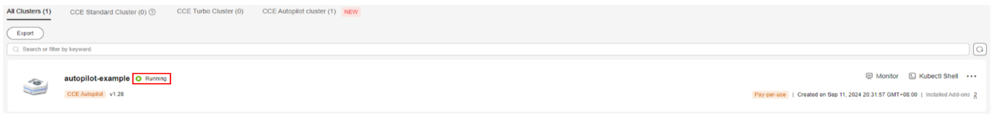
In this example, only the default add-ons, CoreDNS and Kubernetes Metrics Server, are installed.

Step 5 Click **Next: Add-on Configuration** to configure the selected add-ons. Mandatory add-ons cannot be configured.

Step 6 Click **Next: Confirm configuration**, check the displayed cluster resource list, and click **Submit**.

It takes about 5 to 10 minutes to create a cluster. After the cluster is created, the cluster is in the **Running** state.

Figure 2-4 A running cluster



autopilot-example Running
CCE Autopilot v1.28
Pay-per-use | Created on Sep 11, 2024 20:31:57 GMT+08:00 | Installed Add-ons 2

----End

Step 3: Create a Compute-intensive Application and Upload Its Image to SWR

Create a compute-intensive application for the stress test, which will be used to verify whether the HPA policy can automatically adjust the number of pods based on the metrics. Create an image using the application and then push the image to SWR for deployment and management in the cluster.

Step 1 Prepare a Linux ECS that is in the same VPC as the cluster created in **Step 2: Create a CCE Autopilot Cluster** and bind an EIP to the ECS. For details, see **Purchasing and Using Linux ECS**. You can choose to use an existing ECS or buy a new one.

In this example, CentOS 7.9 64-bit (40 GiB) is used as an example to describe how you can use PHP to create a compute-intensive application and upload its image to SWR.

Step 2 Log in to the ECS. For details, see **Logging In to a Linux ECS Using CloudShell**.

Step 3 Install Docker.

1. Run the following command to check whether Docker has been installed: If Docker has been installed, skip this step. If not, take the following steps to install, run, and check Docker.
   ```
   docker --version
   ```
   If the following information is displayed, Docker is not installed:
   ```
   -bash: docker: command not found
   ```
2. Run the following command to install and run Docker:
   ```
   yum install docker
   ```
   Enable Docker to automatically start upon system boot and run immediately.
   ```
   systemctl enable docker
   systemctl start docker
   ```
3. Run the following command to check the installation result:
   ```
   docker --version
   ```

If the following information is displayed, Docker has been installed:

```
Docker version 1.13.1, build 7d71120/1.13.1
```

Step 4 Create a compute-intensive application and use it to build an image.

1. Create a PHP file named **index.php**. You can change the file name as needed. This file means that after receiving a user request, the file performs 1,000,000 cyclic calculations for a square root (0.0001+0.01+0.1+... in this example) and then returns **"OK!"**.

   NOTE

   A Linux file name is case sensitive and can contain letters, digits, underscores (_), and hyphens (-), but cannot contain slashes (/) or null characters (\0). To improve compatibility, do not use special characters, such as spaces, question marks (?), and asterisks (*).

   ```
   vim index.php
   ```

   The file content is as follows:

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

   Press **Esc** to exit editing mode and enter **:wq** to save the file.

2. Run the following command to compile a Dockerfile to create an image:

   ```
   vim Dockerfile
   ```

   The file content is as follows:

   ```
   # Use the official PHP Docker image.
   FROM php:5-apache
   # Copy the local index.php file to the specified directory in the container. This file will be used as the default web page.
   COPY index.php /var/www/html/index.php
   # Modify the permissions of the index.php file to make it readable and executable for all users, ensuring that the file can be correctly accessed on the web server.
   RUN chmod a+rx index.php
   ```

3. Build an image named **hpa-example** and with the **latest** tag. You can change the name and image tag as needed.

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

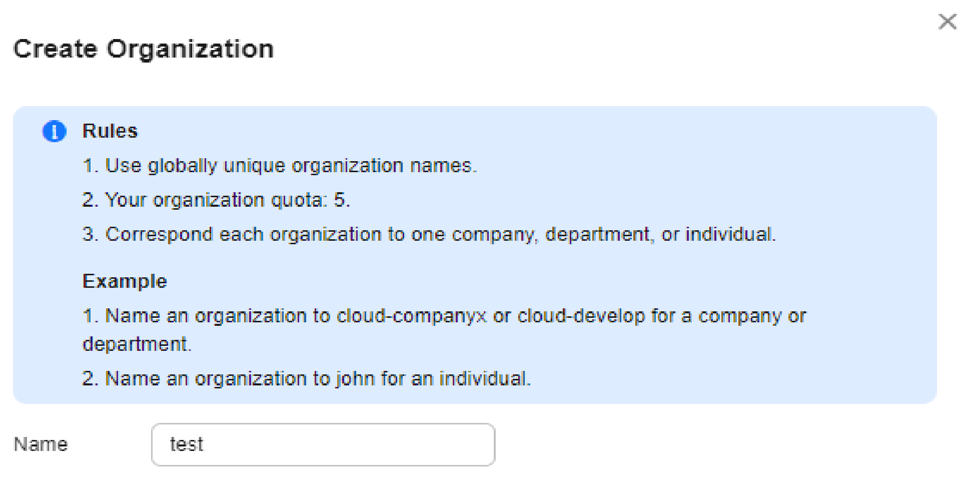
   If the following information is displayed, the image has been built.

   ```
   Sending build context to Docker daemon  158.1MB
   Step 1/3 : FROM php:5-apache
   ...
   Successfully built xxx
   Successfully tagged hpa-example:latest
   ```

Step 5 Upload the **hpa-example** image to SWR.

1. Log in to the **SWR console**. In the navigation pane, choose **Organizations**. In the upper right corner, click **Create Organization**. Set **Name** and click **OK**. If there is already an organization, skip this step.

Figure 2-5 Creating an organization


Create Organization

Rules
1. Use globally unique organization names.
2. Your organization quota: 5.
3. Correspond each organization to one company, department, or individual.

Example
1. Name an organization to cloud-companyx or cloud-develop for a company or department.
2. Name an organization to john for an individual.

Name: test

2. In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. On the page that is displayed, click **Generate Login Command** and click ⧉ to copy the temporary login command for logging in to SWR from ECS.

   Run the copied login command on the ECS.

   ```
   docker login -u cn-east-3xxx swr.cn-east-3.myhuaweicloud.com
   ```

   If information similar to the following is displayed, the login is successful:

   ```
   Login Succeeded
   ```

3. Add a tag to the **hpa-example** image. The code structure is as follows:

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

   Example:

   ```
   docker tag hpa-example:latest swr.cn-east-3.myhuaweicloud.com/test/hpa-example:latest
   ```

Table 2-5 Parameters for adding a tag to an image

Parameter	Example Value	Description
{Image name 1:Tag 1}	hpa-example:latest	Replace the values with the name and tag of the image to be uploaded.
{Image repository address}	swr.cn-east-3.myhuaweicloud.com	Replace it with the domain name at the end of the **login command**.
{Organization name}	test	Replace it with the name of the **created image organization**.
{Image name 2:Tag 2}	hpa-example:latest	Replace the values with the name and tag to be displayed in the SWR image repository. You can change them as needed.

4. Push the image to the image repository. The code structure is as follows:

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

   Example:

   ```
   docker push swr.cn-east-3.myhuaweicloud.com/test/hpa-example:latest
   ```

   If the following information is displayed, the image is successfully pushed:

   ```
   6d6b9812c8ae: Pushed
   ...
   fe4c16cbf7a4: Pushed
   latest: digest: sha256:eb7e3bbdxxx size: xxx
   ```

   On the **SWR console**, refresh the **My Images** page to view the image pushed.

----End

Step 4: Create a Workload Using the hpa-example Image

Use the **hpa-example** image to create a workload and create a Service of the LoadBalancer type for the workload so that the workload can be accessed from the public network. You can use either of the following methods to create the workload.

Using the CCE Console

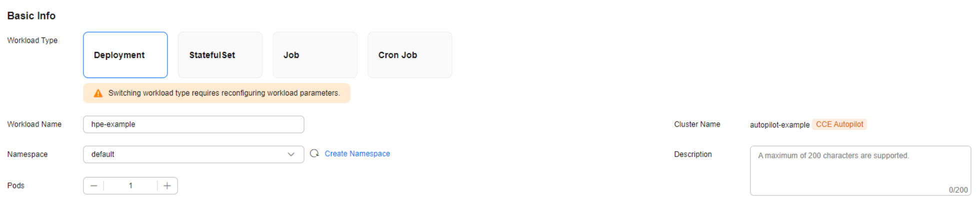
Step 1 Log in to the **CCE console**. Click the cluster name to access the cluster console.

Step 2 In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**.

Step 3 Configure basic workload information. For details about the parameters, see **Figure 2-6** and **Table 2-6**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Creating a Workload**. You can select a reference document based on the workload type.

Figure 2-6 Basic workload information



Basic Info

Workload Type	Deployment (selected), StatefulSet, Job, Cron Job
	Switching workload type requires reconfiguring workload parameters.
Workload Name	hpe-example
Namespace	default
Pods	1
Cluster Name	autopilot-example CCE Autopilot
Description	A maximum of 200 characters are supported. 0/200

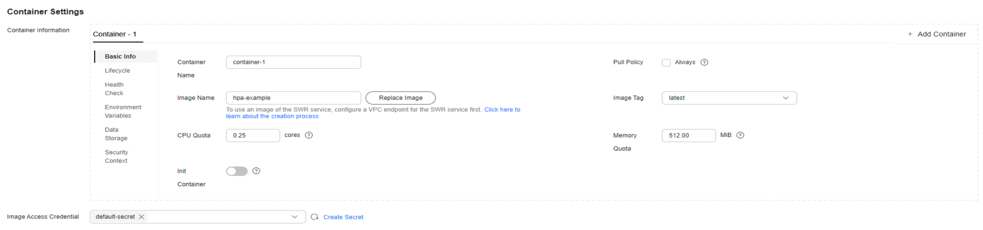
Table 2-6 Basic workload information

Parameter	Example Value	Description
Workload Type	Deployment	A workload defines the creation, status, and lifecycle of pods. By creating a workload, you can manage and control the behavior of multiple pods, such as scaling, update, and restoration. ● **Deployment**: runs a stateless application. It supports online deployment, rolling upgrade, replica creation, and restoration. ● **StatefulSet**: runs a stateful application. Each pod for running the application has an independent state, and the data can be restored when the pod is restarted or migrated, ensuring application reliability and consistency. ● **Job**: a one-off job. After the job is complete, the pods are automatically deleted. ● **Cron Job**: a time-based job runs a specified job in a specified period. For more information about workloads, see **Workload Overview**. hpa-example is mainly used for stress tests and does not need to store any persistent data locally. Therefore, hpa-example is deployed as a Deployment in this example.
Workload Name	hpa-example	Enter a name for the workload. Enter 1 to 63 characters, starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.
Namespace	default	A namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces. After a cluster is created, the **default** namespace is created by default. If there is no special requirement, select **default**.
Pods	1	Enter the number of pods. The policy for setting the number of pods is as follows: ● High availability: To ensure high availability of the workload, at least 2 pods are required to prevent individual faults. ● High performance: Set the number of pods based on the workload traffic and resource requirements to avoid overload or resource waste. In this example, set the number of pods set to **1**.

Step 4 Configure container information. For details about the parameters, see **Figure 2-7** and **Table 2-7**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about other parameters, see **Creating a Workload**. You can select a reference document based on the workload type.

Figure 2-7 Container settings



Container Name: container-1
Pull Policy: Always
Image Name: hpa-example
Image Tag: latest
To use an image of the SWR service, configure a VPC endpoint for the SWR service first. Click here to learn about the creation process
CPU Quota: 0.25 cores
Memory Quota: 512.00 MiB
Init Container
Image Access Credential: default-secret

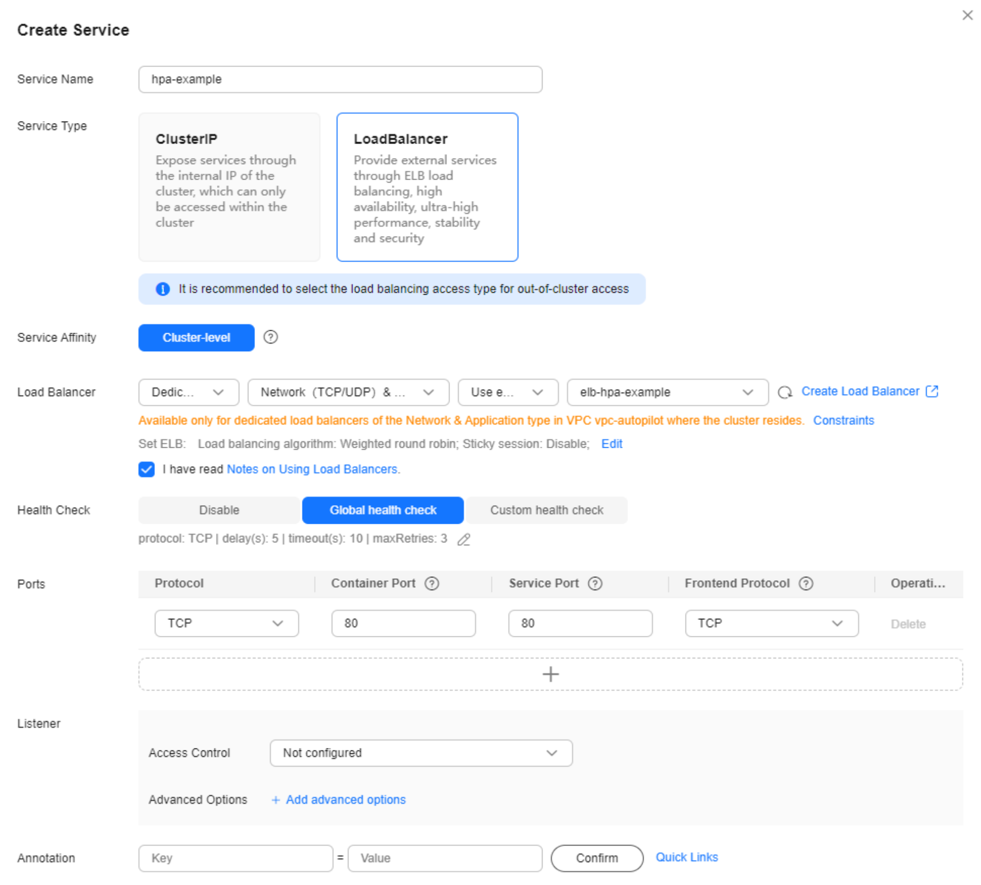
Table 2-7 Container settings

Parameter	Example Value	Description
Image Name	hpa-example	Click **Select Image**. In the displayed dialog box, switch to the **My Images** tab and select the **hpa-example** image pushed.
Image Tag	latest	Select the required image tag.
CPU Quota	0.25 cores	Specify the CPU limit, which defines the maximum number of CPU cores that can be used by a container. The default value is 0.25 cores.
Memory Quota	512 MiB	Specify the memory limit, which is the maximum memory available for a container. The default value is 512 MiB. If the memory exceeds 512 MiB, the container will be terminated.

Step 5 Click + under **Service Settings**. On the page that is displayed, configure the Service. For details about the parameters, see **Figure 2-8** and **Table 2-8**.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see **Service** . Select a reference document based on the Service type.

Figure 2-8 Service settings



Create Service

Service Name: hpa-example

Service Type: [✓ LoadBalancer] ClusterIP

ClusterIP: Expose services through the internal IP of the cluster, which can only be accessed within the cluster

LoadBalancer: Provide external services through ELB load balancing, high availability, ultra-high performance, stability and security

It is recommended to select the load balancing access type for out-of-cluster access

Service Affinity: Cluster-level

Load Balancer: Dedic... | Network (TCP/UDP) & ... | Use e... | elb-hpa-example | Create Load Balancer

Available only for dedicated load balancers of the Network & Application type in VPC vpc-autopilot where the cluster resides. Constraints

Set ELB: Load balancing algorithm: Weighted round robin; Sticky session: Disable; Edit

✓ I have read Notes on Using Load Balancers.

Health Check: Disable [✓ Global health check] Custom health check

protocol: TCP | delay(s): 5 | timeout(s): 10 | maxRetries: 3

Ports:

Protocol	Container Port	Service Port	Frontend Protocol	Operati...
TCP	80	80	TCP	Delete

Listener:

Access Control: Not configured

Advanced Options: + Add advanced options

Annotation: Key = Value | Confirm | Quick Links

Table 2-8 Service settings

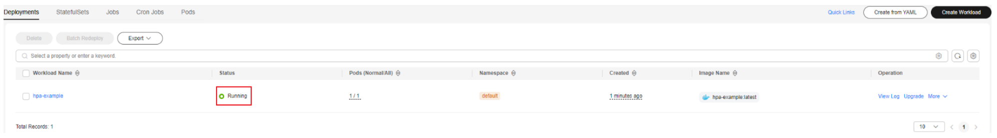
Parameter	Example Value	Description
Service Name	hpa-example	Enter a Service name. Enter 1 to 63 characters, starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.

Parameter	Example Value	Description
Service Type	LoadBalancer	Select a Service type, which determines how the workload is accessed. ● **ClusterIP**: The workload can only be accessed using IP addresses in the cluster. ● **LoadBalancer**: The workload can be accessed from the public network through a load balancer. In this example, external access to **hpa-example** is required. So you need to create a Service of the LoadBalancer type. For more information about the Service types, see **Service**.
Load Balancer	● Dedicated ● Network (TCP/UDP) & Application (HTTP/ HTTPS) ● Use existing ● **elb-hpa-example**	● Select **Use existing** if there is a load balancer available. **NOTE** The load balancer must meet the following requirements: – It is in the same VPC as the cluster. – It is a dedicated load balancer. – It has a private IP address bound. ● If there is no available load balancer, select **Auto create** to create one with an EIP bound. For details, see **Creating a LoadBalancer Service**.
Port	Protocol: TCP	**Protocol**: Select a protocol for the load balancer listener.
	Container Port: 80	**Container Port**: Enter the port on which the application listens. This port must be the same as the listening port provided by the application for external systems.
	Service Port: 80	**Service Port**: Enter a custom port. The load balancer will use this port as the listening port to provide an entry for external traffic.

Step 6 Click **Create Workload**.

After the Deployment is created, it is in the **Running** state in the Deployment list.

Figure 2-9 A running workload

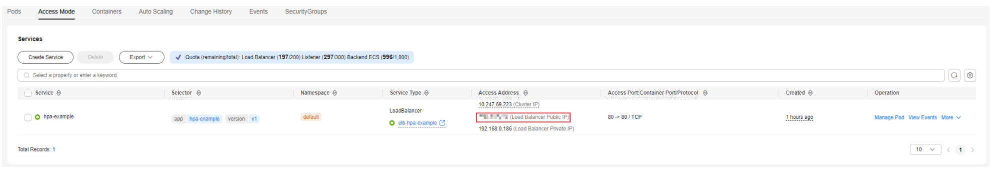


Workload Name	Status	Pods (Normal/All)	Namespace	Created	Image Name	Operation
hpa-example	Running	1 / 1	default	1 minutes ago	hpa-example:latest	View Log Upgrade More

Total Records: 1

Step 7 Click the workload name to go to the workload details page and obtain the external access address of **hpa-example**. On the **Access Mode** tab, you can view the external access address in the format of *{EIP bound to the load balancer}*:*{Access port}*. *{EIP bound to the load balancer}* indicates the public IP address of the load balancer, and *{Access port}* is the **Service Port** in **Step 5**.

Figure 2-10 Access address



Services

Quota (remaining/total): Load Balancer (**197**/200) Listener (**297**/300) Backend ECS (**996**/1,000)

Service	Selector	Namespace	Service Type	Access Address	Access Port:Container Port/Protocol	Created	Operation
hpa-example	app hpa-example version v1	default	LoadBalancer elb-hpa-example	10.247.69.223 (Cluster IP) [redacted] (Load Balancer Public IP) 192.168.0.186 (Load Balancer Private IP)	80 -> 80 / TCP	1 hours ago	Manage Pod View Events More

Total Records: 1

----End

Using kubectl

Step 1 Install kubectl on the ECS. You can check whether kubectl has been installed by running **kubectl version**. If kubectl has been installed, you can skip this step.

1. Run the following command to download kubectl:
   ```
   cd /home
   curl -LO https://dl.k8s.io/release/{v1.28.0}/bin/linux/amd64/kubectl
   ```
   {v1.28.0} specifies the version. Replace it as needed.
2. Run the following command to install kubectl:
   ```
   chmod +x kubectl
   mv -f kubectl /usr/local/bin
   ```

Step 2 Configure a credential for kubectl to access the cluster.

1. Log in to the **CCE console** and click the cluster name to access the cluster console. In the navigation pane, choose **Overview**.
2. Locate the **Connection Information** area. Click **Configure** next to **kubectl** and view the kubectl connection information.
3. In the window that slides out from the right, locate the **Download the kubeconfig file** area, select **Intranet access** for **Current data**, and download the configuration file.
4. Log in to the ECS where kubectl has been installed and copy and paste the downloaded configuration file (for example, **kubeconfig.yaml**) to the **/home** directory.
5. Save the kubectl authentication file to the configuration file in the **$HOME/.kube** directory.
   ```
   cd /home
   mkdir -p $HOME/.kube
   mv -f kubeconfig.yaml $HOME/.kube/config
   ```
6. Run the kubectl command to check whether the cluster can be accessed.

   For example, to view the cluster information, run the following command:
   ```
   kubectl cluster-info
   ```
   If the following information is displayed, the cluster is connected:
   ```
   Kubernetes control plane is running at https://xx.xx.xx.xx:5443
   CoreDNS is running at https://xx.xx.xx.xx:5443/api/v1/namespaces/kube-system/services/coredns:dns/
   proxy
   To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
   ```

Step 3 Create a YAML file for creating a workload. In this example, the file name is **hpa-example.yaml**. You can change the file name as needed.

```
vim hpa-example.yaml
```

The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hpa-example     # Workload name
spec:
  replicas: 1     # Number of pods
  selector:
    matchLabels:  # Selector for selecting resources with specific labels
      app: hpa-example
  template:
    metadata:
      labels:     # Labels
        app: hpa-example
    spec:
      containers:
      - name: container-1
        image: 'swr.cn-east-3.myhuaweicloud.com/test/hpa-example:latest' # Replace it with the address of the image you have pushed to SWR.
          resources:
            limits:  # The value of limits must be the same as that of requests to prevent flapping during scaling.
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
              memory: 512Mi
      imagePullSecrets:
      - name: default-secret
```

NOTE

Replace **'swr.cn-east-3.myhuaweicloud.com/test/hpa-example:latest'** with the image path in SWR. The image path is the content following **docker push** in the step of **pushing the image to the image repository**.

Press **Esc** to exit editing mode and enter **:wq** to save the file.

Step 4 Run the following command to create the workload:

```
kubectl create -f hpa-example.yaml
```

If information similar to the following is displayed, the workload is being created:

```
deployment.apps/hpa-example created
```

Step 5 Run the following command to check the workload status:

```
kubectl get deployment
```

If the value of **READY** is **1/1**, the pod created for the workload is available. This means the workload has been created.

```
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hpa-example    1/1     1            1           4m59s
```

The following table describes the parameters in the command output.

Table 2-9 Parameters in the command output

Parameter	Example Value	Description
NAME	hpa-example	Workload name.
READY	1/1	The number of available pods/desired pods for the workload.
UP-TO-DATE	1	The number of pods that have been updated for the workload.
AVAILABLE	1	The number of pods available for the workload.
AGE	4m59s	How long the workload has run.

Step 6 Run the following command to create the **nginx-elb-svc.yaml** file for configuring the LoadBalancer Service and associating the Service with the created workload. You can change the file name.

An existing load balancer is used to create the Service. To automatically create a load balancer, see **Using kubectl to Create a Load Balancer**.

```
vim hpa-example-elb-svc.yaml
```

The file content is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: hpa-example      # Service name
  annotations:
    kubernetes.io/elb.id: <your_elb_id>        # Load balancer ID. Replace it with the actual value.
    kubernetes.io/elb.class: performance        # Load balancer type
spec:
  selector:             # The value must be the same as the value of matchLabels in the YAML file of the
workload in Step 3.
     app: hpa-example
  ports:
  - name: service0
    port: 80
    protocol: TCP
    targetPort: 80
  type: LoadBalancer
```

Press **Esc** to exit editing mode and enter **:wq** to save the file.

Table 2-10 Parameters for using an existing load balancer

Parameter	Example Value	Description
kubernetes.io/elb.id	405ef586-0397-45c3-bfc4-xxx	ID of an existing load balancer. **NOTE** The load balancer must meet the following requirements: ● It is in the same VPC as the cluster. ● It is a dedicated load balancer. ● It has a private IP address bound. **How to Obtain**: In the navigation pane of **Network Console**, choose **Elastic Load Balance** > **Load Balancers**. Locate the load balancer. The ID is displayed below the name. Click the name of the load balancer. On the **Summary** tab, verify that the load balancer meets the preceding requirements.
kubernetes.io/elb.class	performance	Load balancer type. The value can only be **performance**, which means that only dedicated load balancers are supported.
selector	app: hpa-example	Selector, which is used by the Service to send traffic to pods with specific labels.
ports.port	8080	The port used by the load balancer as an entry for external traffic. You can use any port.
ports.protocol	TCP	Protocol for the load balancer listener.
ports.targetPort	80	Port used by a Service to access the target container. This port is closely related to the applications running in the container.

Step 7 Run the following command to create a Service:

```
kubectl create -f hpa-example-elb-svc.yaml
```

If information similar to the following is displayed, the Service has been created:

```
service/hpa-example created
```

Step 8 Run the following command to check the Service:

```
kubectl get svc
```

If information similar to the following is displayed, the workload access mode has been configured: You can access the workload using *{External access address}*:*{Service port}*. The external access address is the first IP address displayed for **EXTERNAL-IP**, and the Service port is 8080.

```
NAME          TYPE          CLUSTER-IP     EXTERNAL-IP                 PORT(S)          AGE
kubernetes     ClusterIP      10.247.0.1      <none>                   443/TCP          18h
hpa-example      LoadBalancer    10.247.56.18   xx.xx.xx.xx,xx.xx.xx.xx     8080:30581/TCP   5m8s
```

----End

Step 5: Create an HPA Policy

HPA periodically checks pod metrics, calculates the number of replicas required to handle the increased or decreased load, and then adds the required number of pods for the workload. You can use either of the following methods to create an HPA policy.

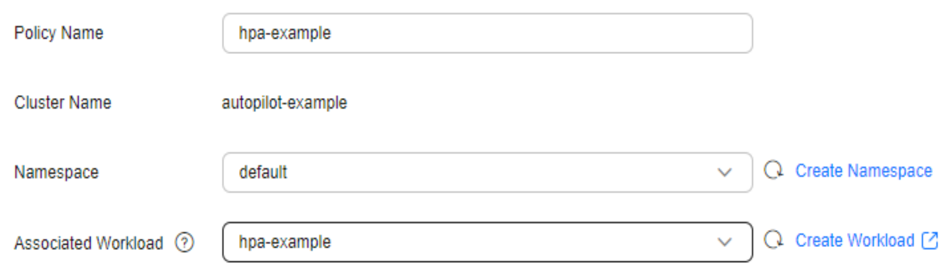
Using the CCE Console

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the default parameters, see **Horizontal Pod Autoscaling**.

Step 1 Return to the cluster console. In the navigation pane, choose **Policies**. In the upper right corner, click **Create HPA Policy**.

Step 2 Configure the parameters. For details about the parameters, see **Table 2-11**.

Figure 2-11 Policy basic Information



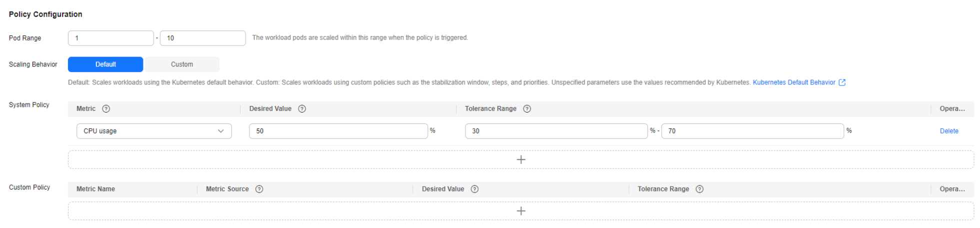
Policy Name	hpa-example
Cluster Name	autopilot-example
Namespace	default
Associated Workload	hpa-example

Table 2-11 Policy basic Information

Parameter	Example Value	Description
Policy Name	hpa-example	Enter a policy name.
Namespace	default	A namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces. After a cluster is created, the **default** namespace is created by default. If there is no special requirement, select **default**.
Associated Workload	hpa-example	Select the workload created in **Step 4: Create a Workload Using the hpa-example Image**. **NOTE** An HPA policy matches the associated workload by name. If the application labels of multiple workloads in the same namespace are the same, the policy will fail to meet the expectation.

Step 3 Configure the policy. For details about the parameters, see **Table 2-12**.

Figure 2-12 Policy configuration



Policy Configuration

Pod Range: 1 - 10 The workload pods are scaled within this range when the policy is triggered.

Scaling Behavior: Default | Custom

Default: Scales workloads using the Kubernetes default behavior. Custom: Scales workloads using custom policies such as the stabilization window, steps, and priorities. Unspecified parameters use the values recommended by Kubernetes. Kubernetes Default Behavior

System Policy

Metric	Desired Value	Tolerance Range	Opera...
CPU usage	50 %	30 % - 70 %	Delete

Custom Policy

Metric Name	Metric Source	Desired Value	Tolerance Range	Opera...

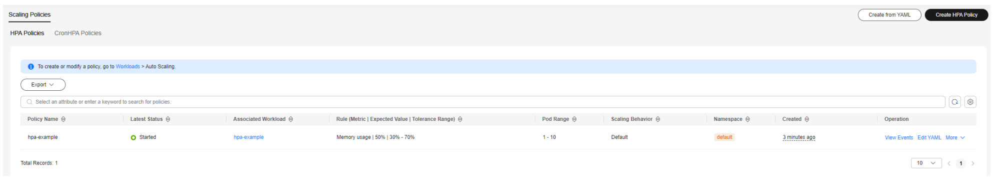
Table 2-12 Policy configuration

Parameter	Example Value	Description
Pod Range	1 to 10	The minimum and maximum numbers of pods that can be created for the workload.
Scaling Behavior	Default	Auto scaling behaviors. The options are as follows: ● **Default**: Default behaviors (such as stabilization window and step) recommended by Kubernetes are used. For more information, see **Horizontal Pod Autoscaling**. ● **Custom**: You can create custom scaling behaviors (such as stabilization window and step) for more flexible configuration. For parameters that are not configured, the default values recommended by Kubernetes are used. For more information, see **Horizontal Pod Autoscaling**.
System Policy	Metric: CPU usage	HPA can trigger pod scaling based on pod resource metrics. ● CPU usage = CPU used by the pod/CPU request ● Memory usage = Memory used by the pod/ Memory request
	Desired Value: 50%	Desired value of the selected metric, which can be used to calculate the number of pods to be scaled. The formula is as follows: Number of pods to be scaled = (Current metric value/Desired value) × Number of current pods

Parameter	Example Value	Description
	Tolerance Range: 30% to 70%	Scale-in and scale-out thresholds. Scaling will not be triggered when the metric value is within the range. This range prevents frequent adjustments caused by slight fluctuation and ensures system stability. Note that the desired value must be within the tolerance range.

Step 4 Click **Create**. You can view the created policy in the HPA policy list. Its status is **Started**.

Figure 2-13 HPA policy started



Scaling Policies

HPA Policies | CronHPA Policies

Create from YAML | Create HPA Policy

To create or modify a policy, go to Workloads > Auto Scaling.

Export

Select an attribute or enter a keyword to search for policies.

Policy Name	Latest Status	Associated Workload	Rule (Metric | Expected Value | Tolerance Range)	Pod Range	Scaling Behavior	Namespace	Created	Operation
hpa-example	Started	hpa-example	Memory usage | 50% | 30% - 70%	1 - 10	Default	default	3 minutes ago	View Events Edit YAML More

Total Records: 1

10 1

----End

Using kubectl

Step 1 Create a YAML file for configuring an HPA policy. In this example, the file name is **hpa-policy.yaml**. You can change it as needed.

```
vim hpa-policy.yaml
```

The file content is as follows:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-example   # HPA policy name
  namespace: default  # Namespace
  annotations:
    # Define the condition for triggering scaling: When the CPU usage is between 30% and 70%, scaling is not performed. Otherwise, scaling is performed.
    extendedhpa.metrics: '[{"type":"Resource","name":"cpu","targetType":"Utilization","targetRange":{"low":"30","high":"70"}}]'
spec:
  scaleTargetRef:     # Associated workload
    kind: Deployment
    name: hpa-example
    apiVersion: apps/v1
  minReplicas: 1      # Minimum number of pods in the workload
  maxReplicas: 10  # Maximum number of pods in the workload
  metrics:
    - type: Resource  # Set the resources to be monitored.
      resource:
        name: cpu       # Set the resource to CPU or memory.
        target:
          type: Utilization    # Set the metric to usage.
          averageUtilization: 50  # The HPA controller keeps the average resource usage of pods at 50%.
```

Press **Esc** to exit editing mode and enter **:wq** to save the file.

Step 2 Create the HPA policy.

```
kubectl create -f hpa-policy.yaml
```

If information similar to the following is displayed, the HPA policy has been created:

```
horizontalpodautoscaler.autoscaling/hpa-example created
```

Step 3 View the HPA policy.

```
kubectl get hpa
```

Information similar to the following is displayed:

```
NAME          REFERENCE                  TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-example   Deployment/hpa-example     <unknown>/50%   1         10        0          10s
```

----End

Step 6: Verify Auto Scaling

Verify that auto scaling can be performed for the **hpa-example** workload. There are two methods for verification: using the console or kubectl.

Using the CCE Console

Step 1 Log in to the ECS used in **Step 3: Create a Compute-intensive Application and Upload Its Image to SWR**.

Step 2 Run the following command in a loop to access the workload. In the command, *ip:port* indicates the access address of the workload, which is the same as *{EIP bound to the load balancer}*:*{Access port}* in **Step 7**.

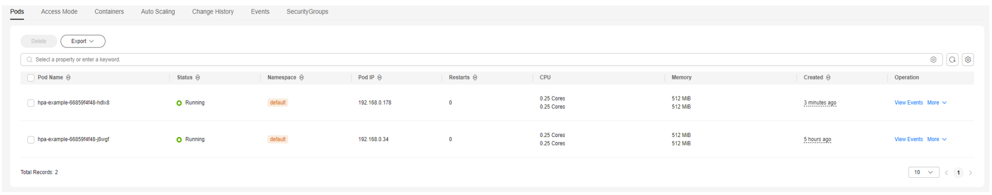
```
while true;do wget -q -O- http://ip:port; done
```

Information similar to the following is displayed:

```
OK!
OK!
...
```

Step 3 Log in to the **CCE console** and click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**. On the page displayed, click the workload name. Click on the right of the search bar to observe the automatic scale-out of the workload. You can find that the number of pods created for the **hpa-example** workload has increased from 1 to 2.

Figure 2-14 Automatic workload scale-out



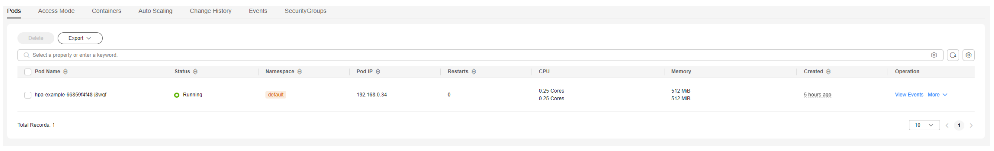
Pod Name	Status	Namespace	Pod IP	Restarts	CPU	Memory	Created	Operation
hpa-example-66859f4f48-hdlx8	Running	default	192.168.0.178	0	0.25 Cores 0.25 Cores	512 MiB 512 MiB	3 minutes ago	View Events More
hpa-example-66859f4f48-j8wgf	Running	default	192.168.0.34	0	0.25 Cores 0.25 Cores	512 MiB 512 MiB	5 hours ago	View Events More

Total Records: 2

Step 4 Return to the ECS, press **Ctrl+C** to stop accessing the workload, and observe the automatic scale-in of the workload.

Return to the CCE console and click on the right of the search bar. The stabilization window for workload scale-in is 300 seconds by default. After the scale-in is triggered, the workload is not scaled in immediately. Instead, it will be cooled down for 300 seconds to prevent short-term fluctuations.

Figure 2-15 Workload scale-in



Pod Name	Status	Namespace	Pod IP	Restarts	CPU	Memory	Created	Operation
hpa-example-66859f4f48-j8wgf	Running	default	192.168.0.34	0	0.25 Cores 0.25 Cores	512 MiB 512 MiB	5 hours ago	View Events More

Total Records: 1

----End

Using kubectl

Step 1 Run the following command in a loop to access the workload. In the command, *ip:port* indicates the access address of the workload, which is the same as *{External access address}*:*{Service port}* in **Step 8**.

```
while true;do wget -q -O- http://ip:port; done
```

Information similar to the following is displayed:

```
OK!
OK!
...
```

Step 2 In the upper left corner of the page, choose **Terminal** > **New Session**. In the terminal that is displayed, run the following command to view the automatic workload scale-out:

```
kubectl get hpa hpa-policy --watch
```

Information similar to the following is displayed:

```
NAME          REFERENCE                  TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-example   Deployment/hpa-example    24%/50%    1         10        1          17m
hpa-example   Deployment/hpa-example    100%/50%   1         10        1          17m
hpa-example   Deployment/hpa-example    100%/50%   1         10        2          18m
hpa-example   Deployment/hpa-example    100%/50%   1         10        2          18m
hpa-example   Deployment/hpa-example    57%/50%    1         10        2          19m
hpa-example   Deployment/hpa-example    57%/50%    1         10        2          20m
...
```

Table 2-13 Parameters in the command output

Parameter	Example Value	Description
NAME	hpa-example	HPA policy name. In this example, the HPA policy name is **hpa-example**.
REFERENCE	Deployment/hpa-example	Object for which the HPA policy takes effect. In this example, the object is a Deployment named **hpa-example**.

Parameter	Example Value	Description
TARGETS	24%/50%	Current and desired values of the metric. In this example, the metric is CPU usage. **24%** indicates the current CPU usage, and **50%** indicates the desired CPU usage.
MINPODS	1	Minimum number of pods allowed by the HPA policy. In this example, the minimum number of pods is 1. When the number of pods is 1, the scale-in will not be performed even if the current CPU usage is lower than the tolerance range.
MAXPODS	10	Maximum number of pods allowed by the HPA policy. In this example, the maximum number of pods is 10. When the number of pods is 10, the scale-out will not be performed even if the current CPU usage is higher than the tolerance range.
REPLICAS	**1**	Number of running pods.
AGE	17m	How long the HPA policy has been used.

- On line 2, the CPU usage of the workload is 100%, which exceeds 70%. As a result, auto scaling is triggered.
- On line 3, the number of pods is increased from 1 to 2.
- On line 5, the CPU usage of the workload decreases to 57%, which is between 30% and 70%. No auto scaling is triggered.

Step 3 In the terminal displayed in **Step 1**, press **Ctrl+C** to stop accessing the workload and observe the automatic scale-in.

On the terminal in **Step 2**, view the command output. Note that the stabilization window for workload scale-in is 300 seconds by default. After the scale-in is triggered, the workload is not scaled in immediately. Instead, it will be cooled down for 300 seconds to prevent short-term fluctuations.

```
NAME          REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
...
hpa-example   Deployment/hpa-example   19%/50%     1         10        2          30m
hpa-example   Deployment/hpa-example   0%/50%      1         10        2          31m
hpa-example   Deployment/hpa-example   0%/50%      1         10        2          35m
hpa-example   Deployment/hpa-example   0%/50%      1         10        1          36m
...
```

- On line 1, the CPU usage of the workload is 19%, which is lower than 30%. As a result, auto scaling is triggered. However, the workload is not scaled in immediately because it is in the cooldown period.

- On line 4, the number of pods is decreased from 2 to 1, which is the minimum number of pods. In this case, auto scaling is not triggered.

----End

Follow-up Operations: Releasing Resources

To avoid additional expenditures, release resources promptly if you no longer need them.

NOTICE

- Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be recovered.
- VPC resources (such as endpoints, NAT gateways, and EIPs for SNAT) associated with a cluster are retained by default when the cluster is deleted. Before deleting a cluster, ensure that the resources are not used by other clusters.

Step 1 Log in to the **CCE console**. In the navigation pane, choose **Clusters**.

Step 2 Locate the cluster, click ⋯ in the upper right corner of the cluster card, and click **Delete Cluster**.

Step 3 In the displayed **Delete Cluster** dialog box, delete related resources as prompted.

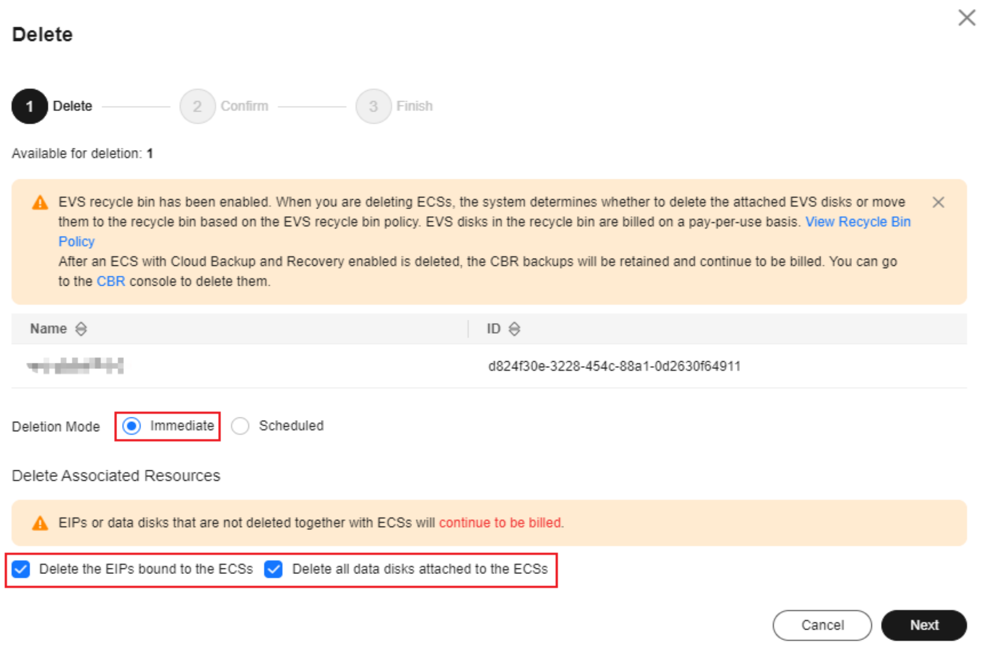
Enter **DELETE** and click **Yes** to start deleting the cluster.

It takes 1 to 3 minutes to delete a cluster. If the cluster name disappears from the cluster list, the cluster has been deleted.

Step 4 Log in to the **ECS console**. In the navigation pane, choose **Elastic Cloud Server**. Locate the target ECS, click **More** > **Delete**.

In the displayed dialog, select **Delete the EIPs bound to the ECSs** and **Delete all data disks attached to the ECSs**, and click **Next**.

Figure 2-16 Deleting ECSs



Delete

1 Delete — 2 Confirm — 3 Finish

Available for deletion: **1**

EVS recycle bin has been enabled. When you are deleting ECSs, the system determines whether to delete the attached EVS disks or move them to the recycle bin based on the EVS recycle bin policy. EVS disks in the recycle bin are billed on a pay-per-use basis. View Recycle Bin Policy
After an ECS with Cloud Backup and Recovery enabled is deleted, the CBR backups will be retained and continue to be billed. You can go to the CBR console to delete them.

Name	ID
[redacted]	d824f30e-3228-454c-88a1-0d2630f64911

Deletion Mode [✓ Immediate] Scheduled

Delete Associated Resources

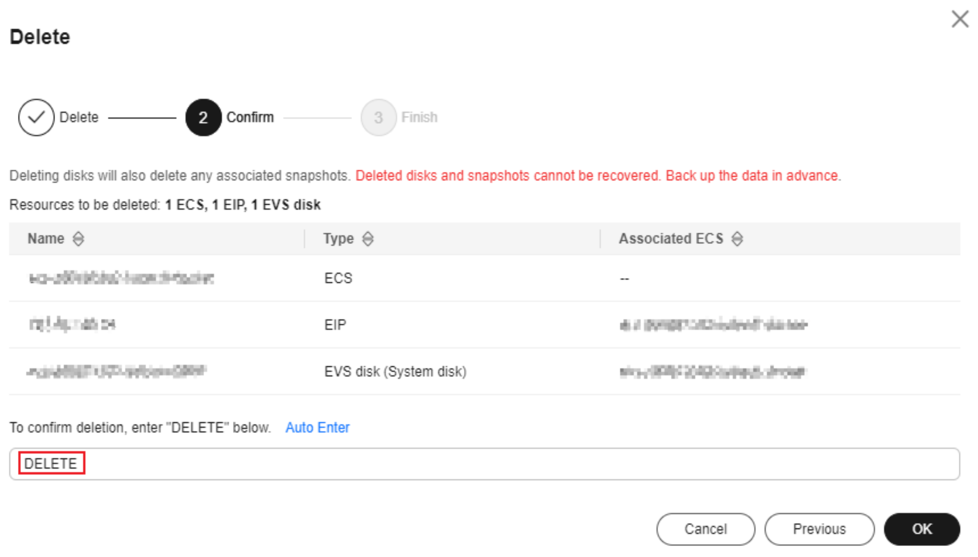
EIPs or data disks that are not deleted together with ECSs will continue to be billed.

☑ Delete the EIPs bound to the ECSs ☑ Delete all data disks attached to the ECSs

Enter **DELETE** and click **OK** to start deleting the ECS.

It takes 0.5 minutes to 1 minute to delete an ECS. If the ECS name disappears from the ECS list, the ECS has been deleted.

Figure 2-17 Confirming the deletion



Delete

✓ Delete — 2 Confirm — 3 Finish

Deleting disks will also delete any associated snapshots. Deleted disks and snapshots cannot be recovered. Back up the data in advance.

Resources to be deleted: **1 ECS, 1 EIP, 1 EVS disk**

Name	Type	Associated ECS
[redacted]	ECS	--
[redacted]	EIP	[redacted]
[redacted]	EVS disk (System disk)	[redacted]

To confirm deletion, enter "DELETE" below. Auto Enter

DELETE

----End