

Cloud Container Engine Autopilot

Quick Start

Issue 01
Date 2024-11-12



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

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

Trademarks and Permissions



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

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

Notice

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

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

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

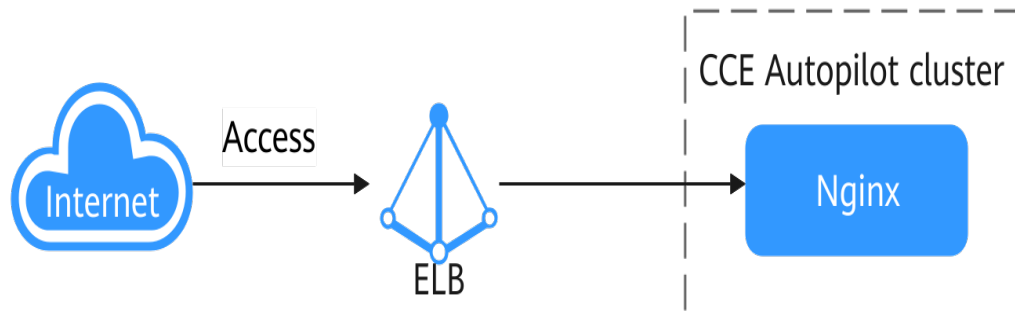
Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 Deploying an Nginx Workload in a CCE Autopilot Cluster.....	1
2 Scaling a Workload Using an HPA Policy.....	16

1 Deploying an Nginx Workload in a CCE Autopilot Cluster

CCE Autopilot is a serverless container service. You can deploy applications without purchasing or managing nodes. This reduces O&M costs and improves application reliability and scalability. Nginx is a high-performance, open-source web server that can also be used as a reverse proxy, load balancer, and HTTP cache. Nginx is used as an example to describe how you can create a CCE Autopilot cluster and deploy a workload in the cluster.



Procedure

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	Description	Billing
Step 2: Create a CCE Autopilot Cluster	Create a CCE Autopilot cluster on the CCE console to simplify the management and operations of Kubernetes clusters.	Cluster management and VPC endpoints are billed. For details, see Billing .
Step 3: Create a Workload and Access It	Create a workload in the cluster to run your containers and create a Service for the workload to enable Internet access.	Pods are billed. 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 on the displayed page and select a region.

Step 3 When you log in to the CCE console in a region for the first time, 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. You can go to the IAM console, choose **Agencies**, and click **cce_admin_trust** to view the delegation records of each region. For details, see [Account Delegation](#).

NOTE

CCE may fail to run as expected if the Tenant Administrator permissions are not assigned. Therefore, do not delete or modify the **cce_admin_trust** agency when using CCE.

----End

Step 2: Create a CCE Autopilot Cluster

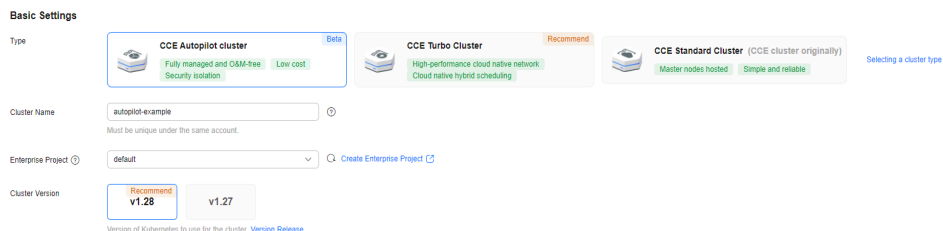
Create a CCE Autopilot cluster on the CCE console to simplify the management and operations of Kubernetes clusters.

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.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see [Buying a CCE Autopilot Cluster](#).

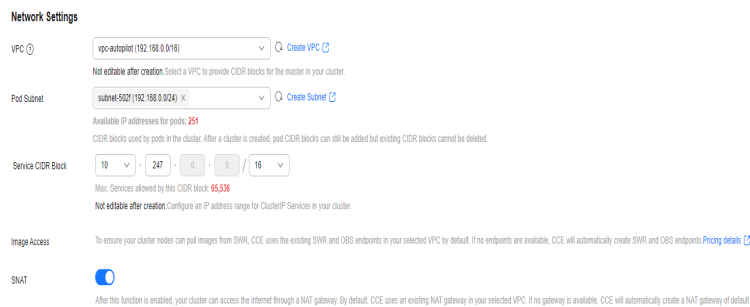


Parameter	Example Value	Description
Type	CCE Autopilot cluster	<p>CCE allows you to create various types of clusters for diverse needs. It provides highly reliable, secure, business-class container services.</p> <ul style="list-style-type: none"> • CCE standard clusters provide highly reliable and secure 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 O&M-free. They greatly reduce O&M costs and improve application reliability and scalability. <p>For more information about cluster types, see Cluster Comparison.</p>
Cluster Name	autopilot-example	Enter a name for the cluster.

Parameter	Example Value	Description
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 the network information.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see [Buying a CCE Autopilot Cluster](#).



Parameter	Example Value	Description
VPC	vpc-autopilot	Select the VPC where the cluster is located. If no option is available, click Create VPC 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 container is located. If no option is available, click Create Subnet to create one. For details, see Creating a VPC and Subnet . The container subnet determines the maximum number of containers in a cluster. You can add more subnets after the cluster is created.

Parameter	Example Value	Description
Service CIDR Block	10.247.0.0/16	Specify the CIDR block for the Service. This CIDR block determines the maximum number of IP addresses that can be used by the Service and is used by containers in the same cluster to access each other. The Service CIDR block cannot be changed after the cluster is created.
Image Access	-	To ensure that the nodes in a cluster can pull images from SWR, existing endpoints in the selected VPC are used by default. If there are no endpoints in the VPC, CCE automatically creates the endpoints for SWR and OBS. VPC endpoints are billed. For details, see VPC Endpoint Price Calculator .
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 automatically creates a NAT gateway with default specifications, binds an EIP to the NAT gateway, and configures an SNAT rule. The NAT gateway will be billed. For details, see NAT Gateway Price Calculator .

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

This example only includes the mandatory add-ons that are automatically installed.

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

In this example, only mandatory add-ons are installed.

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.



----End

Step 3: Create a Workload and Access It

You can create a workload in a cluster to deploy Nginx in containers for high resource utilization and automatic management. You also need to 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 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 the basic information about the workload.

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

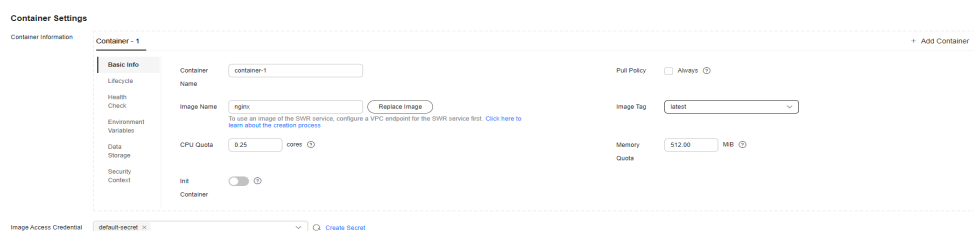
The screenshot shows the 'Basic Info' configuration page for creating a workload. It includes the following fields and options:

- Workload Type:** Four buttons are shown: 'Deployment' (selected), 'StatefulSet', 'Job', and 'Cron Job'. A warning message below reads: 'Switching workload type requires reconfiguring workload parameters'.
- Workload Name:** A text input field containing 'nginx'.
- Namespace:** A dropdown menu showing 'default' with a 'Create Namespace' link next to it.
- Pods:** A numeric input field set to '1' with minus and plus buttons.
- Cluster Name:** A text input field containing 'autopilot-example CCE Autopilot'.
- Description:** A large text area with a placeholder 'A maximum of 200 characters are supported.' and a character count '0/200'.

Parameter	Example Value	Description
Workload Type	Deployment	<p>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, updating, and restoration.</p> <ul style="list-style-type: none"> • 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. • Job: a one-off job. After the job is complete, the pods are automatically deleted. • CronJob: a time-based job runs a specified job in a specified period. <p>For more information about workloads, see Workload Overview.</p> <p>Nginx is mainly used to forward requests, balance loads, and distribute static content. You do not need to store any Nginx persistent data locally. Therefore, Nginx is deployed as a Deployment in this example.</p>
Workload Name	nginx	Enter a name for the workload.
Namespace	default	<p>A namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces.</p> <p>After a cluster is created, the default namespace is created by default. If there is no special requirement, select default.</p>
Pods	1	Enter the number of pods.

Step 4 Configure the containers.

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



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 Specify the Service settings.

Click the plus sign (+) under **Service Settings**. The **Create Service** page is displayed.

In this example, external access to Nginx is required. So you need to create a Service of the LoadBalancer type.

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.

✕

Create Service

Service Name:

Service Type:

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: Dedica... Network (TCP/UDP) & ... Use ex... elb-nginx Create Load Balancer

Supports only dedicated load balancers of 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 [↗](#)

Protocol	Container Port	Service Port	Frontend Protocol	Operation
TCP	-	80	+	-
		8080	+	TCP
				Delete
+				

Listener: Access Control: Not configured

Advanced Options [+ Add advanced options](#)

Annotation: Key = Value Confirm [Quick Links](#)

Cancel
OK

Parameter	Example Value	Description
Service Name	nginx	Enter a name for the Service.
Service Type	LoadBalancer	Select a Service type, which determines how the workload is accessed. <ul style="list-style-type: none"> 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. For more information about the Service types, see Service .

Parameter	Example Value	Description
Load Balancer	Dedicated Network (TCP/UDP) & Application (HTTP/HTTPS) Use existing elb-nginx	<ul style="list-style-type: none"> Select Use existing if there is a load balancer available. <p>NOTE You can only select a dedicated load balancer that is in the same VPC as the cluster and handles network traffic.</p> <ul style="list-style-type: none"> If there is no available load balancer, select Auto create to create one with an EIP bound. For details, see Creating a LoadBalancer Service.
Ports	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. If the nginx image is used, set this port to 80 .
	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.

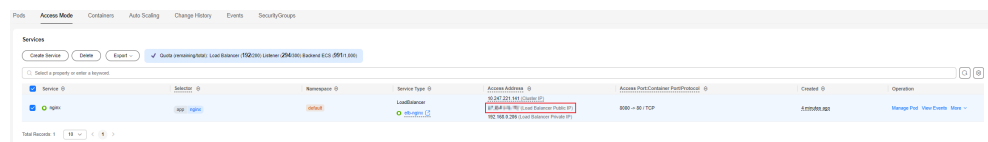
Step 6 Click **Create Workload**.

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



Step 7 Obtain the external access address of Nginx.

Click the name (**nginx**) of the workload to go to its details page. On the **Access Mode** tab, view the access address in the format of *{EIP bound to the load balancer}:{Access port}*.



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



----End

Using kubectl

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

- Perform operations 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](#).
- Perform operations on the ECS that is in the same VPC as the cluster and with an EIP bound. For details, see [Purchasing and Using a Linux ECS](#). You also need to install kubectl and [connect to the cluster through kubectl](#).

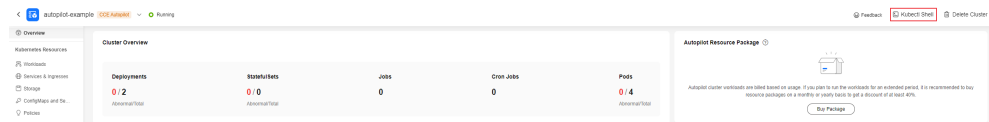
The following uses the first method as an example to describe how you can use kubectl to create an Nginx workload.

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

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

NOTE

Using CloudShell to connect to a cluster is only available in some regions. For details, see the management console.



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.

```
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 for selecting resources with specific labels
      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.

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 Create a Service of the LoadBalancer type for the workload.

An existing load balancer is used to create the Service. To automatically create a load balancer, see [Using kubectl to Create a Load Balancer](#).

Create a YAML file for creating the Service. In this example, the file name is **nginx-elb-svc.yaml**. You can change it as needed.

```
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:
    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.

Parameter	Example Value	Description
kubernetes.io/elb.id	405ef586-0397-45c3-bfc4-xxx	ID of an existing load balancer. NOTE You can only select a dedicated load balancer that is in the same VPC as the cluster and handles network traffic. In the navigation pane of the network console, choose Elastic Load Balance > Load Balancers . Locate the load balancer. The ID is displayed below the name.
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. NOTICE The value must be the same as that of matchLabels in the YAML file for creating the workload. In this example, the value is app: nginx .
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. 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
```

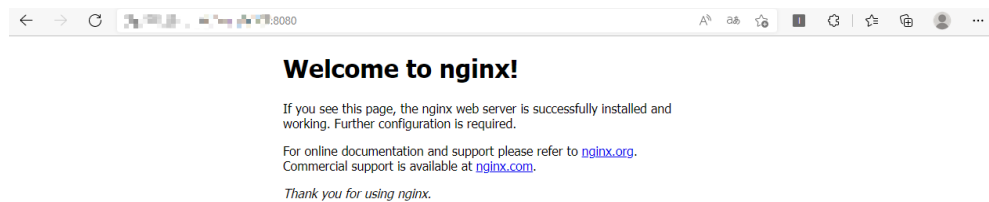
Step 8 Run the following command to check the Service:

```
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.




----End

Follow-up Operations: Releasing Resources

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

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, select all resources to be deleted, including the NAT gateway, EIP configured for the SNAT rule, and endpoints.

Delete Cluster ✕

Are you sure you want to release the following items? Deleted data cannot be recovered.

Cluster Name	Cluster Version	Created
autopilot-example	v1.28	Sep 11, 2024 20:23:37 GMT+08:00

- Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be restored.

- Before deleting a cluster, ensure the VPC-level resources (such as VPC endpoints, NAT gateways, and SNAT outbound EIPs) associated with the cluster are not used.

Please select the resources you want to delete simultaneously:

Resource Type	Operation
Network resources (ELB, etc.)	<input type="checkbox"/> Delete (only delete automatically created ELB resources)
NAT Gateway	To delete VPC resources, go to the Network Console....
SNAT EIP	To delete VPC resources, go to the Network Console....
Endpoint	To delete VPC resources, go to the Network Console....

To confirm deletion, enter "DELETE" below.

DELETE

Closing or refreshing the current page during the deletion may result in residual resources. The dialog box will be automatically closed after the deletion is complete.

No
Yes

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

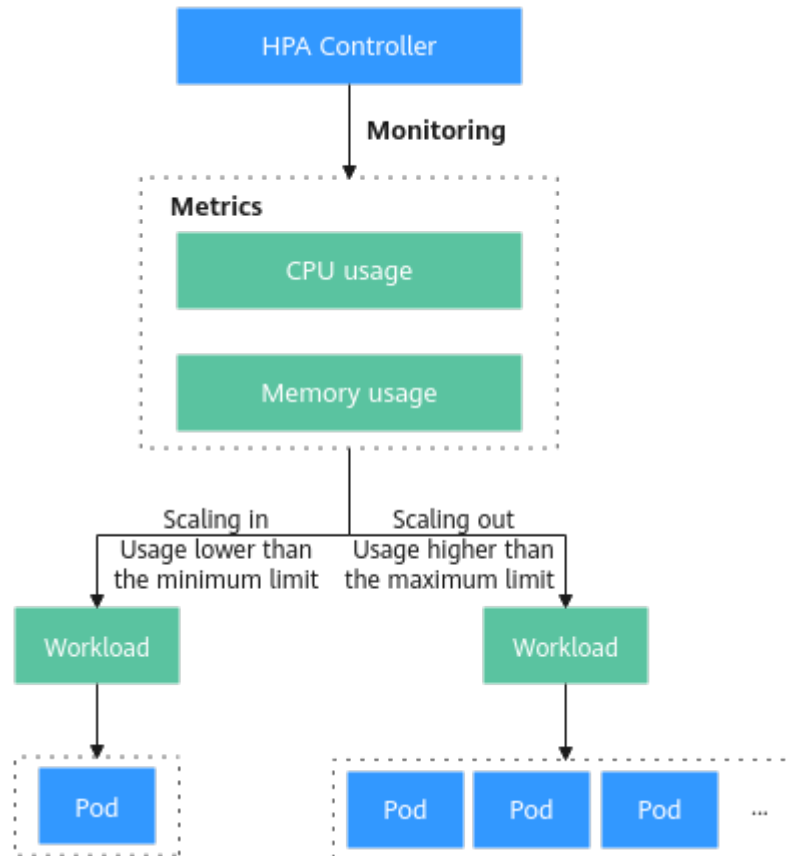
It takes 1 to 3 minutes to delete a cluster.

----**End**

2 Scaling a Workload Using an HPA Policy

The best way to handle surging traffic is to automatically adjust the number of pods for running the workloads based on the traffic volume or resource usage, which is called scaling. In CCE standard or CCE Turbo clusters, configuring auto scaling for workloads requires joint scaling policies for nodes and workloads: a Cluster AutoScaling (CA) policy and a Horizontal Pod Autoscaling (HPA) policy. CA is responsible for auto scaling of nodes (to prevent workload creation failures due to insufficient resources), and HPA takes care of auto scaling of workloads. 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.

HPA controls the number of pods created for a workload. It periodically checks pod metrics such as CPU usage and memory usage, calculates the number of replicas required to handle the increased or decreased load, and then adjusts the number of replicas for the workload. This example describes how you can use HPA to adjust the scale of a workload running in a CCE Autopilot cluster.



Procedure

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 to simplify the management and operations of Kubernetes clusters.	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 stress tests, build an image from the application, and upload the image to SWR, so that the image can be pulled 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	Description	Billing
Step 4: Create a Workload Using the hpa-example Image	Use the image to create a workload and a Service of the LoadBalancer type for the workload so that the workload can be accessed from the public network.	Pods are billed. For details, see Billing .
Step 5: Create an HPA Policy	Create an HPA policy for the workload to control the number of pod replicas 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 on the displayed page and select a region.

Step 3 When you log in to the CCE console in a region for the first time, 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. You can go to the IAM console, choose **Agencies**, and click

`cce_admin_trust` to view the delegation records of each region. For details, see [Account Delegation](#).

NOTE

CCE may fail to run as expected if the Tenant Administrator permissions are not assigned. Therefore, do not delete or modify the `cce_admin_trust` agency when using CCE.

----End

Step 2: Create a CCE Autopilot Cluster

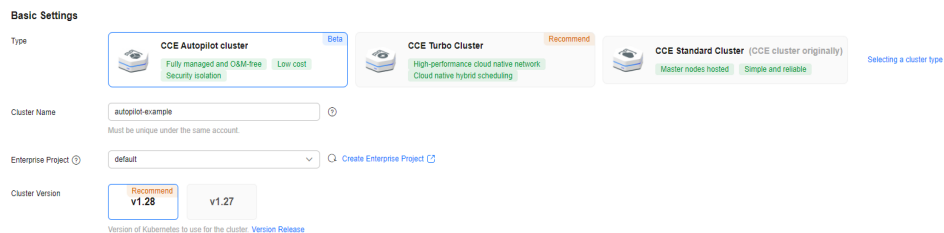
Create a CCE Autopilot cluster to simplify the management and operations of Kubernetes clusters.

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.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see [Buying a CCE Autopilot Cluster](#).

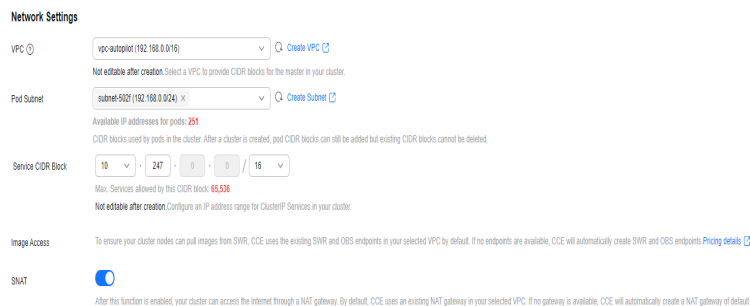


Parameter	Example Value	Description
Type	CCE Autopilot cluster	<p>CCE allows you to create various types of clusters for diverse needs. It provides highly reliable, secure, business-class container services.</p> <ul style="list-style-type: none"> • CCE standard clusters provide highly reliable and secure 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 O&M-free. They greatly reduce O&M costs and improve application reliability and scalability. <p>For more information about cluster types, see Cluster Comparison.</p>

Parameter	Example Value	Description
Cluster Name	autopilot-example	Enter a name for the cluster.
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 the network information.

In this example, only some mandatory parameters are described. You can keep the default values for other parameters. For details about the parameters, see [Buying a CCE Autopilot Cluster](#).



Parameter	Example Value	Description
VPC	vpc-autopilot	Select the VPC where the cluster is located. If no option is available, click Create VPC 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 container is located. If no option is available, click Create Subnet to create one. For details, see Creating a VPC and Subnet . The container subnet determines the maximum number of containers in a cluster. You can add more subnets after the cluster is created.

Parameter	Example Value	Description
Service CIDR Block	10.247.0.0 /16	Specify the CIDR block for the Service. This CIDR block determines the maximum number of IP addresses that can be used by the Service and is used by containers in the same cluster to access each other. The Service CIDR block cannot be changed after the cluster is created.
Image Access	-	To ensure that the nodes in a cluster can pull images from SWR, existing endpoints in the selected VPC are used by default. If there are no endpoints in the VPC, CCE automatically creates endpoints for SWR and OBS. VPC endpoints are billed. For details, see VPC Endpoint Price Calculator .
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 automatically creates a NAT gateway with default specifications, binds an EIP to the NAT gateway, and configures an SNAT rule. The NAT gateway will be billed. For details, see NAT Gateway Price Calculator .

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

This example only includes the mandatory add-ons that are automatically installed.

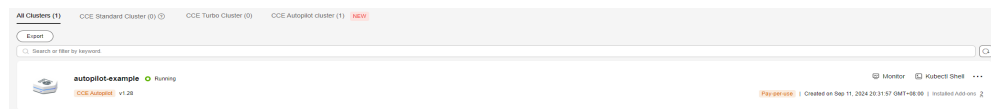
Step 5 Click **Next: Add-on Configuration** to configure the selected add-ons. Mandatory add-ons cannot be configured.

In this example, only mandatory add-ons are installed.

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.



----End

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

Create a compute-intensive application, which will be used to verify whether the HPA policy can automatically adjust the number of pods based on the vCPU and

memory usages. Create an image using the application and then upload the image to SWR for deployment and management in the cluster. This step involves command line operations. 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](#). 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 1 Log in to the ECS. For details, see [Logging In to a Linux ECS using CloudShell](#).

Step 2 Install Docker.

1. Check whether Docker has been installed on the ECS. If Docker has been installed, skip this step.

```
docker --version
```

If the following information is displayed, Docker is not installed:

```
-bash: docker: command not found
```

2. 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. Verify the installation.

```
docker --version
```

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

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

Step 3 Create a compute-intensive application and use it to build an image.

1. Create a PHP file named **index.php**. You can name the file as needed. 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!".

```
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. Compile a **Dockerfile** file 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 permission 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.

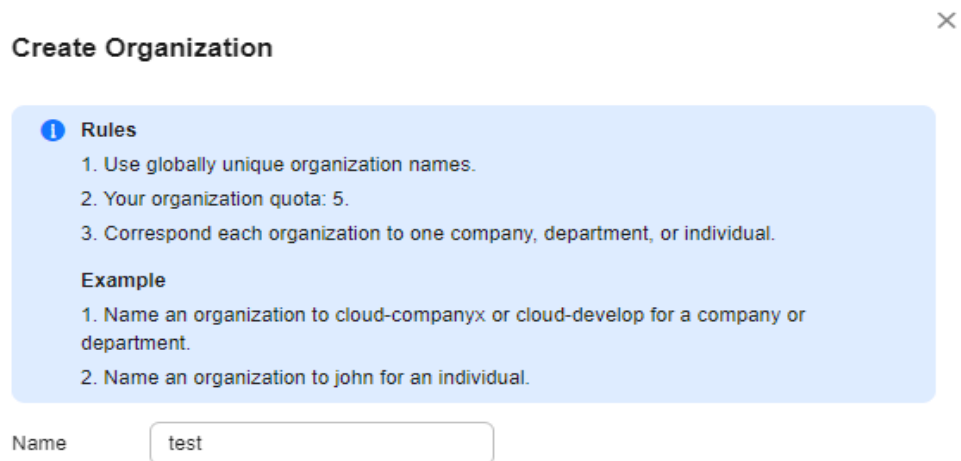
```
docker build -t hpa-example:latest .
```

Step 4 Upload the **hpa-example** image to SWR.

1. Create an image organization. If there is already an organization, skip this step.

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**.



2. Use the ECS to log in to the SWR console.

In the navigation pane, choose **My Images**. In the upper right corner, click **Upload Through Client**. In the displayed dialog, click **Generating Login Instructions** and then click the copy icon to copy the temporary login command.

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
```

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 .

Parameter	Example Value	Description
{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 uploaded image.

----End

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

Use the **hpa-example** image to create a workload and 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 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 the basic information about the workload.

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

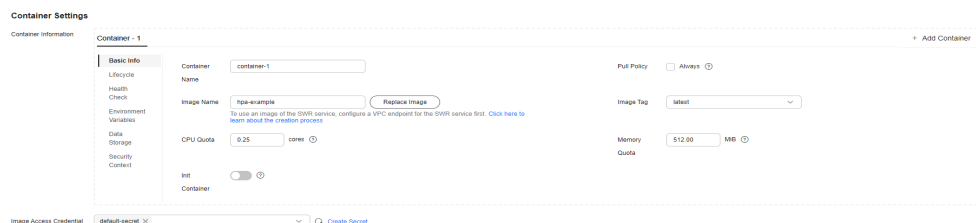
The screenshot shows the 'Basic Info' configuration page for a workload. It includes the following fields and options:

- Workload Type:** Deployment (selected), StatefulSet, Job, Cron Job.
- Workload Name:** hpa-example
- Namespace:** default (with a 'Create Namespace' link)
- Pods:** 1 (with minus and plus buttons)
- Cluster Name:** autopilot-example CCE Autopilot
- Description:** A maximum of 200 characters are supported.

Parameter	Example Value	Description
Workload Type	Deployment	<p>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, updating, and restoration.</p> <ul style="list-style-type: none"> • 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. • Job: a one-off job. After the job is complete, the pods are automatically deleted. • CronJob: a time-based job runs a specified job in a specified period. <p>For more information about workloads, see Workload Overview.</p>
Workload Name	hpa-example	Enter a name for the workload.
Namespace	default	<p>A namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces.</p> <p>After a cluster is created, the default namespace is created by default. If there is no special requirement, select default.</p>
Pods	1	Enter the number of pods.

Step 4 Configure the containers.

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



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 uploaded hpa-example 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 Specify the Service settings.

Click the plus sign (+) under **Service Settings**. The **Create Service** page is displayed.

In this example, external access to **hpa-example** is required. So you need to create a Service of the LoadBalancer type.

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.

Create Service
✕

Service Name

Service Type

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](#)

Parameter	Example Value	Description
Service Name	hpa-example	Enter a Service name.
Service Type	LoadBalancer	Select a Service type, which determines how the workload is accessed. <ul style="list-style-type: none"> 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. For more information about the Service types, see Service .

Parameter	Example Value	Description
Load Balancer	Dedicated Network (TCP/UDP) & Application (HTTP/HTTPS) Use existing elb-hpa-example	<ul style="list-style-type: none"> Select Use existing if there is a load balancer available. <p>NOTE You can only select a dedicated load balancer that is in the same VPC as the cluster and handles network traffic.</p> <ul style="list-style-type: none"> If there is no available load balancer, select Auto create to create one with an EIP bound. For details, see Creating a LoadBalancer Service.
Ports	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.

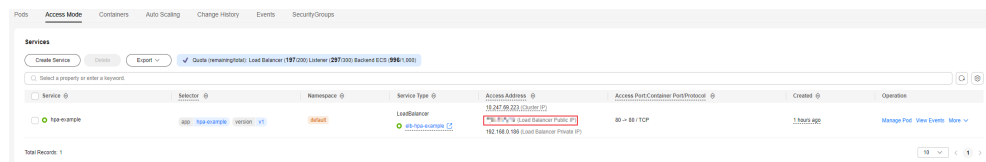
Step 6 Click **Create Workload**.

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



Step 7 Obtain the external access address of **hpa-example**.

Click the name (**hpa-example**) of the workload to go to its details page. On the **Access Mode** tab, you can view the external access address in the format of *{EIP bound to the load balancer}:{Access port}*.



----End

Using kubectl

The ECS used in [Step 3: Create a Compute-intensive Application and Upload Its Image to SWR](#) is still used in this step. You also need to install kubectl and [connect to the cluster through kubectl](#).

Step 1 Install kubectl.

You can check whether kubectl has been installed by running **kubectl version**. If kubectl has been installed, you can skip this step.

The Linux environment is used as an example to describe how to install and configure kubectl. For more installation methods, see [kubectl](#).

1. Log in to the ECS and 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. 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](#). Click the cluster name to access the cluster console.
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 the kubectl client 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/coresdns: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.

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 Create a Service of the LoadBalancer type for the workload.

An existing load balancer is used to create the Service. To automatically create a load balancer, see [Using kubectl to Create a Load Balancer](#).

Create a YAML file for creating the Service. In this example, the file name is **nginx-elb-svc.yaml**. You can change it as needed.

```
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:
    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.

Parameter	Example Value	Description
kubernetes.io/elb.id	405ef586-0397-45c3-bfc4-xxx	ID of an existing load balancer. In the navigation pane of the network console, choose Elastic Load Balance > Load Balancers . Locate the load balancer. The ID is displayed below the name.
kubernetes.io/elb.class	performance	Load balancer type. The value can only be performance , which means that only dedicated load balancers are supported. NOTE If a LoadBalancer Service accesses an existing dedicated load balancer, the dedicated load balancer must support TCP/UDP networking.
selector	app: hpa-example	Selector, which is used by the Service to send traffic to pods with specific labels. NOTICE The value must be the same as that of matchLabels in the YAML file for creating the workload. In this example, the value is app: hpa-example .
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.

Parameter	Example Value	Description
ports.target Port	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 replicas for the workload. For more information about HPA, see [How HPA Works](#). 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](#).

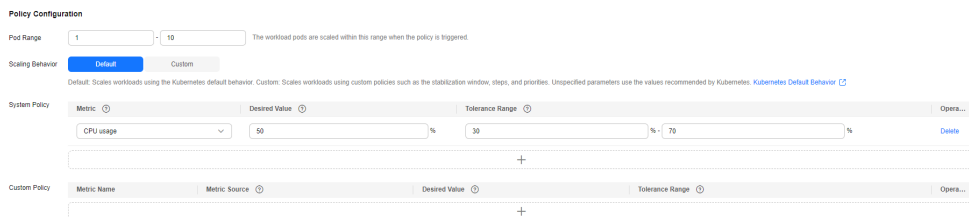
Step 1 Return to the cluster console. In the navigation pane, choose **Policies**. In the upper right corner, click **Create HPA Policy**.

Step 2 Enter basic policy information.

Policy Name	<input type="text" value="hpa-example"/>
Cluster Name	autopilot-example
Namespace	<input type="text" value="default"/> Q Create Namespace
Associated Workload ?	<input type="text" value="hpa-example"/> Q Create Workload

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.

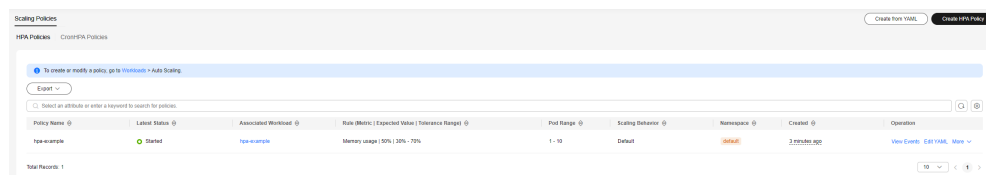


The screenshot shows the 'Policy Configuration' interface. At the top, 'Pod Range' is set to '1 - 10'. Below that, 'Scaling Behavior' is set to 'Default'. Under 'System Policy', the metric is 'CPU usage', the desired value is '50%', and the tolerance range is '30 - 70%'. There is also a 'Custom Policy' section which is currently empty.

Parameter	Example Value	Description
Pod Range	1-10	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: <ul style="list-style-type: none"> ● 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.

Parameter	Example Value	Description
System Policy	Metric: CPU usage	<p>HPA can trigger pod scaling based on pod resource metrics.</p> <ul style="list-style-type: none"> 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
	Tolerance Range: 30% to 70%	Range at which scaling will not be triggered. Scaling will not be triggered when the metric value is within the range. 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**.



----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
```

```
maxReplicas: 10
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](#).

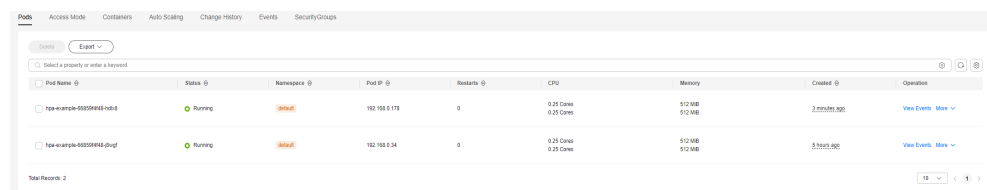
```
while true;do wget -q -O- http://ip:port; done
```

Information similar to the following is displayed:

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


Step 3 Observe the automatic scale-out process.

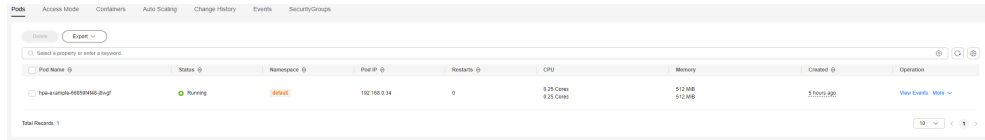
Log in to the [CCE console](#). Click the cluster name to access the cluster console. In the navigation pane, choose **Workloads**. Click the workload name and click  on the right of the search bar. The number of pods created for the **hpa-example** workload increases from 1 to 2.



Step 4 Observe the automatic scale-in process.

Return to the ECS and use **Ctrl+c** to stop accessing the workload.

Return to the CCE console and click  on the right of the search bar. 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.



----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 Open a new terminal and observe the automatic scale-out process.

```
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
...
```

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 expected 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, 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 Observe the automatic scale-in process.

On the terminal in **Step 1**, use **Ctrl+c** to stop accessing the workload.

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%. 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.


- 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, select all resources to be deleted, including the NAT gateway, EIP configured for the SNAT rule, and endpoints.

Delete Cluster

×

Are you sure you want to release the following items? Deleted data cannot be recovered.

Cluster Name	Cluster Version	Created
autopilot-example	v1.28	Sep 11, 2024 20:23:37 GMT+08:00

 - Deleting a cluster will delete the workloads and Services in the cluster, and the deleted data cannot be restored.

- Before deleting a cluster, ensure the VPC-level resources (such as VPC endpoints, NAT gateways, and SNAT outbound EIPs) associated with the cluster are not used.

Please select the resources you want to delete simultaneously:

Resource Type	Operation
Network resources (ELB, etc.)	<input type="checkbox"/> Delete (only delete automatically created ELB resources)
NAT Gateway	To delete VPC resources, go to the Network Console....
SNAT EIP	To delete VPC resources, go to the Network Console....
Endpoint	To delete VPC resources, go to the Network Console....

To confirm deletion, enter "DELETE" below.

DELETE

Closing or refreshing the current page during the deletion may result in residual resources. The dialog box will be automatically closed after the deletion is complete.

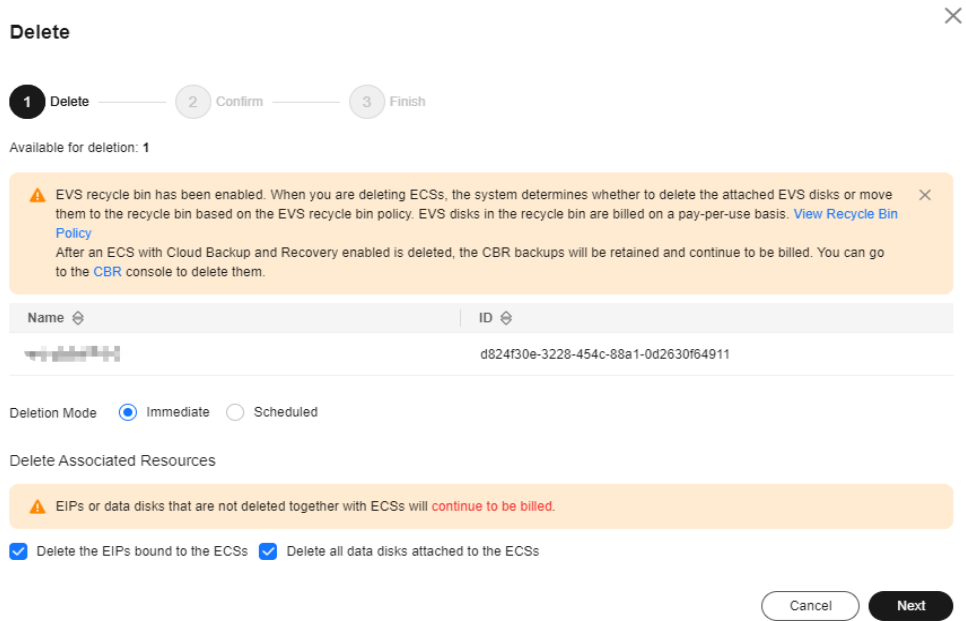
No
Yes

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

It takes 1 to 3 minutes to delete a cluster.

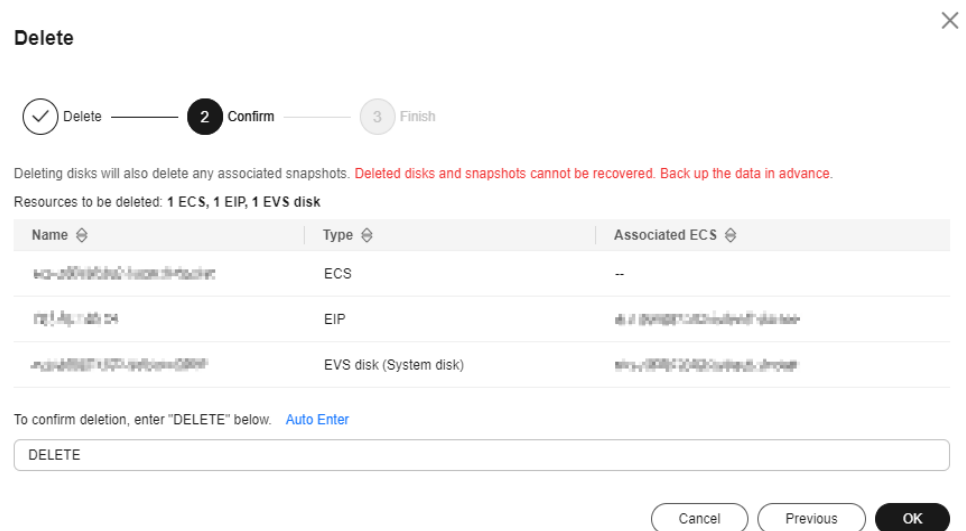
- Step 4** Log in to the ECS console. In the navigation pane, choose **Elastic Cloud Server**. Locate the target ECS, click **More > Delete**.

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



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

It takes 0.5 minutes to 1 minute to delete an ECS.



----End