

# Cloud Container Engine

## Getting Started

**Issue**            01  
**Date**             2024-09-29



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

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

## **Trademarks and Permissions**



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

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

## **Notice**

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

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

---

# Contents

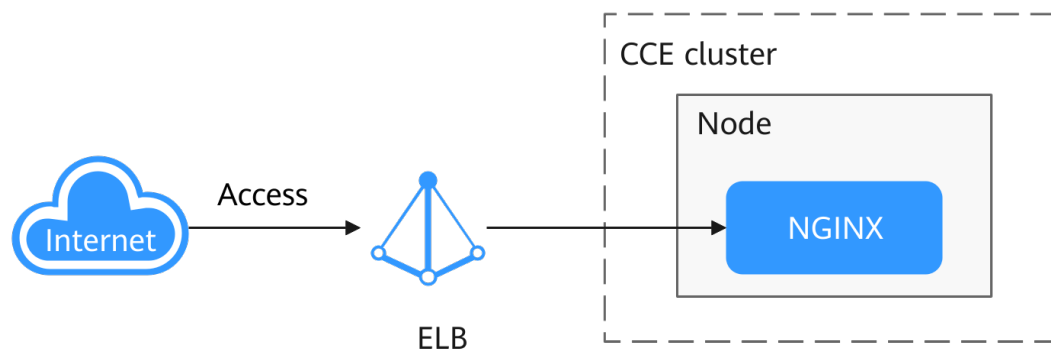
---

<b>1 Deploying an NGINX Deployment in a CCE Cluster.....</b>	<b>1</b>
<b>2 Deploying a WordPress StatefulSet in a CCE Cluster.....</b>	<b>18</b>
<b>3 Deploying an Application in a CCE Cluster Using a Helm Chart.....</b>	<b>45</b>

# 1 Deploying an NGINX Deployment in a CCE Cluster

Deployments are a type of workload in Kubernetes. They are ideal for applications that do not require data consistency and durability, such as web and application servers. Each pod in a Deployment is independent of the others, and there is no difference in running status between them. This means that if one pod fails, requests can be redirected to other healthy pods, ensuring uninterrupted services. Deployment pods are also independent from one another and can be replaced with new ones. You can easily adjust the number of pods based on real-time service requirements, such as adding more during peak hours to handle increased traffic.

This section uses the lightweight web server NGINX as an example to describe how to deploy a Deployment in a CCE cluster.



## Procedure

Step	Description
<b>Preparations</b>	Register a Huawei account and top up the account.
<b>Step 1: Enable CCE for the First Time and Perform Authorization</b>	Obtain the required permissions for your account when you use the CCE service in the current region for the first time.

Step	Description
<a href="#">Step 2: Create a Cluster</a>	Create a CCE cluster to provide Kubernetes services.
<a href="#">Step 3: Create a Node Pool and Nodes in the Cluster</a>	Create a node in the cluster to run your containerized applications.
<a href="#">Step 4: Create a Workload and Access It</a>	Create a workload in the cluster to run your containers and create a Service for the workload to enable Internet access.
<a href="#">Follow-up Operations: Releasing Resources</a>	To avoid additional charges, delete the cluster resources promptly if you no longer require them after practice.


## Preparations

- Before starting, register a Huawei account 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

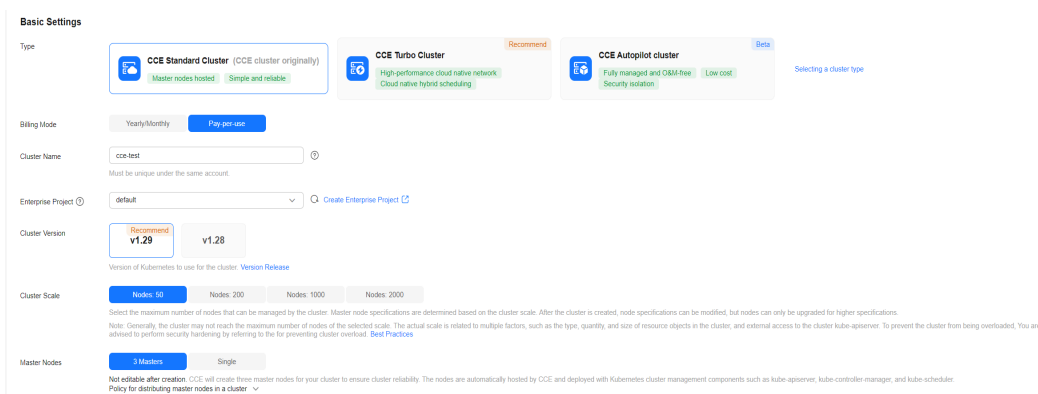
## Step 2: Create a Cluster

### Step 1 Log in to the [CCE console](#).

- If you have no clusters, click **Buy Cluster** on the wizard page.
- If you have CCE clusters, choose **Clusters** in the navigation pane, click **Buy Cluster** in the upper right corner.

### Step 2 Configure basic cluster parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the parameter configurations, see [Buying a CCE Standard/Turbo Cluster](#).

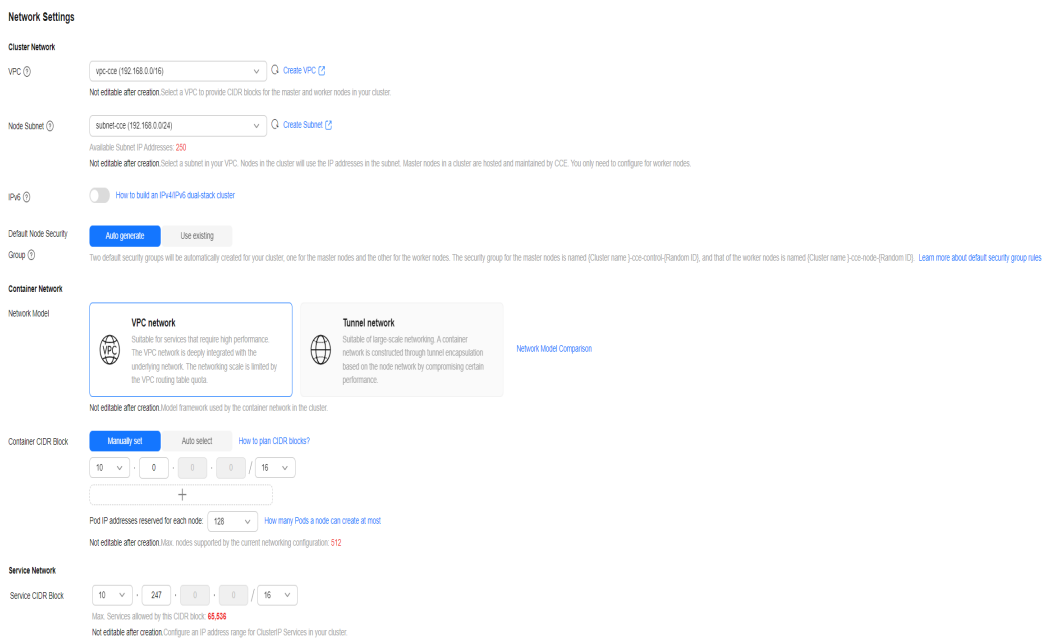


Parameter	Example	Description
Type	CCE Standard Cluster	<p>CCE allows you to create various types of clusters for diverse needs. It provides highly reliable, secure, business-class container services.</p> <p>You can select <b>CCE Standard Cluster</b> or <b>CCE Turbo Cluster</b> as required.</p> <ul style="list-style-type: none"> <li>• CCE standard clusters provide highly reliable, secure, business-class containers.</li> <li>• 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.</li> </ul> <p>For details about cluster types, see <a href="#">Comparison Between Cluster Types</a>.</p>

Parameter	Example	Description
Billing Mode	Pay-per-use	<p>Select a billing mode for the cluster.</p> <ul style="list-style-type: none"> <li>• <b>Yearly/Monthly:</b> a prepaid billing mode. Resources will be billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. If you choose this billing mode, you will need to set the desired duration and decide whether to enable automatic subscription renewal. Monthly subscriptions renew automatically every month, while yearly subscriptions renew automatically every year.</li> <li>• <b>Pay-per-use:</b> a postpaid billing mode. It is suitable for scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.</li> </ul> <p>For details, see <a href="#">Billing Modes</a>.</p>
Cluster Name	cce-test	Name of the cluster to be created
Enterprise Project	default	<p>Enterprise projects facilitate project-level management and grouping of cloud resources and users. For more details, see <a href="#">Enterprise Management</a>.</p> <p>This parameter is displayed only for enterprise users who have enabled Enterprise Project Management.</p>
Cluster Version	<b>The recommended version, for example, v1.29</b>	Select the latest commercial release for improved stability, reliability, new functionalities. CCE offers various versions of Kubernetes software.
Cluster Scale	Nodes: 50	Configure the parameter as required. This parameter controls the maximum number of worker nodes that the cluster can manage. After the cluster is created, it can only be scaled out.

Parameter	Example	Description
Master Nodes	3 Masters	<p>Select the number of master nodes. The master nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler.</p> <ul style="list-style-type: none"> <li>● <b>3 Masters:</b> Three master nodes will be created for high cluster availability.</li> <li>● <b>Single:</b> Only one master node will be created in your cluster.</li> </ul> <p>This parameter cannot be changed after the cluster is created.</p>

### Step 3 Configure network parameters.



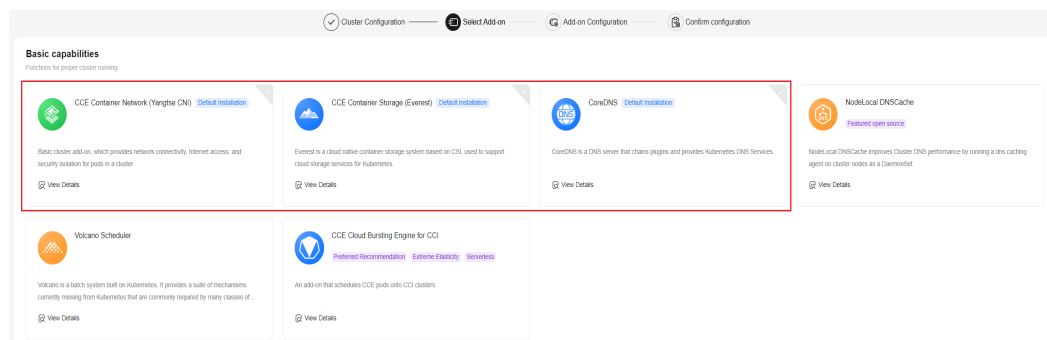
Parameter	Example	Description
VPC	vpc-cce	<p>Select a VPC for the cluster.</p> <p>If no VPC is available, click <b>Create VPC</b> to create one. After the VPC is created, click the refresh icon. For details about how to create a VPC, see <a href="#">Creating a VPC and Subnet</a>.</p>
Node Subnet	subnet-cce	<p>Select a subnet. Nodes in the cluster are assigned with the IP addresses in the subnet.</p>



Parameter	Example	Description
Network Model	VPC network	Select <b>VPC network</b> or <b>Tunnel network</b> . By default, the VPC network model is selected. For details about the differences between different container network models, see <a href="#">Container Network</a> .
Container CIDR Block	10.0.0.0/16	Configure the CIDR block used by containers. It controls how many pods can run in the cluster.
Service CIDR Block	10.247.0.0/16	Configure the ClusterIP CIDR block for the cluster. It controls how many Services can be created in the cluster and cannot be changed after configuration.

**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.



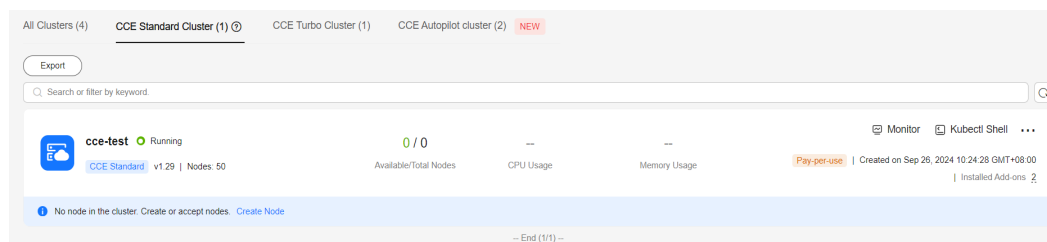
**Step 5** Click **Next: Add-on Configuration**. There is no need to set up the add-ons that are installed by default.

**Step 6** Click **Next: Confirm configuration**, confirm the resources on the page displayed, and click **Submit**.

Wait until the cluster is created. It takes about 5 to 10 minutes to create a cluster.

The created cluster will be displayed on the **Clusters** page, and there are zero nodes in it.

**Figure 1-1** Cluster created

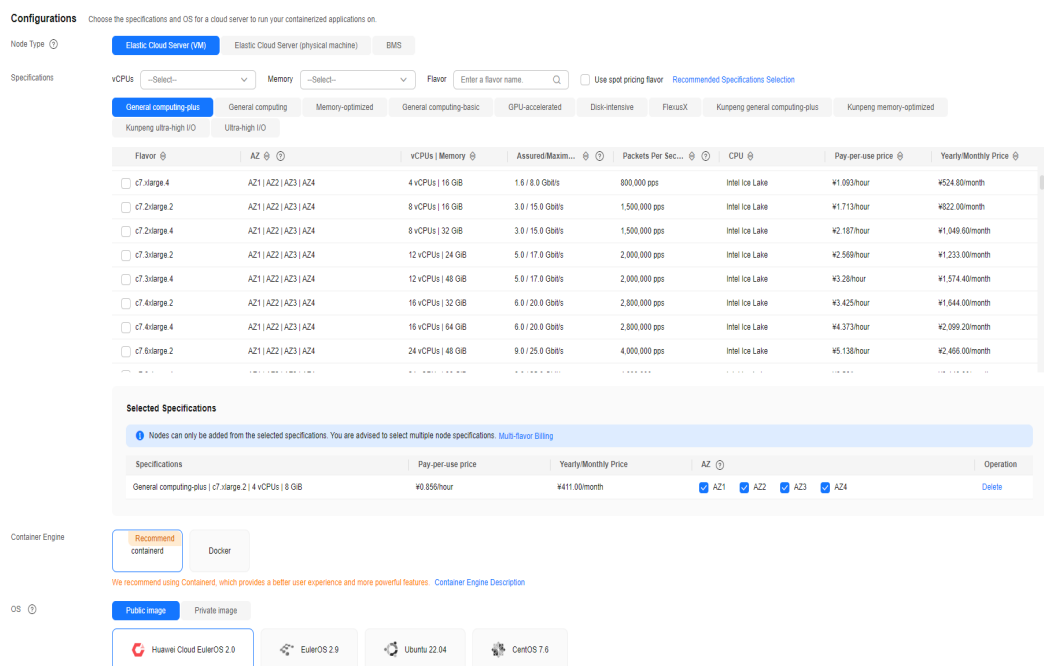


----End

### Step 3: Create a Node Pool and Nodes in the Cluster

- 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 **Nodes**. On the **Node Pools** tab, click **Create Node Pool** in the upper right corner.
- Step 3** Configure the node pool parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the configuration parameters, see [Creating a Node Pool](#).



Parameter	Example	Description
Node Type	Elastic Cloud Server (VM)	Select a node type based on service requirements. Then, the available node flavors will be automatically displayed in the <b>Specifications</b> area for you to select.
Specifications	4 vCPUs   8 GiB	Select a node flavor that best fits your service needs.  For optimal performance of the cluster components, you are advised to set up the node with a minimum of 4 vCPUs and 8 GiB of memory.

Parameter	Example	Description
Container Engine	containerd	Select a container engine based on service requirements. For details about the differences between container engines, see <a href="#">Container Engines</a> .
OS	Huawei Cloud EulerOS 2.0	Select an OS for the node.
Login Mode	<b>A custom password</b>	<ul style="list-style-type: none"> <li><b>Password:</b> Enter a password for logging in to the node and confirm the password. The default username is <b>root</b>. Keep the password secure. If you forget the password, the system is unable to retrieve it.</li> <li><b>Key Pair:</b> Select a key pair for logging to the node and select the check box to acknowledge that you have obtained the key file and without this file you will not be able to log in to the node. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click <b>Create Key Pair</b> to create one. For details, see <a href="#">Creating a Key Pair on the Management Console</a>.</li> </ul>

**Step 4** Configure parameters in **Storage Settings** and **Network Settings**. In this example, you can keep the default values for the parameters. You only need to select **I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.** and click **Next: Confirm**.

**Storage Settings** Configure storage resources for containers and applications on the node.

System Disk: General-purpose SSD (AZ3 | AZ2 | AZ1 | AZ4) 50 GB

System Disk Encryption: Not encrypted

Data Disk: General-purpose SSD (AZ3 | AZ2 | AZ1 | AZ4) 100 GB | Quantity: 1 | Default Data Disk

Used by the container runtime and kubelet. Do not uninstall this disk. Otherwise, the node will become unavailable. [How do I set data disk size?](#) [How do I allocate data disk space?](#)

Container Engine: Shared disk space | Pod: All | Write Mode: Linear | Data Disk Encryption: Not encrypted

**Add**

You can add 15 more EVS data disks.

---

**Network Settings** Configure networking resources for node and application communication.

Virtual Private Cloud: vpc-cce

Node subnet: Multiple subnet | **Single subnet** | subnet-cce (192.168.0.0/24) (Subnet) | Available Subnet IP Addresses: 248

If the single subnet IP resources associated with your node pool are tight, it is recommended that you configure multiple subnets for the node pool.

**Warning:** If the default DNS server of the subnet is modified, ensure that the custom DNS server can resolve the OBS service domain name. Otherwise, the node cannot be created.

Node IP: **Automatic**

Associate Security Group: cce-test-cce-node-r8mb7 | **Default** | X | [Create Security Group](#)

Configure security group rules for worker nodes in your cluster. The rules will take effect on all worker nodes in the cluster. For additional security group configurations, go to the Security Groups page. The security group for master nodes is automatically created by CCE. [View Default Security Group Rules](#)

Not editable after creation.

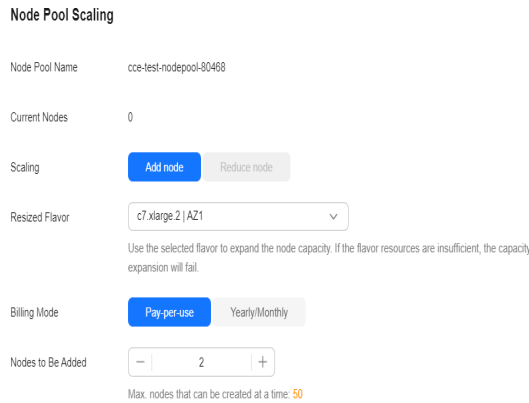
I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.

**Step 5** Check the node specifications, read the instructions on the page, and click **Submit**.

**Step 6** Locate the row containing the target node pool and click **Scaling**. There are zero nodes in the created node pool by default.



**Step 7** Set the number of nodes to be added to **2**, which means two more nodes will be created in the node pool.



**Step 8** Wait until the nodes are created. It takes about 5 to 10 minutes to complete the node creation.



----End

## Step 4: Create a Workload and Access It

You can deploy a workload using the console or [kubectl](#). This section uses an NGINX image as an example.

### Using the CCE Console

**Step 1** In the navigation pane, choose **Workloads**. Then, click **Create Workload** in the upper right corner.

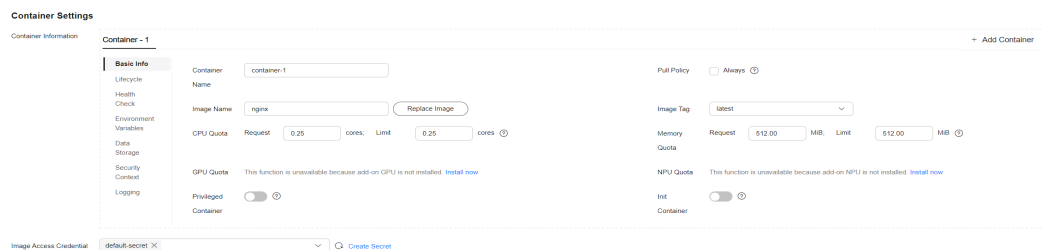
**Step 2** Configure the basic information about the workload.

In this example, configure the following parameters and keep the default values for other parameters. (For details about the configuration parameters, see [Creating a Deployment](#).)

Parameter	Example	Description
Workload Type	Deployment	In Kubernetes clusters, a workload refers to an application that is currently running. There are various built-in workloads available, each designed for different functions and application scenarios. For details about workload types, see <a href="#">Workloads</a> .
Workload Name	nginx	Enter a workload name.
Namespace	default	In a Kubernetes cluster, a namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces.  After a cluster is created, a namespace named <b>default</b> is generated by default. You can directly use the namespace.
Pods	1	Enter the number of pods.

### Step 3 Configure container parameters.

Configure the following parameters and keep the default values for other parameters.



Parameter	Example	Description
Image Name	The nginx image of the latest version	Click <b>Select Image</b> . In the displayed dialog box, click the <b>Open Source Images</b> tab and select a public image.

Parameter	Example	Description
CPU Quota	Request: 0.25 cores; Limit: 0.25 cores	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of CPUs pre-allocated to the container. The default value is 0.25 cores.</li> <li>• <b>Limit:</b> Enter the maximum number of CPUs that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>
Memory Quota	Request: 512 MiB; Limit: 512 MiB	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of memory resources pre-allocated to the container. The default value is <b>512</b> MiB.</li> <li>• <b>Limit:</b> Enter the maximum number of memory resources that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>

**Step 4** Configure access settings.

In the **Service Settings** area, click the plus sign (+) and create a Service for accessing the workload from external networks. This example shows how to create a LoadBalancer Service. You can configure the following parameters in the window that slides out from the right.

**Create Service**

Service Name:

Service Type:
 

**ClusterIP**

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

**NodePort**

Expose services via IP and static port (NodePort) on each node

**LoadBalancer**

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

**DNAT**

Expose cluster node access type services through NAT gateway, support multiple nodes to share and use elastic IP

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

Service Affinity:  Cluster-level  Node-level ?

Load Balancer:

Create a load balancing instance based on the following configurations. The automatically created instance will be automatically deleted when the current resource is deleted.  
Automatically created load balancers are pay-per-use.

Instance Name:

Enterprise Project:  Create Enterprise Project ?

AZ:  × ?

Frontend Subnet:  View Subnet ?

Backend Subnet:  ?

Network Specifications:  Elastic  Fixed ?

A single AZ instance supports a maximum of 400,000 new connections, 20,000,000 concurrent connections, and 10,000,000 Mbit/s bandwidth. The instance performance increases with the number of AZs.

EIP:   Auto create ?

Line:  Dynamic BGP  Static BGP

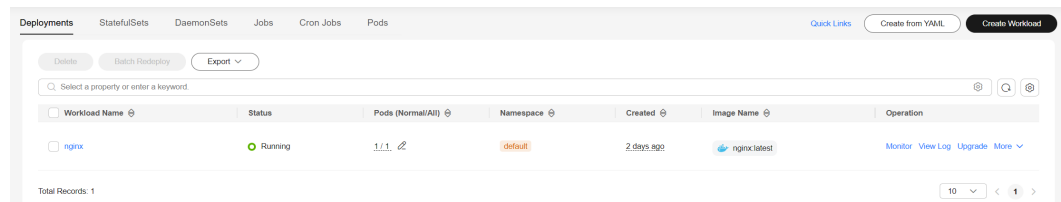
Parameter	Example	Description
Service Name	nginx	Enter a Service name.
Service Type	LoadBalancer	Select a Service type, which refers to the Service access mode. For details about the differences between Service types, see <a href="#">Service</a> .
Load Balancer	<ul style="list-style-type: none"> <li>Dedicated</li> <li>AZ: at least one AZ, for example, <b>AZ1</b></li> <li>EIP: Auto create</li> </ul> <p>Keep the default values for other parameters.</p>	<p>Select <b>Use existing</b> if there is one.</p> <p>If no load balancer is available, select <b>Auto create</b> to create one and bind an EIP to it. For details about the parameters, see <a href="#">Creating a LoadBalancer Service</a>.</p>

Parameter	Example	Description
Ports	<ul style="list-style-type: none"> <li>• Protocol: TCP</li> <li>• Container Port: 80</li> <li>• Service Port: 8080</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Protocol:</b> Select a protocol for the load balancer listener.</li> <li>• <b>Container Port:</b> Enter the listening port of the containerized application. The value must be the same as the listening port provided by the application for external systems. If the <b>nginx</b> image is used, set this parameter to <b>80</b>.</li> <li>• <b>Service Port:</b> Enter a custom port. Load balancer will use this port to create a listener and provide an entry for external traffic. You can customize the port for external access.</li> </ul>

**Step 5** Click **Create Workload**.

Wait until the workload is created. The created workload will be displayed on the **Deployments** tab.

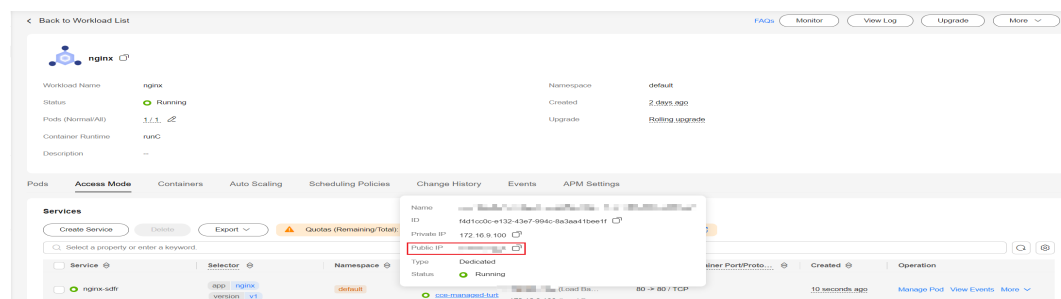
**Figure 1-2** Workload created



**Step 6** Obtain the external access address of Nginx.

Click the name of the **nginx** workload to go to its details page. On the page displayed, click the **Access Mode** tab, view the IP address of nginx. The public IP address is the external access address.

**Figure 1-3** Obtaining the external access address



**Step 7** In the address box of a browser, enter *{External access address:Service port}* to access the workload. The value of *{Service port}* is the same as the service port specified in **Step 4**. In this example, the value is **8080**.



Figure 1-4 Accessing nginx



----End

## Using kubectl

### NOTICE

If you use kubectl to access the cluster, prepare an ECS **that has been bound with an EIP in the same VPC as the cluster.**

**Step 1** Log in to the target ECS. For details, see [Logging In to a Linux ECS](#).

**Step 2** 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.

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

1. Download kubectl.

```
cd /home  
curl -LO https://dl.k8s.io/release/{v1.29.0}/bin/linux/amd64/kubectl
```

*{v1.29.0}* specifies the version. You can replace it as required.

2. Install kubectl.

```
chmod +x kubectl  
mv -f kubectl /usr/local/bin
```

**Step 3** Configure a credential for kubectl to access the Kubernetes cluster.

1. Log in to the [CCE console](#) and click the cluster name to access the cluster console. Choose **Overview** in the navigation pane.
2. On the cluster overview page, locate the **Connection Info** 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 corresponding configuration file.
4. Log in to the VM where the kubectl client has been installed and copy and paste the configuration file (for example, **kubeconfig.yaml**) downloaded in the previous step 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 see whether the cluster can be accessed.

For example, to view the cluster information, run the following command:

```
kubectl cluster-info
```

Information similar to the following is displayed:

```
Kubernetes master is running at https://*:*:5443
CoreDNS is running at https://*:*:5443/api/v1/namespaces/kube-system/services/coredns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- Step 4** Create a YAML file named `nginx-deployment.yaml`. `nginx-deployment.yaml` is an example file name. You can rename it as required.

```
vi nginx-deployment.yaml
```

The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

- Step 5** Run the following command to deploy the workload:

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

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

```
deployment "nginx" created
```

- Step 6** Run the following command to check the workload status:

```
kubectl get deployment
```

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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	1/1	1	1	4m5s

The parameters in the command output are described as follows:

- **NAME:** specifies the name of a workload.
- **READY:** indicates the number of available pods/expected pods for the workload.
- **UP-TO-DATE:** specifies the number of pods that have been updated for the workload.
- **AVAILABLE:** specifies the number of pods available for the workload.
- **AGE:** specifies how long the workload has run.

- Step 7** Create a YAML file named `nginx-elb-svc.yaml` and change the value of `selector` to that of `matchLabels` (`app: nginx` in this example) in the `nginx-deployment.yaml` file to associate the Service with the backend application.

```
vi nginx-elb-svc.yaml
```

For details about the parameters in the following example, see [Using kubectl to Create a Service \(Automatically Creating a Load Balancer\)](#).

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate:
      {
        "type": "public",
        "bandwidth_name": "cce-bandwidth",
        "bandwidth_chargemode": "bandwidth",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp"
      }
  labels:
    app: nginx
  name: nginx
spec:
  ports:
    - name: service0
      port: 8080
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

- Step 8** Run the following command to create the 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 9** Run the following command to check the Service:

```
kubectl get svc
```

If information similar to the following is displayed, the access type has been configured, and the workload is accessible:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.247.0.1	<none>	443/TCP	3d
nginx	LoadBalancer	10.247.130.196	**.*.*.*	8080:31540/TCP	51s

- Step 10** Enter the URL (for example, `**.*.*.*:8080`) in the address box of a browser. `**.*.*.*` specifies the EIP of the load balancer, and `8080` indicates the access port.

**Figure 1-5** Accessing nginx using the LoadBalancer Service

# 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](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

----End

## Follow-up Operations: Releasing Resources

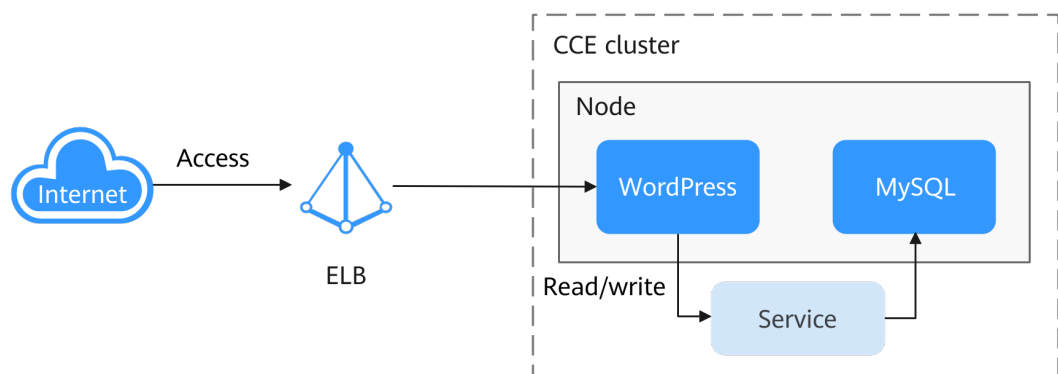
To avoid additional charges, make sure to release resources promptly if you no longer require the cluster. For details, see [Deleting a Cluster](#).

# 2 Deploying a WordPress StatefulSet in a CCE Cluster

StatefulSets are a specific type of workload in Kubernetes that are designed to manage stateful applications. Unlike Deployments, StatefulSets are ideal for applications that require data consistency and durability. Each application instance has its own unique identifier and must be deployed and scaled in a specific sequence. Examples of stateful applications include databases (like MySQL) and message queues (such as Kafka). This section uses the WordPress blogging platform and a MySQL database as an example to describe how to deploy a StatefulSet in a CCE cluster.

WordPress started as a blogging platform using PHP and MySQL, but it has evolved into a complete content management system. You can use a CCE cluster to quickly set up your own blog. For more information about WordPress, see the [WordPress official website](#).

WordPress and a database (a MySQL database in this example) are often used together, with WordPress managing content and the database storing website data. In a containerized deployment, WordPress and MySQL typically run in separate containers. WordPress accesses MySQL through a Service.



## Procedure

Step	Description
<a href="#">Preparations</a>	Register a Huawei account and top up the account.

Step	Description
<a href="#">Step 1: Enable CCE for the First Time and Perform Authorization</a>	Obtain the required permissions for your account when you use the CCE service in the current region for the first time.
<a href="#">Step 2: Create a Cluster</a>	Create a CCE cluster to provide Kubernetes services.
<a href="#">Step 3: Create a Node Pool and Nodes in the Cluster</a>	Create a node in the cluster to run your containerized applications.
<a href="#">Step 4: Deploy MySQL</a>	Create a MySQL workload in the cluster and create a ClusterIP Service for WordPress access.
<a href="#">Step 5: Deploy WordPress</a>	Create a WordPress workload in the cluster and create a LoadBalancer Service for the workload for Internet access.
<a href="#">Step 6: Access WordPress</a>	Access the WordPress website from the Internet to start your blog.
<a href="#">Follow-up Operations: Releasing Resources</a>	To avoid additional charges, delete the cluster resources promptly if you no longer require them after practice.


## Preparations

- Before starting, register a Huawei account 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

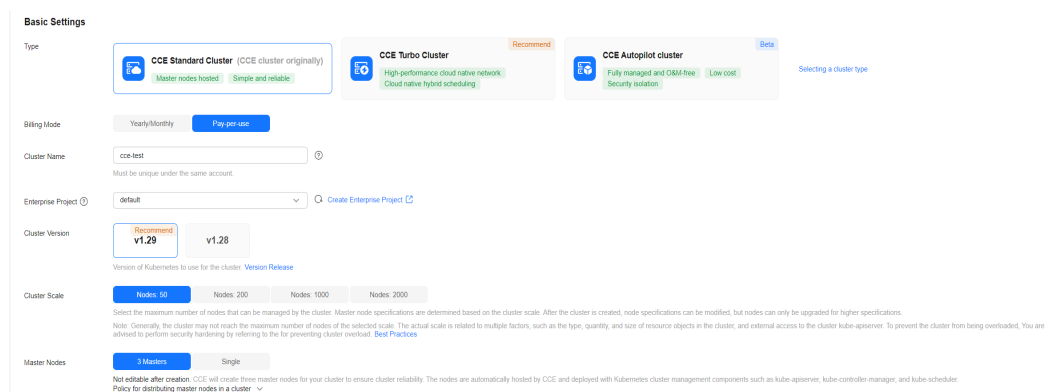
## Step 2: Create a Cluster

**Step 1** Log in to the [CCE console](#).

- If you have no clusters, click **Buy Cluster** on the wizard page.
- If you have CCE clusters, choose **Clusters** in the navigation pane, click **Buy Cluster** in the upper right corner.

**Step 2** Configure basic cluster parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the parameter configurations, see [Buying a CCE Standard/Turbo Cluster](#).



The screenshot shows the 'Basic Settings' page for creating a CCE cluster. The page is divided into several sections:

- Type:** Three options are available: **CCE Standard Cluster** (CCE cluster originally), **CCE Turbo Cluster** (Recommended), and **CCE Autopilot cluster** (Beta). The Standard Cluster is selected.
- Billing Mode:** Two options: **Yearly/Monthly** and **Pay per use**. 'Pay per use' is selected.
- Cluster Name:** A text input field containing 'cce-test'.
- Enterprise Project:** A dropdown menu showing 'default' and a link to 'Create Enterprise Project'.
- Cluster Version:** Two options: **v1.29** (Recommended) and **v1.28**.
- Cluster Scale:** Four options: **Nodes 50** (Selected), **Nodes 200**, **Nodes 1000**, and **Nodes 2000**.
- Master Nodes:** Two options: **3 Masters** (Selected) and **Single**.

Below the 'Cluster Scale' section, there is a note: 'Select the maximum number of nodes that can be managed by the cluster. Master node specifications are determined based on the cluster scale. After the cluster is created, node specifications can be modified, but nodes can only be upgraded for higher specifications. Note: Generally, the cluster may reach the maximum number of nodes of the selected scale. The actual scale is related to multiple factors, such as the type, quantity, and size of resource objects in the cluster, and external access to the cluster kube-apiserver. To prevent the cluster from being overloaded, you are advised to perform security hardening by referring to the for preventing cluster overload. [Best Practices](#)'

Below the 'Master Nodes' section, there is a note: 'Not editable after creation. CCE will create three master nodes for your cluster to ensure cluster reliability. The nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller manager, and kube-scheduler. [Policy for distributing master nodes in a cluster](#)'

Parameter	Example	Description
Type	CCE Standard Cluster	<p>CCE allows you to create various types of clusters for diverse needs. It provides highly reliable, secure, business-class container services.</p> <p>You can select <b>CCE Standard Cluster</b> or <b>CCE Turbo Cluster</b> as required.</p> <ul style="list-style-type: none"> <li>• CCE standard clusters provide highly reliable, secure, business-class containers.</li> <li>• 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.</li> </ul> <p>For details about cluster types, see <a href="#">Comparison Between Cluster Types</a>.</p>
Billing Mode	Pay-per-use	<p>Select a billing mode for the cluster.</p> <ul style="list-style-type: none"> <li>• <b>Yearly/Monthly</b>: a prepaid billing mode. Resources will be billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. If you choose this billing mode, you will need to set the desired duration and decide whether to enable automatic subscription renewal. Monthly subscriptions renew automatically every month, while yearly subscriptions renew automatically every year.</li> <li>• <b>Pay-per-use</b>: a postpaid billing mode. It is suitable for scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.</li> </ul> <p>For details, see <a href="#">Billing Modes</a>.</p>
Cluster Name	cce-test	Name of the cluster to be created
Enterprise Project	default	<p>Enterprise projects facilitate project-level management and grouping of cloud resources and users. For more details, see <a href="#">Enterprise Management</a>.</p> <p>This parameter is displayed only for enterprise users who have enabled Enterprise Project Management.</p>



Parameter	Example	Description
Cluster Version	<b>The recommended version, for example, v1.29</b>	Select the latest commercial release for improved stability, reliability, new functionalities. CCE offers various versions of Kubernetes software.
Cluster Scale	Nodes: 50	Configure the parameter as required. This parameter controls the maximum number of worker nodes that the cluster can manage. After the cluster is created, it can only be scaled out.
Master Nodes	3 Masters	<p>Select the number of master nodes. The master nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler.</p> <ul style="list-style-type: none"> <li>● <b>3 Masters:</b> Three master nodes will be created for high cluster availability.</li> <li>● <b>Single:</b> Only one master node will be created in your cluster.</li> </ul> <p>This parameter cannot be changed after the cluster is created.</p>

### Step 3 Configure network parameters.

**Network Settings**

**Cluster Network**

VPC  [Create VPC](#)

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

Node Subnet  [Create Subnet](#)

Available Subnet IP Addresses: 250

Not editable after creation Select a subnet in your VPC. Nodes in the cluster will use the IP addresses in the subnet. Master nodes in a cluster are hosted and maintained by CCE. You only need to configure for worker nodes.

IPv6  [How to build an IPv4/IPv6 dual-stack cluster](#)

**Default Node Security Group**

Group

Two default security groups will be automatically created for your cluster, one for the master nodes and the other for the worker nodes. The security group for the master nodes is named [Cluster name]-cos-control-[Random ID], and that of the worker nodes is named [Cluster name]-cos-nodes-[Random ID]. [Learn more about default security group rules.](#)

**Container Network**

Network Model

**VPC network**

Suitable for services that require high performance. The VPC network is deeply integrated with the underlying network. The networking scale is limited by the VPC routing table quota.

**Tunnel network**

Suitable of large-scale networking. A container network is constructed through tunnel encapsulation based on the node network by compromising certain performance.

Not editable after creation Model framework used by the container network in the cluster. [Network Model Comparison](#)

**Container CIDR Block**

Manually set  Auto select [How to plan CIDR blocks?](#)

/

Pod IP addresses reserved for each node: 128 [How many Pods a node can create at most](#)

Not editable after creation Max. nodes supported by the current networking configuration: 512

**Service Network**

Service CIDR Block     /

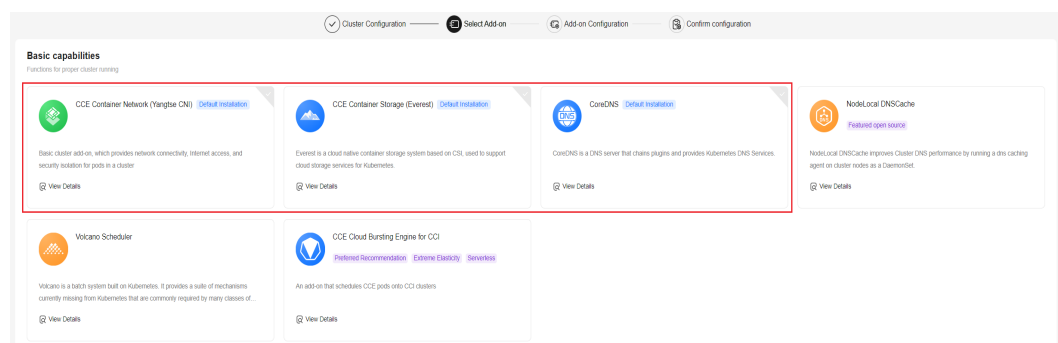
Max. Services allowed by this CIDR block: 65,536

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

Parameter	Example	Description
VPC	vpc-cce	Select a VPC for the cluster. If no VPC is available, click <b>Create VPC</b> to create one. After the VPC is created, click the refresh icon. For details about how to create a VPC, see <a href="#">Creating a VPC and Subnet</a> .
Node Subnet	subnet-cce	Select a subnet. Nodes in the cluster are assigned with the IP addresses in the subnet.
Network Model	VPC network	Select <b>VPC network</b> or <b>Tunnel network</b> . By default, the VPC network model is selected. For details about the differences between different container network models, see <a href="#">Container Network</a> .
Container CIDR Block	10.0.0.0/16	Configure the CIDR block used by containers. It controls how many pods can run in the cluster.
Service CIDR Block	10.247.0.0/16	Configure the ClusterIP CIDR block for the cluster. It controls how many Services can be created in the cluster and cannot be changed after configuration.

**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.



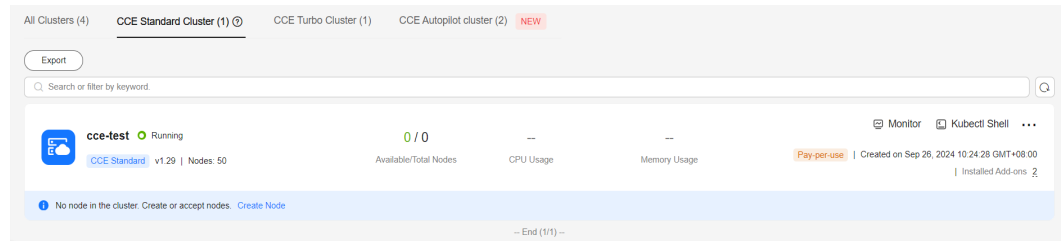
**Step 5** Click **Next: Add-on Configuration**. There is no need to set up the add-ons that are installed by default.

**Step 6** Click **Next: Confirm configuration**, confirm the resources on the page displayed, and click **Submit**.

Wait until the cluster is created. It takes about 5 to 10 minutes to create a cluster.

The created cluster will be displayed on the **Clusters** page, and there are zero nodes in it.

Figure 2-1 Cluster created

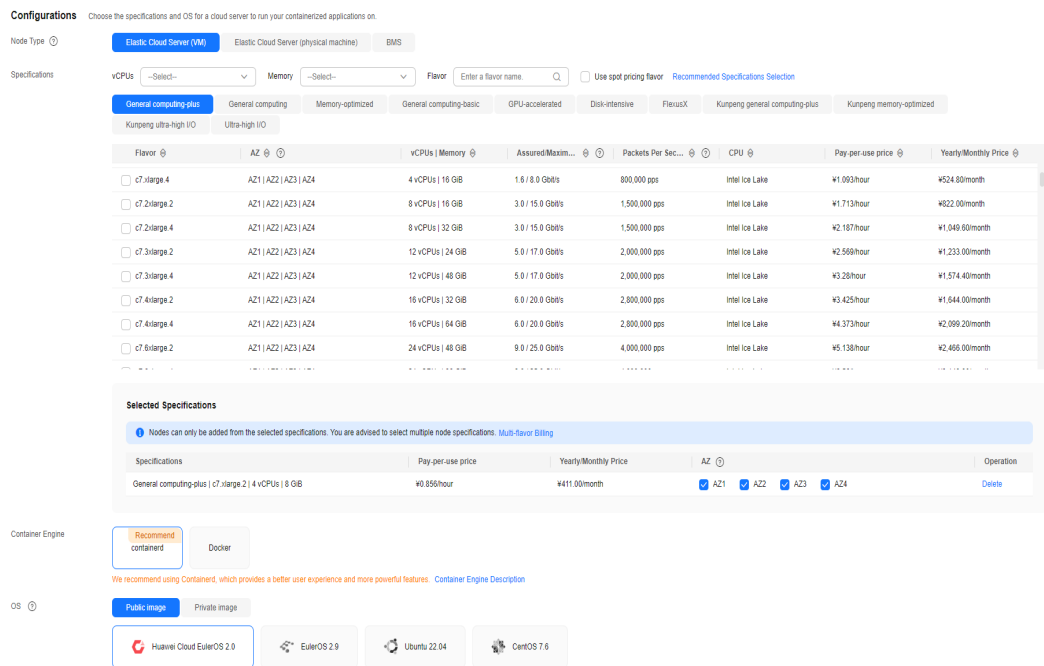


----End

### Step 3: Create a Node Pool and Nodes in the Cluster

- 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 **Nodes**. On the **Node Pools** tab, click **Create Node Pool** in the upper right corner.
- Step 3** Configure the node pool parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the configuration parameters, see [Creating a Node Pool](#).



Parameter	Example	Description
Node Type	Elastic Cloud Server (VM)	Select a node type based on service requirements. Then, the available node flavors will be automatically displayed in the <b>Specifications</b> area for you to select.

Parameter	Example	Description
Specifications	4 vCPUs   8 GiB	Select a node flavor that best fits your service needs.  For optimal performance of the cluster components, you are advised to set up the node with a minimum of 4 vCPUs and 8 GiB of memory.
Container Engine	containerd	Select a container engine based on service requirements. For details about the differences between container engines, see <a href="#">Container Engines</a> .
OS	Huawei Cloud EulerOS 2.0	Select an OS for the node.
Login Mode	<b>A custom password</b>	<ul style="list-style-type: none"> <li>• <b>Password:</b> Enter a password for logging in to the node and confirm the password. The default username is <b>root</b>. Keep the password secure. If you forget the password, the system is unable to retrieve it.</li> <li>• <b>Key Pair:</b> Select a key pair for logging to the node and select the check box to acknowledge that you have obtained the key file and without this file you will not be able to log in to the node. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click <b>Create Key Pair</b> to create one. For details, see <a href="#">Creating a Key Pair on the Management Console</a>.</li> </ul>

**Step 4** Configure parameters in **Storage Settings** and **Network Settings**. In this example, you can keep the default values for the parameters. You only need to select **I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.** and click **Next: Confirm**.

**Storage Settings** Configure storage resources for containers and applications on the node.

System Disk: General-purpose SSD (AZ3 | AZ2 | AZ1 | AZ4) 50 GB

Expand System Disk Encryption: Not encrypted

Data Disk: General-purpose SSD (AZ3 | AZ2 | AZ1 | AZ4) 100 GB Quantity: 1

Used by the container runtime and kubelet. Do not uninstall this disk. Otherwise, the node will become unavailable. [How do I set data disk size?](#) [How do I allocate data disk space?](#)

Expand Container Engine: Shared disk space | Pod: All | Write Mode: Linear | Data Disk Encryption: Not encrypted

[Add](#)

You can add 15 more EVS data disks.

---

**Network Settings** Configure networking resources for node and application communication.

Virtual Private Cloud: vpc-cce

Node subnet: Multiple subnet | Single subnet | subnet-cce (192.168.0.0/24) (Subnet) Available Subnet IP Addresses: 248

If the single subnet IP resources associated with your node pool are tight, it is recommended that you configure multiple subnets for the node pool.

**Warning:** If the default DNS server of the subnet is modified, ensure that the custom DNS server can resolve the OBS service domain name. Otherwise, the node cannot be created.

Node IP: Automatic

Associate Security Group: cce-test-cce-node-r6mb7 (Default) X [Create Security Group](#)

Configure security group rules for worker nodes in your cluster. The rules will take effect on all worker nodes in the cluster. For additional security group configurations, go to the Security Groups page. The security group for master nodes is automatically created by CCE. [View Default Security Group Rules](#)

Not editable after creation.

I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.

**Step 5** Check the node specifications, read the instructions on the page, and click **Submit**.

**Step 6** Locate the row containing the target node pool and click **Scaling**. There are zero nodes in the created node pool by default.

Node Pools Nodes Quick Links Create Node Pool

Export View Events Operation Records Quota (remaining/total): Cluster nodes (50/50) ECS (999/1,000) CPU(cores) (7,996/8,000)

Search or filter by keyword

▼ cce-test-nodepool-80468 Normal View Node Update Scaling Auto Scaling More

Node Type: Elastic Cloud Server (VM) Enterprise Project: default Total number of nodes (actual/expected): 0/0 CPU Usage/Request: --/-- Memory Usage/Request: --/--

**Step 7** Set the number of nodes to be added to 2, which means two more nodes will be created in the node pool.

**Node Pool Scaling**

Node Pool Name: cce-test-nodepool-80468

Current Nodes: 0

Scaling: Add node Reduce node

Resized Flavor: c7.xlarge.2 | AZ1

Use the selected flavor to expand the node capacity. If the flavor resources are insufficient, the capacity expansion will fail.

Billing Mode: Pay-per-use Yearly/Monthly

Nodes to Be Added: 2

Max. nodes that can be created at a time: 50

**Step 8** Wait until the nodes are created. It takes about 5 to 10 minutes to complete the node creation.

^ cce-test-nodepool-80468 Normal View Node Update Scaling Auto Scaling More

Node Type: Elastic Cloud Server (VM) Enterprise Project: default Total number of nodes (actual/expected): 2/2 CPU Usage/Request: --/-- Memory Usage/Request: --/--

Specifications	AZ	Status	Actual/Desired Nodes	Number of yearly/month...	On-Demand Nodes	Auto Scaling	Operation
c7.xlarge.2   4 vCPUs   8 GB	AZ1	Normal	2/2	0	0	Close	<a href="#">View Node</a> <a href="#">Scaling</a>

----End

## Step 4: Deploy MySQL

You can deploy a MySQL workload in different ways.

### Using the CCE Console

- Step 1** Log in to the [CCE console](#).
- Step 2** Click the name of the target cluster to access the cluster console.
- Step 3** In the navigation pane, choose **Workloads**. Then, click **Create Workload** in the upper right corner.
- Step 4** Configure the basic information about the workload.

In this example, configure the following parameters and keep the default values for other parameters. For details about the configuration parameters, see [Creating a StatefulSet](#).

**Basic Info**

Workload Type

Deployment

StatefulSet

DaemonSet

Job

Cron Job

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

Workload Name

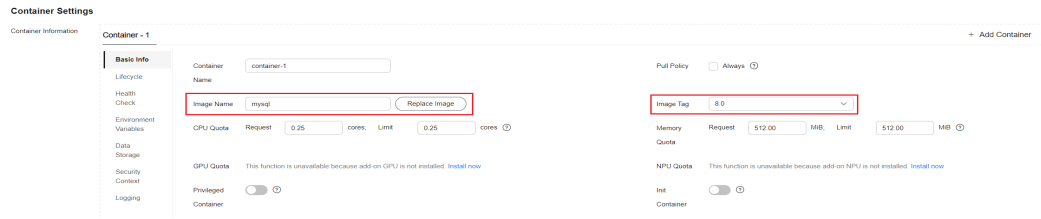
Namespace  [Create Namespace](#)

Pods

Parameter	Example	Description
Workload Type	StatefulSet	In Kubernetes clusters, a workload refers to an application that is currently running. There are various built-in workloads available, each designed for different functions and application scenarios. For details about workload types, see <a href="#">Workloads</a> .
Workload Name	mysql	Enter a workload name.
Namespace	default	In a Kubernetes cluster, a namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces.  After a cluster is created, a namespace named <b>default</b> is generated by default. You can directly use the namespace.

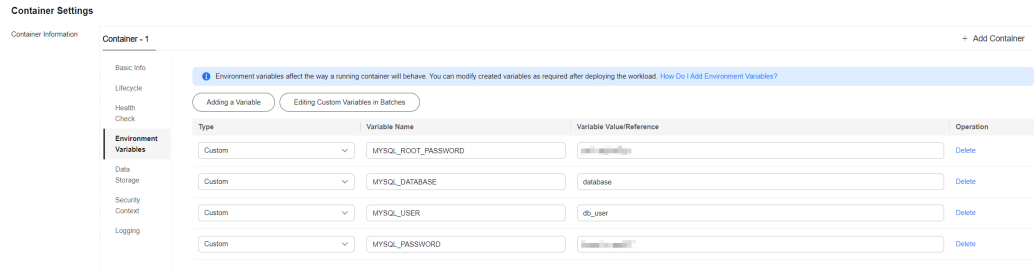
Parameter	Example	Description
Pods	1	Enter the number of pods.

**Step 5** Configure the basic information about the container.



Parameter	Example	Description
Image Name	A mysql image of version 8.0	In the <b>Container Settings</b> area, click <b>Basic Info</b> and click <b>Select Image</b> . In the dialog box displayed, select <b>Open Source Images</b> , search for <b>mysql</b> , select the <b>mysql</b> image, and select <b>8.0</b> from the drop-down list for <b>Image Tag</b> .
CPU Quota	Request: 0.25 cores; Limit: 0.25 cores	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of CPUs pre-allocated to the container. The default value is 0.25 cores.</li> <li>• <b>Limit:</b> Enter the maximum number of CPUs that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>
Memory Quota	Request: 512 MiB; Limit: 512 MiB	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of memory resources pre-allocated to the container. The default value is <b>512</b> MiB.</li> <li>• <b>Limit:</b> Enter the maximum number of memory resources that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>

**Step 6** Click **Environment Variables** and add four environment variables. For details about the environment variables supported by MySQL, see [MySQL](#).



Environment Variable	Example	Description
MYSQL_ROOT_PASSWORD	<b>A custom password</b>	Password of the <b>root</b> user of the MySQL database, which can be customized
MYSQL_DATABASE	database	Name of the database to be created when the image is started, which can be customized
MYSQL_USER	db_user	Database username, which can be customized
MYSQL_PASSWORD	<b>A custom password</b>	Database user password, which can be customized

**Step 7** Click **Data Storage**, click **Add Volume**, select **VolumeClaimTemplate (VTC)** from the drop-down list, and add an EVS disk for MySQL.

Click **Create PVC** and configure the following parameters: (Keep the default values for other parameters.)



✕

### Create PVC

PVC Type: EVS Local PV DSS

PVC Name:

Creation Method: Dynamically provision ⓘ  
Creating underlying storage incurs fees.

Storage Classes:  Q

Storage Volume Name Prefix:   
If this parameter is left blank, the default value is "pvc". The actual storage volume name is the combination of the storage volume name prefix and PVC UID.

AZ: AZ1 (2)  
The current AZ has 2 nodes. EVS disks can be attached only to nodes in the same AZ. After an EVS disk is created, AZs cannot be changed. Exercise caution when selecting an EVS disk.

Disk Type: High I/O General-purpose SSD Ultra-high I/O Extreme SSD  
General Purpose SSD V2 High I/O(ext) General-purpose SSD(ext)  
Ultra-high I/O(ext) Extreme SSD(ext)

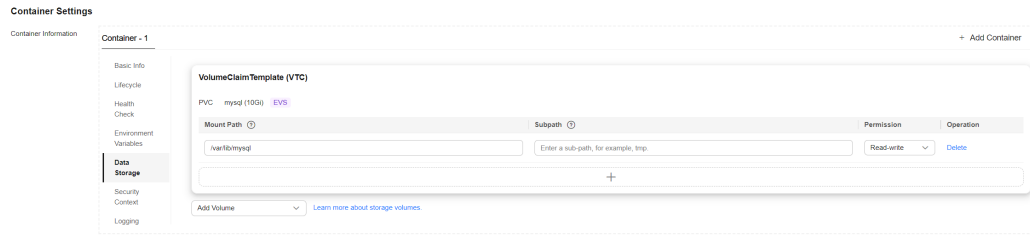
Capacity (GiB):  + -

Billing Mode: Pay-per-use Yearly/Monthly

Access Mode: ReadWriteOnce ⓘ

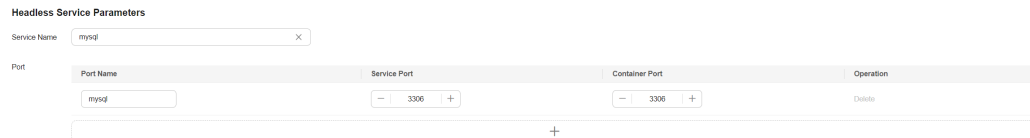
Parameter	Example	Description
PVC Type	EVS	Select a type for the underlying storage volume used by the PVC.
PVC Name	mysql	Enter a custom PVC name, for example, <b>mysql</b> .
Storage Classes	csi-disk	The default value is <b>csi-disk</b> .
AZ	AZ1	Select an AZ. The EVS disk can only be attached to nodes in the same AZ. After an EVS disk is created, the AZ where the disk locates cannot be changed.
Disk Type	General-purpose SSD	Select a proper type as required.
Capacity (GiB)	10 GiB	Enter the capacity as required. The default value is <b>10</b> GiB.

Click **Create** and enter the path for mounting the storage volume to the container. The default path used by MySQL is **`/var/lib/mysql`**.



**Step 8** In the **Headless Service Parameters** area, configure a headless Service.

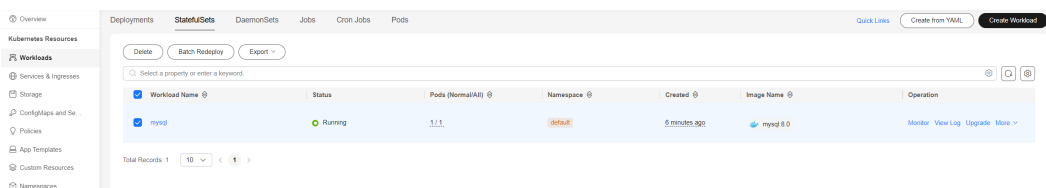
The headless Service is used for networking between StatefulSet pods. It generates a domain name for each pod for accessing a specific StatefulSet pod. For a MySQL database that has master/slave relationship and multiple replicas, a headless Service is needed to read and write data from and into the MySQL database server (known as a source) and copy the data to other replicas. In this example, MySQL is deployed in one pod. For details about how to deploy MySQL in multiple pods, see [Run a Replicated Stateful Application](#).



Parameter	Example	Description
Service Name	mysql	Enter a custom headless Service name.
Port Name	mysql	Enter a custom port name, which is used to distinguish different ports in the same Service. In this example, only one port is used.
Service Port	3306	Enter a custom port number. This port is used by the Service for external access. In this example, the port is the same as the container port.
Container Port	3306	The actual listening port of the application in the container. It is determined by the port opened by the application image. For example, the MySQL database open port is 3306.

**Step 9** Click **Create Workload**.

Wait until the workload is created. After it is created, it will be displayed on the **StatefulSets** tab.



----End

## Using kubectl

### NOTICE

You need to create an ECS **bound with an EIP in the same VPC as the cluster** first.

#### 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.

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

1. Download kubectl.

```
cd /home
curl -LO https://dl.k8s.io/release/{v1.29.0}/bin/linux/amd64/kubectl
{v1.29.0} specifies the version. You can replace it as required.
```

2. Install kubectl.

```
chmod +x kubectl
mv -f kubectl /usr/local/bin
```

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

1. Log in to the [CCE console](#) and click the cluster name to access the cluster console. Choose **Overview** in the navigation pane.
2. On the cluster overview page, locate the **Connection Info** 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 corresponding configuration file.
4. Log in to the VM where the kubectl client has been installed and copy and paste the configuration file (for example, **kubeconfig.yaml**) downloaded in the previous step 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 see whether the cluster can be accessed.

For example, to view the cluster information, run the following command:

```
kubectl cluster-info
```

Information similar to the following is displayed:

```
Kubernetes master is running at https://*:*:5443
CoreDNS is running at https://*:*:5443/api/v1/namespaces/kube-system/services/coresdns/dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

#### Step 3 Create a description file named **mysql.yaml**. **mysql.yaml** is an example file name. You can rename it as required.

```
vi mysql.yaml
```

The file content is as follows:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
      version: v1
  template:
    metadata:
      labels:
        app: mysql
        version: v1
    spec:
      containers:
        - name: container-1
          image: mysql:8.0
          env:
            - name: MYSQL_ROOT_PASSWORD # Password of the root user of MySQL, which can be
              customized
              value: *****
            - name: MYSQL_DATABASE # Name of the database to be created when the image is started,
              which can be customized
              value: database
            - name: MYSQL_USER # Database username, which can be customized
              value: db_user
            - name: MYSQL_PASSWORD # Database user password, which can be customized
              value: *****
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          volumeMounts:
            - name: mysql
              mountPath: /var/lib/mysql
      imagePullSecrets:
        - name: default-secret
      serviceName: mysql
      volumeClaimTemplates: # Dynamically attach the EVS disk to the workload.
        - apiVersion: v1
          kind: PersistentVolumeClaim
          metadata:
            name: mysql
            namespace: default
          annotations:
            everest.io/disk-volume-type: SSD # EVS disk type
          labels:
            failure-domain.beta.kubernetes.io/region: ap-southeast-1 #Region where the EVS disk is in
            failure-domain.beta.kubernetes.io/zone: #AZ where the EVS disk is in. It must be the same as the
            AZ of the node that runs the workload.
          spec:
            accessModes:
              - ReadWriteOnce # ReadWriteOnce for an EVS disk
            resources:
              requests:
                storage: 10Gi
            storageClassName: csi-disk # Storage class name. The value is csi-disk for an EVS disk.
          ---
          apiVersion: v1
```

```
kind: Service
metadata:
  name: mysql
  namespace: default
  labels:
    app: mysql
    version: v1
spec:
  selector:
    app: mysql
    version: v1
  clusterIP: None
  ports:
  - name: mysql
    protocol: TCP
    port: 3306
    targetPort: 3306
  type: ClusterIP
```

**Step 4** Create a MySQL workload.

```
kubectl apply -f mysql.yaml
```

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

```
statefulset "mysql" created
```

**Step 5** Check the workload status.

```
kubectl get statefulset
```

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

NAME	READY	AGE
mysql	1/1	4m5s

**Step 6** Check the Service.

```
kubectl get svc
```

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

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.247.0.1	<none>	443/TCP	3d
mysql	ClusterIP	None	<none>	3306/TCP	51s

----End

## Step 5: Deploy WordPress

You can deploy a WordPress workload using either of the following ways.

### Using the CCE Console

**Step 1** Log in to the [CCE console](#).

**Step 2** Click the name of the target cluster to access the cluster console.

**Step 3** In the navigation pane, choose **Workloads**. Then, click **Create Workload** in the upper right corner.

**Step 4** Configure the basic information about the workload.

In this example, configure the following parameters and keep the default values for other parameters. For details about the configuration parameters, see [Creating a StatefulSet](#).

Basic Info

Workload Type: **Deployment** | StatefulSet | DaemonSet | Job | Cron Job

Workload Name:

Namespace:  [Create Namespace](#)

Pods:

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

Parameter	Example	Description
Workload Type	Deployment	In Kubernetes clusters, a workload refers to an application that is currently running. There are various built-in workloads available, each designed for different functions and application scenarios. For details about workload types, see <a href="#">Workloads</a> .
Workload Name	wordpress	Enter a workload name.
Namespace	default	In a Kubernetes cluster, a namespace is a conceptual grouping of resources or objects. Each namespace provides isolation for data from other namespaces. After a cluster is created, a namespace named <b>default</b> is generated by default. You can directly use the namespace.
Pods	1	Enter the number of pods.

**Step 5** Configure the basic information about the container.

Container Settings

Container Information: Container - 1

Basic Info: Container Name:

Image Name:  [Replace Image](#)

Image Tag:

CPU Quota: Request:  cores, Limit:  cores

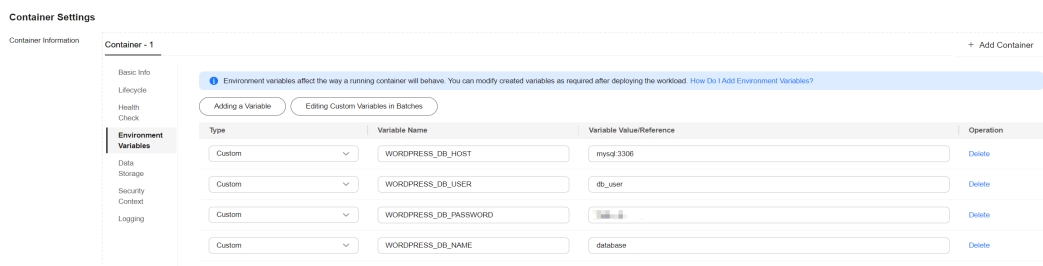
Memory Quota: Request:  MB, Limit:  MB

Privileged Container:

Parameter	Example	Description
Image Name	The wordpress image of the latest version	In the <b>Container Information</b> area, click <b>Basic Info</b> and click <b>Select Image</b> . In the dialog box displayed, select <b>Open Source Images</b> , search for <b>wordpress</b> , select the <b>wordpress</b> image, and select <b>latest</b> from the drop-down list for <b>Image Tag</b> .

Parameter	Example	Description
CPU Quota	Request: 0.25 cores; Limit: 0.25 cores	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of CPUs pre-allocated to the container. The default value is 0.25 cores.</li> <li>• <b>Limit:</b> Enter the maximum number of CPUs that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>
Memory Quota	Request: 512 MiB; Limit: 512 MiB	<ul style="list-style-type: none"> <li>• <b>Request:</b> Enter the number of memory resources pre-allocated to the container. The default value is <b>512</b> MiB.</li> <li>• <b>Limit:</b> Enter the maximum number of memory resources that can be used by the container. The default value is the same as that of the resource request. If the resource limit is greater than the resource request, it indicates that the pre-allocated resource limit can be temporarily exceeded in burst scenarios.</li> </ul> <p>For details, see <a href="#">Configuring Container Specifications</a>.</p>

**Step 6** Click **Environment Variables** and add environment variables listed in the table to add the MySQL database information to WordPress.



Environment Variable	Example	Description
WORDPRESS_DB_HOST	mysql:3306	IP address for accessing the database. In this example, you need to enter the access mode of the MySQL workload, that is, the headless Service in <a href="#">Step 4: Deploy MySQL</a> . You can use the internal domain name <b>mysql.default.svc.cluster.local:3306</b> of the cluster to access the workload. You can omit <b>.default.svc.cluster.local</b> and simply use <b>mysql:3306</b> .
WORDPRESS_DB_USER	db_user	Username for accessing data. The value must be the same as that of <b>MYSQL_USER</b> in <a href="#">Step 4: Deploy MySQL</a> . This username is used to establish a connection with the MySQL database.
WORDPRESS_DB_PASSWORD	<b>A custom database password</b>	Password for accessing the database. The value must be the same as that of <b>MYSQL_PASSWORD</b> in <a href="#">Step 4: Deploy MySQL</a> .
WORDPRESS_DB_NAME	database	Name of the database to be accessed. The value must be the same as that of <b>MYSQL_DATABASE</b> in <a href="#">Step 4: Deploy MySQL</a> .

**Step 7** Click **Data Storage**, click **Add Volume**, select **PVC**, and add an EVS disk as the MySQL storage.

Click **Create PVC** and configure the following parameters: (Keep the default values for other parameters.)



**Create PVC** [Create from YAML](#)

PVC Type:  EVS  SFS  OBS  SFS Turbo  
 Local PV  DSS

PVC Name:

Namespace:

Creation Method:  Dynamically provision  Use existing  Create new   
Creating underlying storage incurs fees.

Storage Classes:

Storage Volume Name Prefix:   
If this parameter is left blank, the default value is "pvc". The actual storage volume name is the combination of the storage volume name prefix and PVC UID.

AZ:  AZ1 (2)  AZ2 (0)  AZ3 (0)  AZ4 (0)  
The current AZ has 2 nodes. EVS disks can be attached only to nodes in the same AZ. After an EVS disk is created, AZs cannot be changed. Exercise caution when selecting an EVS disk.

Disk Type:  High I/O  General-purpose SSD  Ultra-high I/O  Extreme SSD  
 General Purpose SSD V2  High I/O(ext)  General-purpose SSD(ext)  
 Ultra-high I/O(ext)  Extreme SSD(ext)

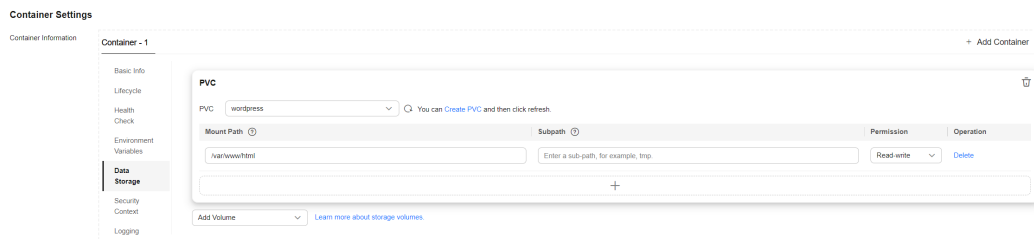
Capacity (GiB):

Billing Mode:  Pay-per-use  Yearly/Monthly

Parameter	Example	Description
PVC Type	EVS	Select a type for the underlying storage volume used by the PVC.
PVC Name	wordpress	Enter a custom PVC name.
Creation Method	Dynamically provision	In this example, select <b>Dynamically provision</b> . The PVC, PV, and underlying storage volume will be automatically created. This method is ideal when no underlying storage volume is available.
Storage Classes	csi-disk	The default value is <b>csi-disk</b> .
AZ	AZ1	Select an AZ. The EVS disk can only be attached to nodes in the same AZ. After an EVS disk is created, the AZ where the disk locates cannot be changed.
Disk Type	General-purpose SSD	Select a proper type as required.

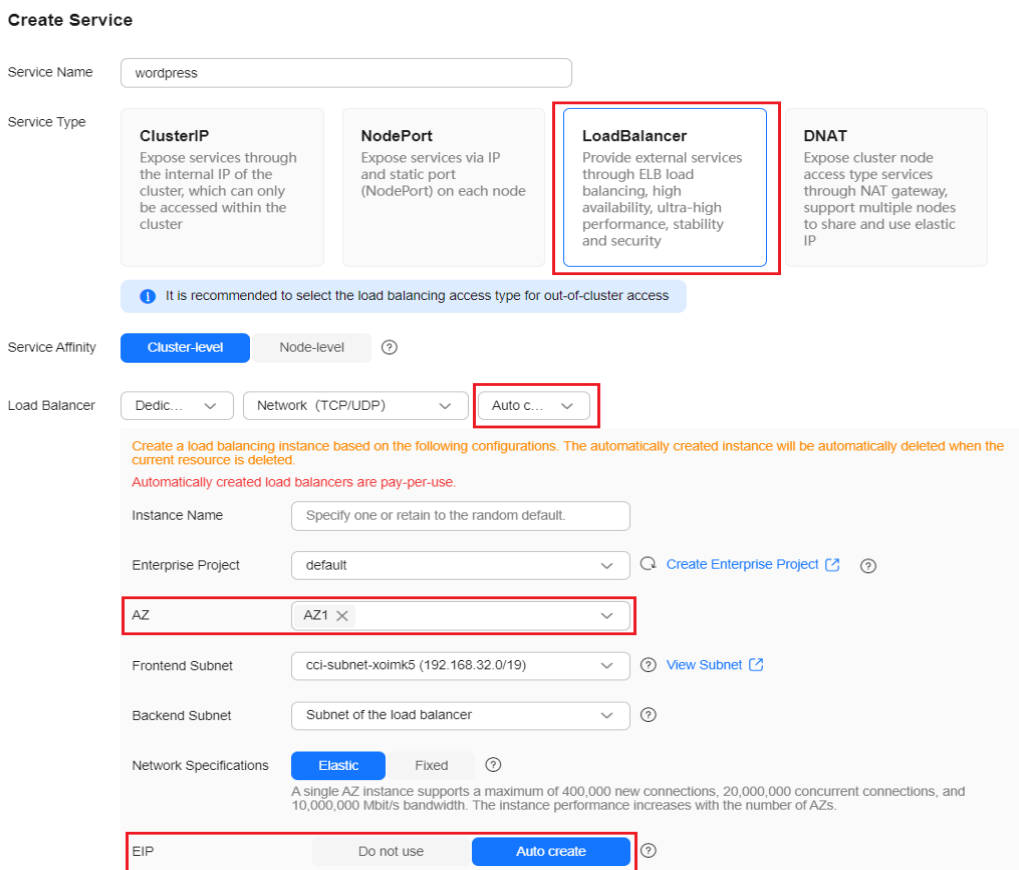
Parameter	Example	Description
Capacity (GiB)	10 GiB	Enter the capacity as required. The default value is <b>10 GiB</b> .

Click **Create** and enter the path for mounting the storage volume to the container. The default path used by WordPress is `/var/www/html`.



### Step 8 Configure access settings.

In the **Service Settings** area, click the plus sign (+) and create a Service for accessing the workload from external networks. This example shows how to create a LoadBalancer Service. You can configure the following parameters in the window that slides out from the right.



Parameter	Example	Description
Service Name	wordpress	Enter a Service name.
Service Type	LoadBalancer	Select a Service type, which refers to the Service access mode. For details about the differences between Service types, see <a href="#">Service</a> .
Load Balancer	<ul style="list-style-type: none"> <li>Dedicated</li> <li>AZ: at least one AZ, for example, <b>AZ1</b></li> <li>EIP: Auto create</li> </ul> <p>Keep the default values for other parameters.</p>	<p>Select <b>Use existing</b> if there is one.</p> <p>If no load balancer is available, select <b>Auto create</b> to create one and bind an EIP to it. For details about the parameters, see <a href="#">Creating a LoadBalancer Service</a>.</p>
Ports	<ul style="list-style-type: none"> <li>Protocol: TCP</li> <li>Container Port: 80</li> <li>Service Port: 8080</li> </ul>	<ul style="list-style-type: none"> <li><b>Protocol:</b> Select a protocol for the load balancer listener.</li> <li><b>Container Port:</b> Enter the listening port of the containerized application. The value must be the same as the listening port provided by the application for external systems. If the <b>wordpress</b> image is used, set this parameter to <b>80</b>.</li> <li><b>Service Port:</b> Enter a custom port. Load balancer will use this port to create a listener and provide an entry for external traffic. You can customize the port for external access.</li> </ul>

**Step 9 Click Create Workload.**

Wait until the workload is created. After it is created, it will be displayed on the **Deployments** tab.



----End

## Using kubectl

**Step 1** Log in to the ECS where kubectl has been installed.

**Step 2** Create a description file named **wordpress-deployment.yaml**. **wordpress-deployment.yaml** is an example file name. You can rename it as required.

```
vi wordpress-deployment.yaml
```

The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
      version: v1
  template:
    metadata:
      labels:
        app: wordpress
        version: v1
    spec:
      containers:
        - name: container-1
          image: wordpress:latest
          env:
            - name: WORDPRESS_DB_HOST
              value: mysql:3306
            - name: WORDPRESS_DB_USER
              value: db_user
            - name: WORDPRESS_DB_PASSWORD
              value: ****
            - name: WORDPRESS_DB_NAME
              value: database
      resources:
        requests:
          cpu: 250m
          memory: 512Mi
        limits:
          cpu: 250m
          memory: 512Mi
      volumeMounts:
        - name: wordpress
          readOnly: false
          mountPath: /var/www/html
      imagePullSecrets:
        - name: default-secret
      volumes:
        - name: wordpress
          persistentVolumeClaim:
            claimName: wordpress
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress
  namespace: default
annotations:
  everest.io/disk-volume-type: SSD
  everest.io/enterprise-project-id: '0'
  labels:
    failure-domain.beta.kubernetes.io/region: ap-southeast-1 # Region where the EVS disk is in
    failure-domain.beta.kubernetes.io/zone: # AZ where the EVS disk is in. It must be the same as the AZ
of the node that runs the workload.
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

```
requests:
  storage: 10Gi
storageClassName: csi-disk
```

**Step 3** Create the WordPress workload.

```
kubectl apply -f wordpress-deployment.yaml
```

Check the workload status.

```
kubectl get deployment
```

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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
wordpress	1/1	1	1	4m5s

**Step 4** Create a description file named **wordpress-service.yaml**. **wordpress-service.yaml** is an example file name. You can rename it as required.

```
vi wordpress-service.yaml
```

The file content is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  namespace: default
  annotations:
    kubernetes.io/elb.class: union
    kubernetes.io/elb.autocreate:
      {
        "type": "public",
        "bandwidth_name": "cce-wordpress",
        "bandwidth_chargemode": "bandwidth",
        "bandwidth_size": 5,
        "bandwidth_sharetype": "PER",
        "eip_type": "5_bgp"
      }
spec:
  selector:
    app: wordpress
  externalTrafficPolicy: Cluster
  ports:
    - name: cce-service-0
      targetPort: 80
      nodePort: 0
      port: 8080
      protocol: TCP
  type: LoadBalancer
```

**Step 5** Create a Service.

```
kubectl create -f wordpress-service.yaml
```

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

```
service/wordpress created
```

**Step 6** Check the Service.

```
kubectl get svc
```

If information similar to the following is displayed, the workload's access mode has been configured. You can use the LoadBalancer Service to access the WordPress workload from the Internet. **\*\*.\*\*.\*\*.\*\*** specifies the EIP of the load balancer, and **8080** indicates the access port.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.247.0.1	<none>	443/TCP	3d

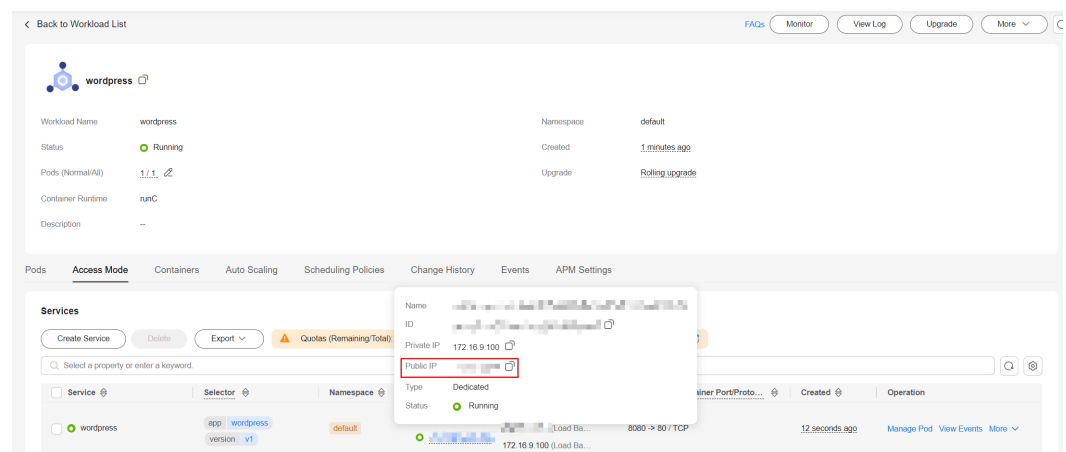
```
mysql ClusterIP 10.247.202.20 <none> 3306/TCP 8m
wordpress LoadBalancer 10.247.130.196 **.**.**.** 8080:31540/TCP 51s
```

----End

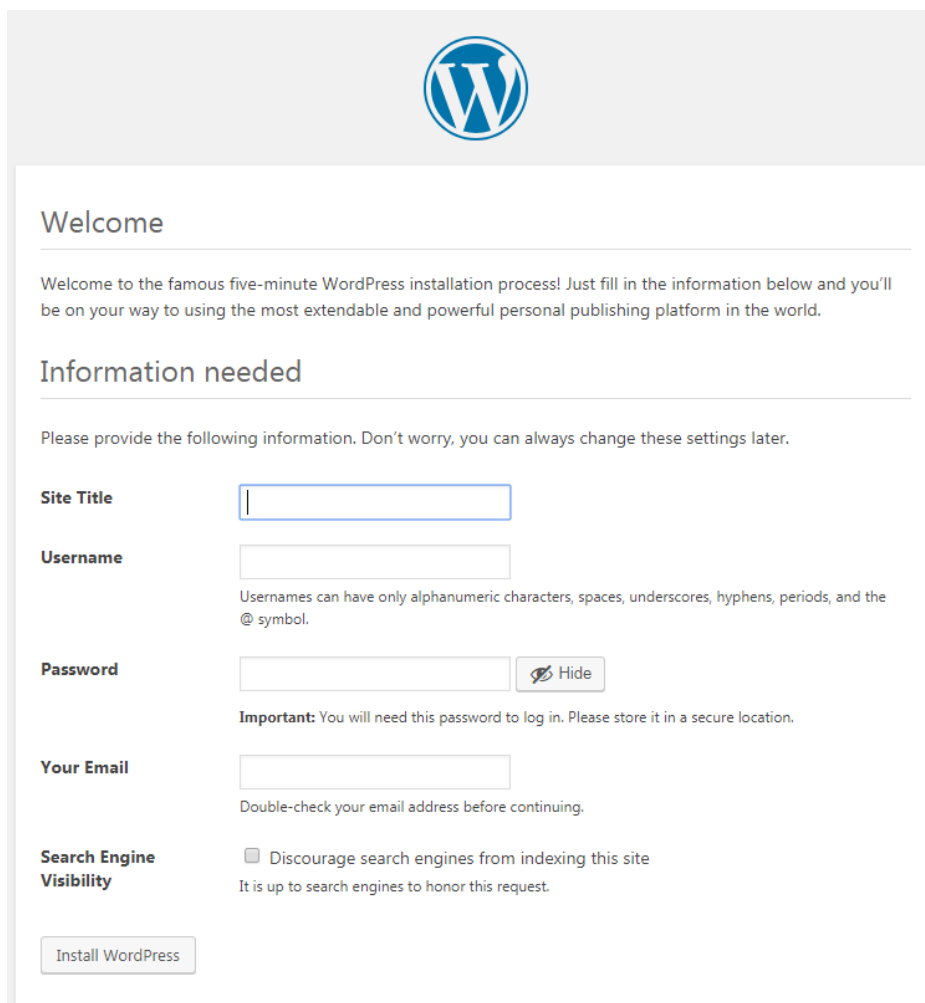
## Step 6: Access WordPress

**Step 1** Obtain the external access address of WordPress.

Click the WordPress workload name to enter its details page. On the page displayed, click the **Access Mode** tab, view the IP address of WordPress. The public IP address is the external access address.



**Step 2** Enter  $\{External\ access\ address:Port\}$  in the address box of a browser to access the application. The port number is the value of Service port configured in **Step 8**, which is **8080**.



The screenshot shows the WordPress installation 'Welcome' screen. At the top center is the WordPress logo. Below it, the heading 'Welcome' is followed by a paragraph: 'Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.' The next section is 'Information needed', with a sub-heading: 'Please provide the following information. Don't worry, you can always change these settings later.' The form contains several fields: 'Site Title' with an empty text box; 'Username' with an empty text box and a note: 'Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.'; 'Password' with an empty text box, a 'Hide' button, and an 'Important' note: 'You will need this password to log in. Please store it in a secure location.'; 'Your Email' with an empty text box and a note: 'Double-check your email address before continuing.'; and 'Search Engine Visibility' with a checkbox labeled 'Discourage search engines from indexing this site' and a note: 'It is up to search engines to honor this request.' At the bottom left is an 'Install WordPress' button.

----End

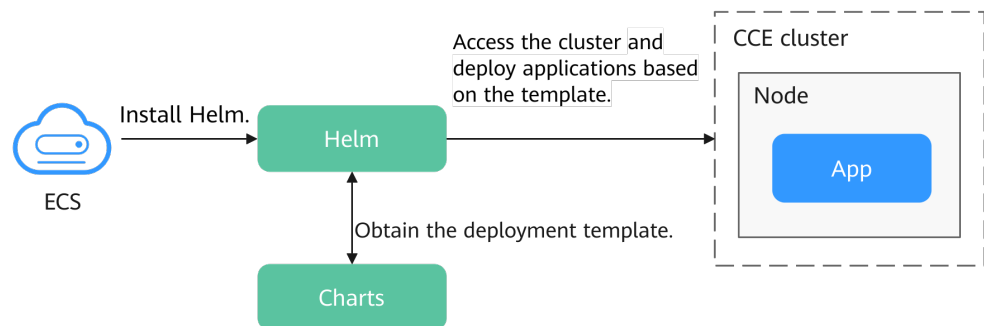
## Follow-up Operations: Releasing Resources

To avoid additional charges, make sure to release resources promptly if you no longer require the cluster. For details, see [Deleting a Cluster](#).

# 3 Deploying an Application in a CCE Cluster Using a Helm Chart

Helm is a package manager that streamlines the deployment, upgrade, and management of Kubernetes applications. Helm uses charts, which are a packaging format that defines Kubernetes resources, to package all components deployed by Kubernetes. This includes application code, dependencies, configuration files, and deployment instructions. By doing so, Helm enables the distribution and deployment of complex Kubernetes applications in a more efficient, consistent manner. Moreover, Helm facilitates application upgrade and rollback, simplifying application lifecycle management.

This section describes how to deploy a WordPress workload using Helm.



## Procedure

Step	Description
<b>Preparations</b>	Register a Huawei account and top up the account.
<b>Step 1: Enable CCE for the First Time and Perform Authorization</b>	Obtain the required permissions for your account when you use the CCE service in the current region for the first time.
<b>Step 2: Create a Cluster</b>	Create a CCE cluster to provide Kubernetes services.



Step	Description
<a href="#">Step 3: Create a Node Pool and Nodes in the Cluster</a>	Create a node in the cluster to run your containerized applications.
<a href="#">Step 4: Access the Cluster Using Kubectl</a>	Before using Helm charts, access the cluster on a VM using kubectl.
<a href="#">Step 5: Install Helm</a>	Install Helm on the VM with kubectl installed.
<a href="#">Step 6: Deploy the Template</a>	Create a WordPress workload in the cluster using the Helm installation command and create a Service for the workload for Internet access.
<a href="#">Step 7: Access WordPress</a>	Access the WordPress website from the Internet to start your blog.
<a href="#">Follow-up Operations: Releasing Resources</a>	To avoid additional charges, delete the cluster resources promptly if you no longer require them after practice.


## Preparations

- Before starting, register a Huawei account 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

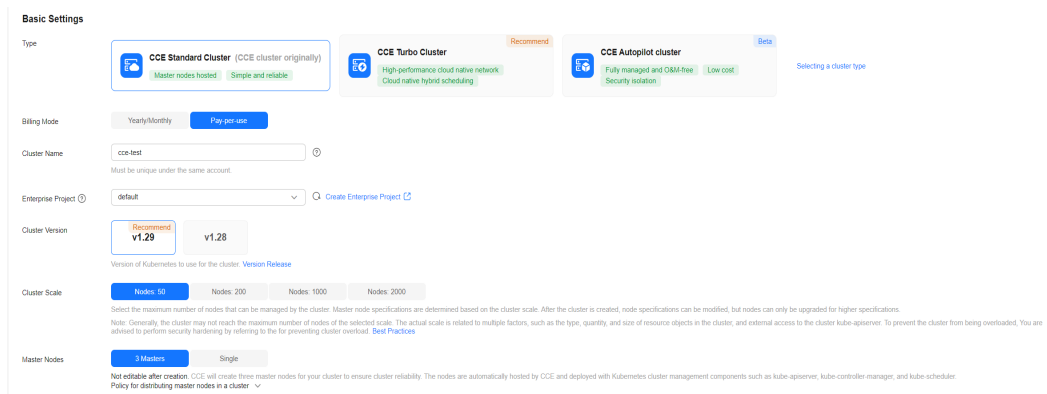
## Step 2: Create a Cluster

**Step 1** Log in to the [CCE console](#).

- If you have no clusters, click **Buy Cluster** on the wizard page.
- If you have CCE clusters, choose **Clusters** in the navigation pane, click **Buy Cluster** in the upper right corner.

**Step 2** Configure basic cluster parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the parameter configurations, see [Buying a CCE Standard/Turbo Cluster](#).

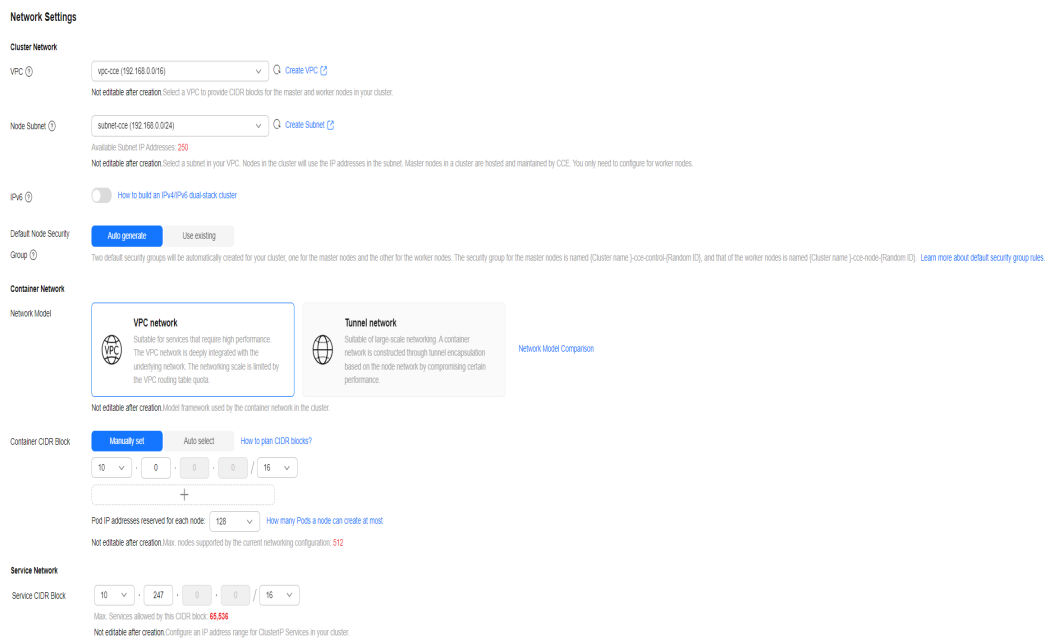


Parameter	Example	Description
Type	CCE Standard Cluster	<p>CCE allows you to create various types of clusters for diverse needs. It provides highly reliable, secure, business-class container services.</p> <p>You can select <b>CCE Standard Cluster</b> or <b>CCE Turbo Cluster</b> as required.</p> <ul style="list-style-type: none"> <li>• CCE standard clusters provide highly reliable, secure, business-class containers.</li> <li>• 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.</li> </ul> <p>For details about cluster types, see <a href="#">Comparison Between Cluster Types</a>.</p>

Parameter	Example	Description
Billing Mode	Pay-per-use	<p>Select a billing mode for the cluster.</p> <ul style="list-style-type: none"> <li>• <b>Yearly/Monthly:</b> a prepaid billing mode. Resources will be billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. If you choose this billing mode, you will need to set the desired duration and decide whether to enable automatic subscription renewal. Monthly subscriptions renew automatically every month, while yearly subscriptions renew automatically every year.</li> <li>• <b>Pay-per-use:</b> a postpaid billing mode. It is suitable for scenarios where resources will be billed based on usage frequency and duration. You can provision or delete resources at any time.</li> </ul> <p>For details, see <a href="#">Billing Modes</a>.</p>
Cluster Name	cce-test	Name of the cluster to be created
Enterprise Project	default	<p>Enterprise projects facilitate project-level management and grouping of cloud resources and users. For more details, see <a href="#">Enterprise Management</a>.</p> <p>This parameter is displayed only for enterprise users who have enabled Enterprise Project Management.</p>
Cluster Version	<b>The recommended version, for example, v1.29</b>	Select the latest commercial release for improved stability, reliability, new functionalities. CCE offers various versions of Kubernetes software.
Cluster Scale	Nodes: 50	Configure the parameter as required. This parameter controls the maximum number of worker nodes that the cluster can manage. After the cluster is created, it can only be scaled out.

Parameter	Example	Description
Master Nodes	3 Masters	<p>Select the number of master nodes. The master nodes are automatically hosted by CCE and deployed with Kubernetes cluster management components such as kube-apiserver, kube-controller-manager, and kube-scheduler.</p> <ul style="list-style-type: none"> <li>● <b>3 Masters:</b> Three master nodes will be created for high cluster availability.</li> <li>● <b>Single:</b> Only one master node will be created in your cluster.</li> </ul> <p>This parameter cannot be changed after the cluster is created.</p>

### Step 3 Configure network parameters.

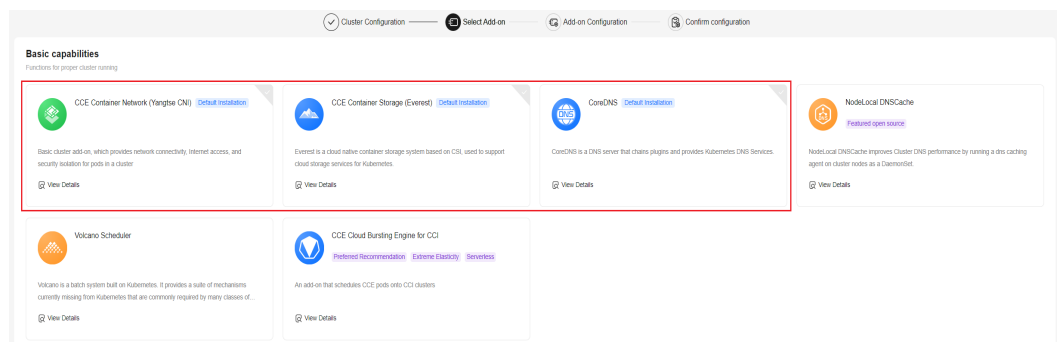


Parameter	Example	Description
VPC	vpc-cce	<p>Select a VPC for the cluster.</p> <p>If no VPC is available, click <b>Create VPC</b> to create one. After the VPC is created, click the refresh icon. For details about how to create a VPC, see <a href="#">Creating a VPC and Subnet</a>.</p>
Node Subnet	subnet-cce	<p>Select a subnet. Nodes in the cluster are assigned with the IP addresses in the subnet.</p>

Parameter	Example	Description
Network Model	VPC network	Select <b>VPC network</b> or <b>Tunnel network</b> . By default, the VPC network model is selected. For details about the differences between different container network models, see <a href="#">Container Network</a> .
Container CIDR Block	10.0.0.0/16	Configure the CIDR block used by containers. It controls how many pods can run in the cluster.
Service CIDR Block	10.247.0.0/16	Configure the ClusterIP CIDR block for the cluster. It controls how many Services can be created in the cluster and cannot be changed after configuration.

**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.



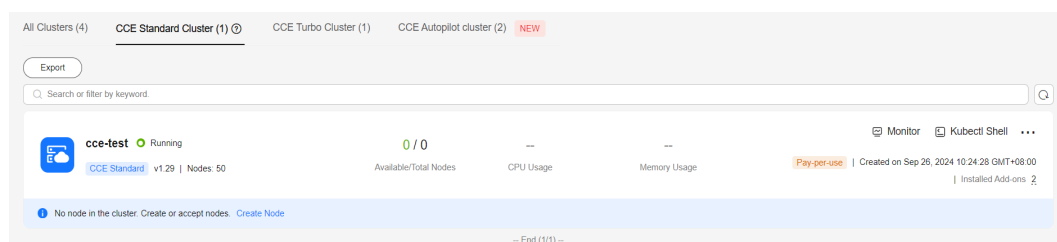
**Step 5** Click **Next: Add-on Configuration**. There is no need to set up the add-ons that are installed by default.

**Step 6** Click **Next: Confirm configuration**, confirm the resources on the page displayed, and click **Submit**.

Wait until the cluster is created. It takes about 5 to 10 minutes to create a cluster.

The created cluster will be displayed on the **Clusters** page, and there are zero nodes in it.

**Figure 3-1** Cluster created

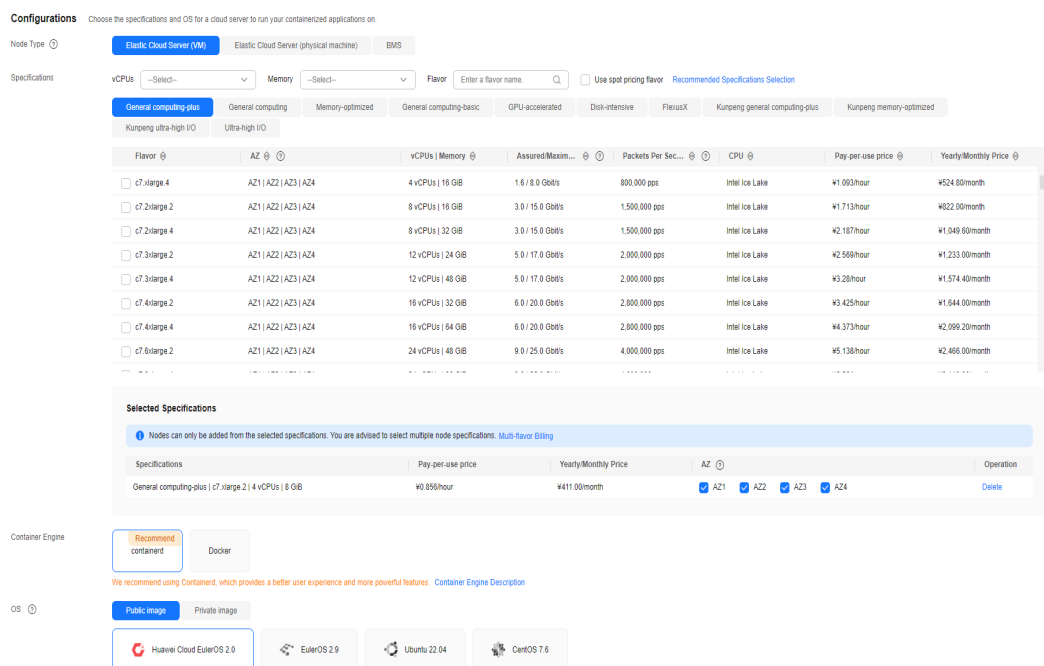


----End

### Step 3: Create a Node Pool and Nodes in the Cluster

- 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 **Nodes**. On the **Node Pools** tab, click **Create Node Pool** in the upper right corner.
- Step 3** Configure the node pool parameters.

Only mandatory parameters are described in this example. You can keep the default values for most other parameters. For details about the configuration parameters, see [Creating a Node Pool](#).



Parameter	Example	Description
Node Type	Elastic Cloud Server (VM)	Select a node type based on service requirements. Then, the available node flavors will be automatically displayed in the <b>Specifications</b> area for you to select.
Specifications	4 vCPUs   8 GiB	Select a node flavor that best fits your service needs.  For optimal performance of the cluster components, you are advised to set up the node with a minimum of 4 vCPUs and 8 GiB of memory.

Parameter	Example	Description
Container Engine	containerd	Select a container engine based on service requirements. For details about the differences between container engines, see <a href="#">Container Engines</a> .
OS	Huawei Cloud EulerOS 2.0	Select an OS for the node.
Login Mode	<b>A custom password</b>	<ul style="list-style-type: none"> <li><b>Password:</b> Enter a password for logging in to the node and confirm the password. The default username is <b>root</b>. Keep the password secure. If you forget the password, the system is unable to retrieve it.</li> <li><b>Key Pair:</b> Select a key pair for logging to the node and select the check box to acknowledge that you have obtained the key file and without this file you will not be able to log in to the node. A key pair is used for identity authentication when you remotely log in to a node. If no key pair is available, click <b>Create Key Pair</b> to create one. For details, see <a href="#">Creating a Key Pair on the Management Console</a>.</li> </ul>

**Step 4** Configure parameters in **Storage Settings** and **Network Settings**. In this example, you can keep the default values for the parameters. You only need to select **I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.** and click **Next: Confirm**.

**Storage Settings** Configure storage resources for containers and applications on the node.

System Disk   GB

Expand  System Disk Encryption: Not encrypted

Data Disk   GB | Quantity

Used by the container runtime and kubelet. Do not uninstall this disk. Otherwise, the node will become unavailable. [How do I set data disk size?](#) [How do I allocate data disk space?](#)

Expand  Container Engine: Shared disk space | Pod: All | Write Mode: Linear | Data Disk Encryption: Not encrypted

You can add **15** more EVS data disks.

---

**Network Settings** Configure networking resources for node and application communication.

Virtual Private Cloud

Node subnet     Available Subnet IP Addresses: **248**

If the single subnet IP resources associated with your node pool are tight, it is recommended that you configure multiple subnets for the node pool.

Associate Security Group

Configure security group rules for worker nodes in your cluster. The rules will take effect on all worker nodes in the cluster. For additional security group configurations, go to the Security Groups page. The security group for master nodes is automatically created by CCE. [View Default Security Group Rules](#)  
Not editable after creation.

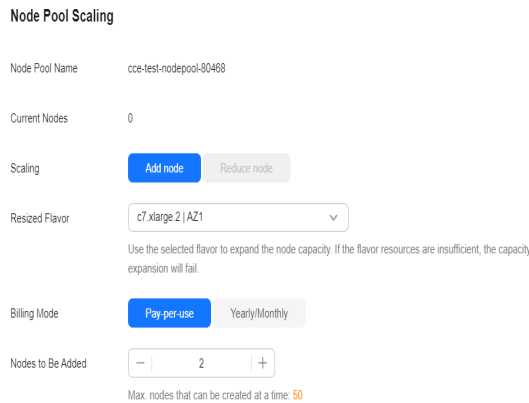
I have confirmed that the security group rules have been correctly configured for nodes to communicate with each other.

**Step 5** Check the node specifications, read the instructions on the page, and click **Submit**.

**Step 6** Locate the row containing the target node pool and click **Scaling**. There are zero nodes in the created node pool by default.



**Step 7** Set the number of nodes to be added to **2**, which means two more nodes will be created in the node pool.



**Step 8** Wait until the nodes are created. It takes about 5 to 10 minutes to complete the node creation.



----End

## Step 4: Access the Cluster Using Kubectl

### NOTICE

You need to create an ECS **bound with an EIP in the same VPC as the cluster** first.

**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.

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

1. Download kubectl.

```
cd /home
curl -LO https://dl.k8s.io/release/{v1.29.0}/bin/linux/amd64/kubectl
```

{v1.29.0} specifies the version. You can replace it as required.



2. Install kubectl.  

```
chmod +x kubectl  
mv -f kubectl /usr/local/bin
```

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

1. Log in to the [CCE console](#) and click the cluster name to access the cluster console. Choose **Overview** in the navigation pane.
2. On the cluster overview page, locate the **Connection Info** 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 corresponding configuration file.
4. Log in to the VM where the kubectl client has been installed and copy and paste the configuration file (for example, **kubeconfig.yaml**) downloaded in the previous step 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 see whether the cluster can be accessed.

For example, to view the cluster information, run the following command:

```
kubectl cluster-info
```

Information similar to the following is displayed:

```
Kubernetes master is running at https://*:*:5443  
CoreDNS is running at https://*:*:5443/api/v1/namespaces/kube-system/services/coresdns/dns/proxy  
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

----End

## Step 5: Install Helm

This section uses Helm v3.7.0 as an example. If other versions are needed, see [Helm](#).

**Step 1** Download the Helm client to a VM in a cluster.

```
wget https://get.helm.sh/helm-v3.7.0-linux-amd64.tar.gz
```

**Step 2** Decompress the Helm package.

```
tar -xzf helm-v3.7.0-linux-amd64.tar.gz
```

**Step 3** Copy and paste Helm to the system path, for example, **/usr/local/bin/helm**.

```
mv linux-amd64/helm /usr/local/bin/helm
```

**Step 4** Check the Helm version.

```
helm version  
version.BuildInfo{Version:"v3.7.0",GitCommit:"eeac83883cb4014fe60267ec6373570374ce770b",GitTreeState:"  
clean",GoVersion:"g01.16.8"}
```

----End

## Step 6: Deploy the Template

This section uses the WordPress template as an example.

**Step 1** Add the official WordPress repository.

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

**Step 2** Run the following commands to create a WordPress workload:

```
helm install myblog bitnami/wordpress \
--set mariadb.primary.persistence.enabled=true \
--set mariadb.primary.persistence.storageClass=csi-disk \
--set mariadb.primary.persistence.size=10Gi \
--set persistence.enabled=false
```

The custom instance name is specified by *myblog*. The remaining parameters serve the following functions:

- Persistent storage volumes are used by the MariaDB database that is connected to WordPress to store data. StorageClass is used to automatically create persistent storage. The EVS disk type (csi-disk) is used, with a size of 10GiB.
- WordPress requires no data persistence, so you can set **persistence.enabled** to **false** for the PV.

The command output is as follows:

```
coalesce.go:223: warning: destination for mariadb.networkPolicy.egressRules.customRules is a table.
Ignoring non-table value ([])
NAME: myblog
LAST DEPLOYED: Mon Mar 27 11:47:58 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: wordpress
CHART VERSION: 15.2.57
APP VERSION: 6.1.1
```

**\*\* Be patient while the chart is being deployed.\*\***

Your WordPress site can be accessed through the following DNS name from within your cluster:

```
myblog-wordpress.default.svc.cluster.local (port 80)
```

To access your WordPress site from outside the cluster, follow the steps below:

1. Get the WordPress URL by running these commands:

```
NOTE: It may take a few minutes for the LoadBalancer IP to be available.
Watch the status with: 'kubectl get svc --namespace default -w myblog-wordpress'

export SERVICE_IP=$(kubectl get svc --namespace default myblog-wordpress --template "{{ range
(index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")
echo "WordPress URL: http://$SERVICE_IP/"
echo "WordPress Admin URL: http://$SERVICE_IP/admin"
```

2. Open a browser and access WordPress using the obtained URL.

3. Log in with the following credentials below to see your blog:

```
echo Username: user
echo Password: $(kubectl get secret --namespace default myblog-wordpress -o
jsonpath="{.data.wordpress-password}" | base64 -d)
```

**----End**

## Step 7: Access WordPress

### Step 1 Modify the WordPress Service configuration.

To use a LoadBalancer Service in CCE, you need to configure it with additional annotations. Unfortunately, **bitnami/wordpress** does not come with this configuration, so you will have to modify it manually.

```
kubectl edit svc myblog-wordpress
```

Add **kubernetes.io/elb.autocreate** and **kubernetes.io/elb.class** to **metadata.annotations** and save the changes. These two annotations are used to create a shared load balancer, which allows access to the WordPress workload via the EIP of the load balancer.

```
apiVersion: v1
kind: Service
metadata:
  name: myblog-wordpress
  namespace: default
  annotations:
    kubernetes.io/elb.autocreate: '{ "type": "public", "bandwidth_name": "myblog-wordpress",
"bandwidth_chargemode": "bandwidth", "bandwidth_size": 5, "bandwidth_sharetype": "PER", "eip_type":
"5_bgp" }'
    kubernetes.io/elb.class: union
spec:
  ports:
    - name: http
  ...
```

### Step 2 Check the Service.

```
kubectl get svc
```

If information similar to the following is displayed, the workload's access mode has been configured. You can use the LoadBalancer Service to access the WordPress workload from the Internet. **\*\*.\*\*.\*\*.\*\*** specifies the EIP of the load balancer, and **80** indicates the access port.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.247.0.1	<none>	443/TCP	3d
myblog-mariadb	ClusterIP	10.247.202.20	<none>	3306/TCP	8m
<b>myblog-wordpress</b>	<b>LoadBalancer</b>	<b>10.247.130.196</b>	<b>**.**.**.**</b>	<b>80:31540/TCP</b>	<b>8m</b>

### Step 3 Access WordPress.

- To access the WordPress web page: In the address box of a browser, enter **<EIP of the load balancer>:80**.

User's Blog!

Sample Page

## Mindblown: a blog about philosophy.

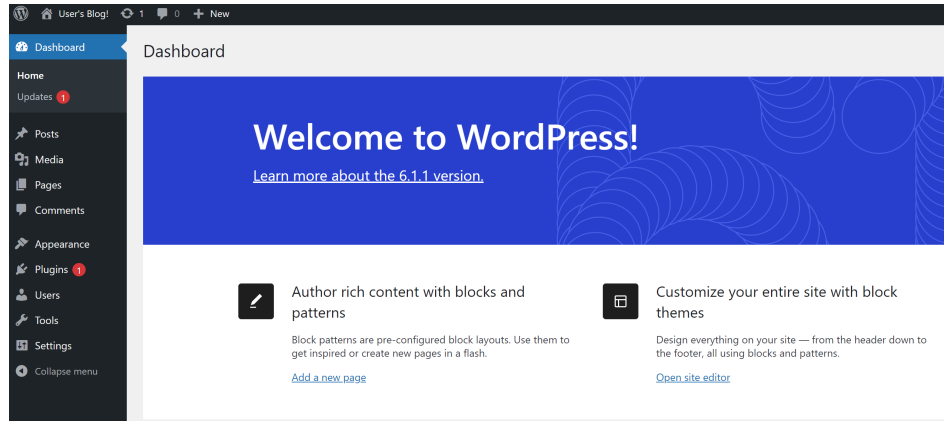
Hello world!

Welcome to WordPress. This is your first post.  
Edit or delete it, then start writing!

March 27, 2023

- To access the WordPress management console:
  - a. Run the following command to obtain the password of **user**:

```
kubectl get secret --namespace default myblog-wordpress -o jsonpath="{.data.wordpress-password}" | base64 -d
```
  - b. In the address box of a browser, enter **<EIP of the load balancer>:80/login** to access the WordPress backend. The user name is **user**, and the password is the character string obtained in the previous step.



----End

## Follow-up Operations: Releasing Resources

To avoid additional charges, make sure to release resources promptly if you no longer require the cluster. For details, see [Deleting a Cluster](#).