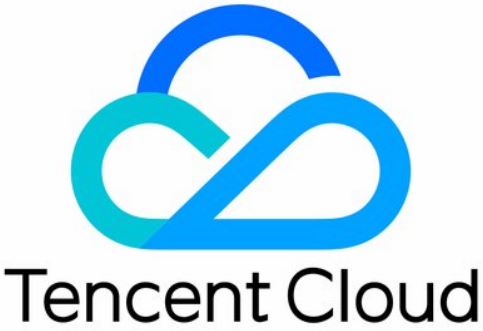
Tencent Kubernetes Engine

Quick Start

Product Documentation



Contents

Quick Start
Beginner's Guide

Last updated：2023-05-22 17:28:43

This document helps you quickly understand and get started with Tencent Kubernetes Engine (TKE) as instructed.

1. What Is TKE?

Based on the native Kubernetes system, Tencent Kubernetes Engine (TKE) provides container-centric, highly scalable and high-performance container management services. It works closely with Tencent Cloud IaaS products to help you quickly implement business containerization. For more information, see Overview.
TKE allows you to manipulate clusters and services in the TKE console or TencentCloud API.

2. TKE Billing

TKE allows you to create different types of Kubernetes clusters with different billable items and billing standards. For more information about the billing modes and prices, see Purchase Guide.

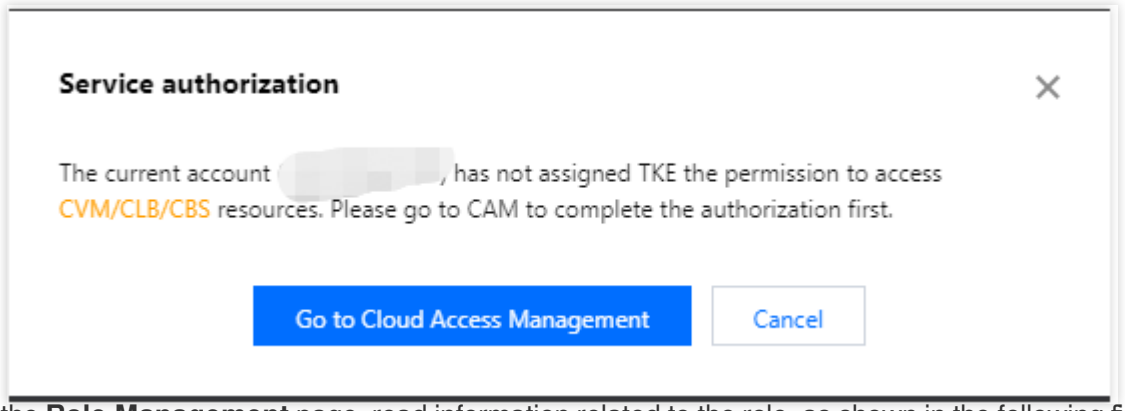
3. Using TKE

3.1 Register on Tencent Cloud

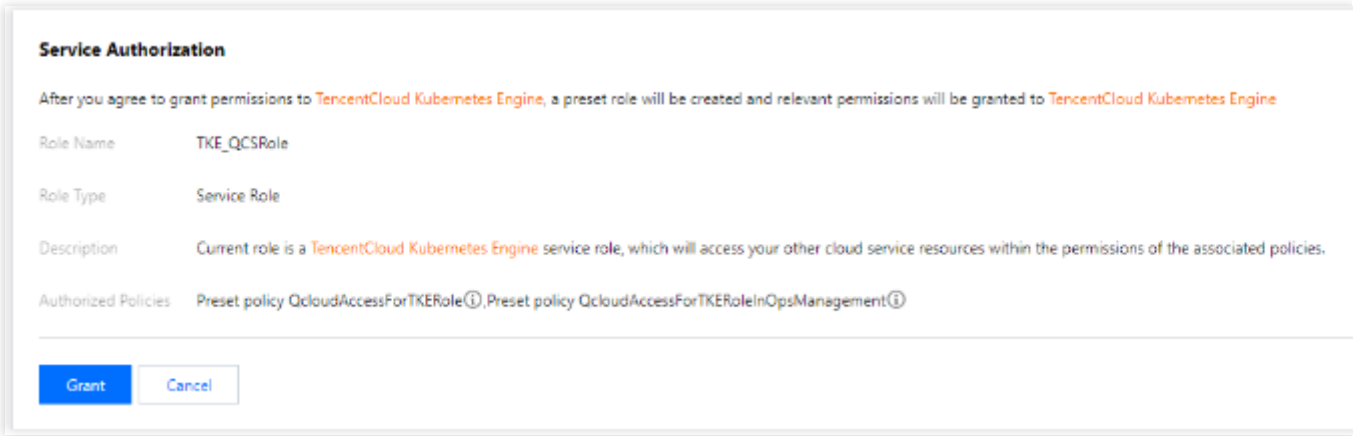
Before using TKE, you need to sign up for a Tencent Cloud account and complete the identity verification.

3.2 Role authorization

You need to authorize the current service role and grant operation permissions for TKE before accessing your other Tencent Cloud service resources.
Open the Tencent Cloud console, select **Products** > **Tencent Kubernetes Engine** to enter the TKE console and authorize TKE according to the prompts. After that, get relevant resource operation permissions, and you can start to create a cluster. Steps are as follows:
1. View information in the displayed **Service Authorization** dialog box, and click **Go to Cloud Access Management**, as shown in the following figure.



2. On the **Role Management** page, read information related to the role, as shown in the following figure.



3. Click **Grant** to grant authorization. Now you can go to the TKE console to create clusters and purchase related products.

3.3 Creating a cluster

For how to quickly create a standard managed cluster, see Quickly Creating a Standard Cluster. For the complete process of creating a standard managed cluster, see Creating a Cluster.
If you need more types of clusters, see Creating Serverless Cluster, Creating a Container Instance, and Creating Edge Cluster.

3.4 Deploying workloads

You can deploy workloads by deploying images or orchestrating the YAML file.
If you want to deploy stateless workloads through image templates, see directions in Creating Simple Nginx Service or WordPress with Single Pod.
If you want to deploy workloads through custom images, see directions in Building Hello World Service Manually.

3.5 Cluster operations

TKE is a management platform for clusters, applications, storage and networks. For more information or directions, please refer to the table below.

Operation	Reference

Connect to a TKE cluster from a local client using Kubectl, the Kubernetes command line tool	Connecting to a Cluster
Upgrade a running Kubernetes cluster	Upgrading a Cluster
Add a pod to the created Kubernetes cluster	Adding a Node
Manage nodes in a Kubernetes cluster	Creating a Node Pool
Operate native Kubernetes objects in the console	Kubernetes Object Management
Provide a fixed access entry for a set of containers through service	Basic Features
Configure different forwarding rules through Ingress resources	Ingress Management
Leverage TKE's storage capability	Storage Management Overview
Assign the IP addresses within the container network address range to containers in the cluster	Container Network Overview
Store and analyze service logs in Kubernetes clusters	Log Collection
Monitor clusters	TMP Instance Management
Use a private image hosted in Tencent Container Registry (TCR) to deploy applications	Using a Container Image in a TCR Enterprise Instance to Create a Workload

4. Beginner's Guide

Can I use TKE in classic network?

No. Currently, you can use TKE in a VPC but not a classic network.

Can I add an existing CVM to a cluster?

Yes. After creating a cluster, you can add an existing CVM to it. For more information, see Adding a Node.

Why does my service keep starting?

If there is no process running in the container, the service may keep starting. For more information on service startup, see Event FAQs.

How do I perform network planning before creating a cluster?

When creating a cluster, make sure that the IP ranges of the cluster network and container network do not overlap. Generally, you can select a subnet of a VPC instance as the node network of the cluster. For more information, see Container Network and Cluster Network Descriptions.

How do I access a created service?

Different access methods have different access entries. For more information, see the "Service Access" section in Service Management Overview.

**How does a container access the public network?

If the host where the container resides has a public IP address and public bandwidth, the container can directly access the public network. Otherwise, a NAT gateway is required for accessing the public network.

Can I use TKE if I don't know how to create an image?

The features related to Helm 3.0 that are integrated in TKE enable you to create products and services such as Helm Chart, TCR, and software services. Created applications will run in the cluster you specify to offer corresponding capabilities. For more information, see Managing Applications.

How do I manage configuration files or environment variables for my services?

You can manage configuration files by editing configuration items.

How do services access each other?

In a cluster, services with the same namespace can directly access one another, whereas those with different namespaces access one another by using `<service-name>.<namespace-name>.svc.cluster.local` .

5. Feedback and Suggestions

If you have any doubts or suggestions when using TKE products and services, you can submit your feedback through the following channels. Dedicated personnel will contact you to solve your problems.

For questions about the product documentation, such as links, content, or APIs, click **Send Feedback** on the right of the document page.

If you have any questions about products, submit a ticket.

Quickly Creating a Standard Cluster

Last updated：2023-09-26 15:38:25

This document describes how to quickly create a container cluster using TKE.

Step 1. Sign Up for a Tencent Cloud Account

Before using TKE, you need to sign up for a Tencent Cloud account and complete the identity verification.

Step 2. Top Up Online

TKE charges cluster management fees based on the specifications of the managed clusters, and charges cloud resources fees based on the actual usage. For billing modes and prices, see TKE Billing Overview. In this document, a managed cluster is created. You still need to pay for services such as cluster worker nodes, persistent storage, and CLB instances bound to the service. Before making a purchase, top up your account as instructed in Payment Methods.

Step 3. Authorize TKE

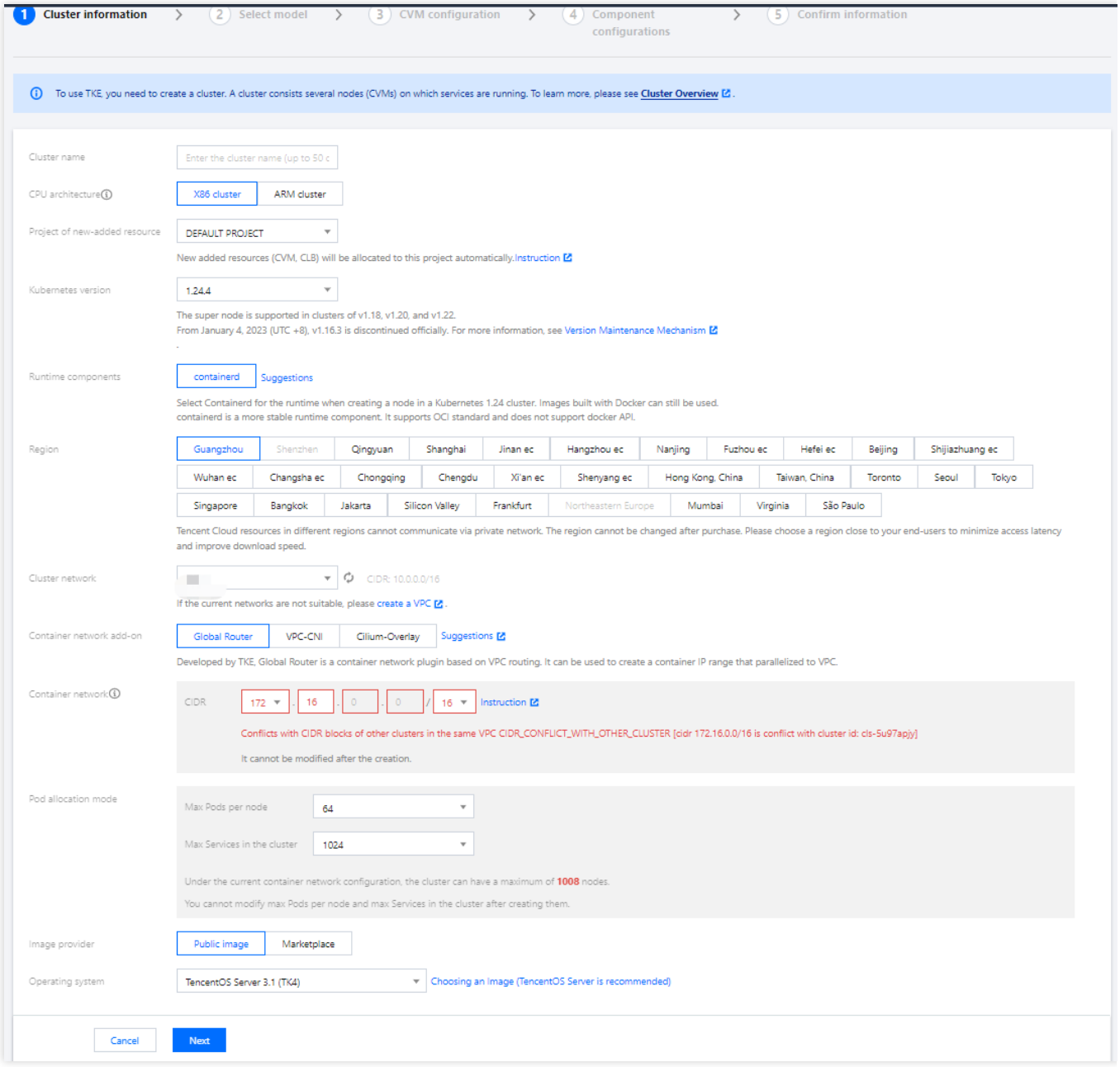
Log in to the Tencent Cloud console, select **Tencent Cloud services** > **Tencent Kubernetes Engine** to enter the TKE console and authorize TKE according to the prompts. If you have already authorized TKE, skip this step.

Step 4. Create a Cluster

Log in to the TKE console to create a cluster.

Cluster information

On the **Cluster Information** page, enter the cluster name and select the region of the cluster, the cluster network, and container network. Keep other default options unchanged and click **Next**.



Cluster Name: Enter the cluster name. We use "test" as the cluster name in this document.

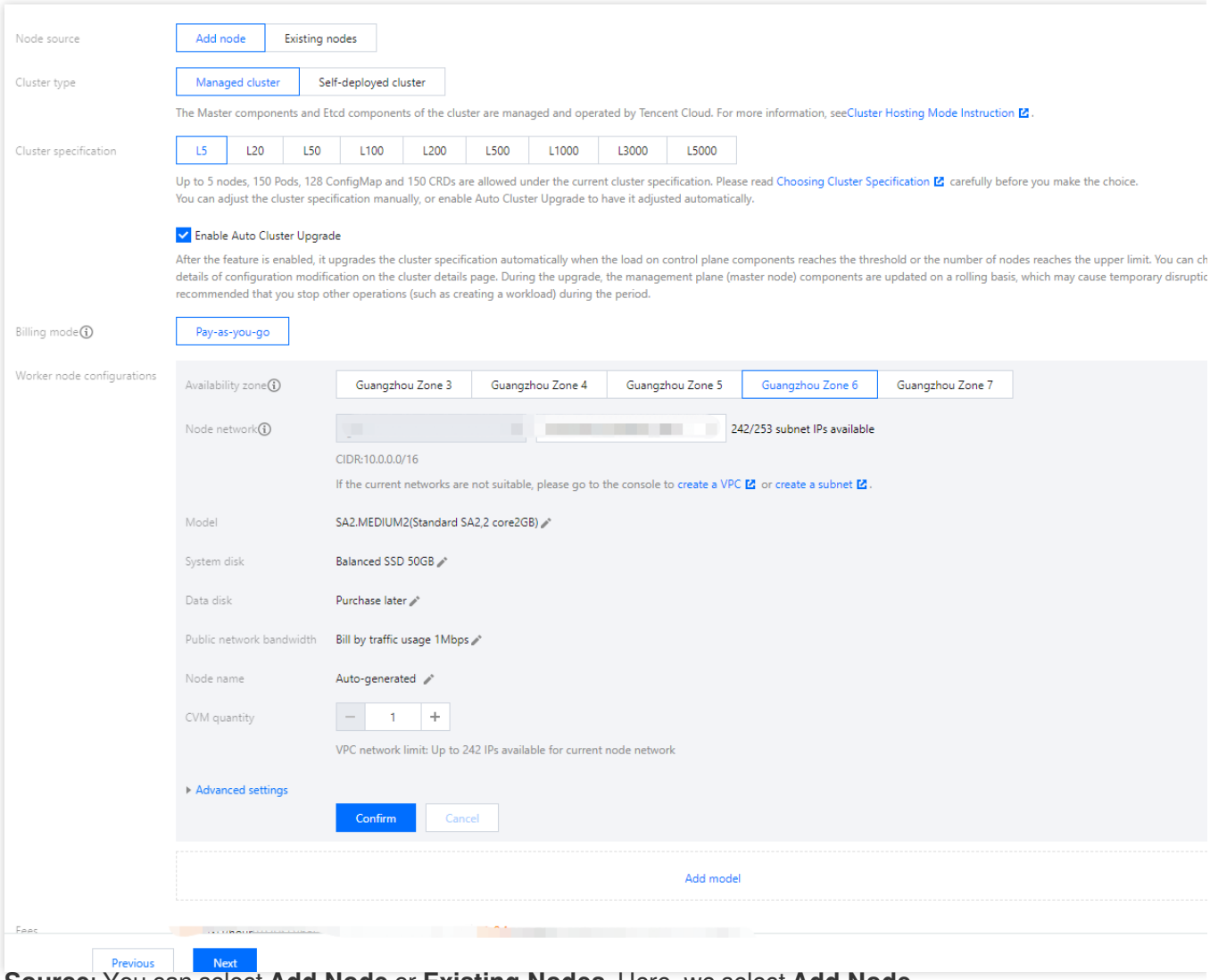
Region: select a region closest to you. For example, if you are in “Shenzhen”, please select “Guangzhou”.

Cluster Network: Assign IP addresses within the node network address range to the servers in the cluster. Here we select VPC.

Container Network: Assign IP addresses within the container network address range to the containers in the cluster. Here, we select an available container network.

Selecting a model

On the **Select Model** page, confirm the billing mode, select an availability zone and the corresponding subnet, confirm the node model, and click **Next**.



Node Source: You can select **Add Node** or **Existing Nodes**. Here, we select **Add Node**.

Cluster Type: You can select **Self-deployed Cluster** or **Managed Cluster**. Here we select **Managed Cluster**.

Cluster Specification: Multiple cluster specifications are provided. Here we select **L5**.

Billing Mode: Only **Pay-as-you-go** is available.

Worker Configurations: You only need to select an availability zone and the corresponding subnet and confirm the node model. Keep other default settings unchanged.

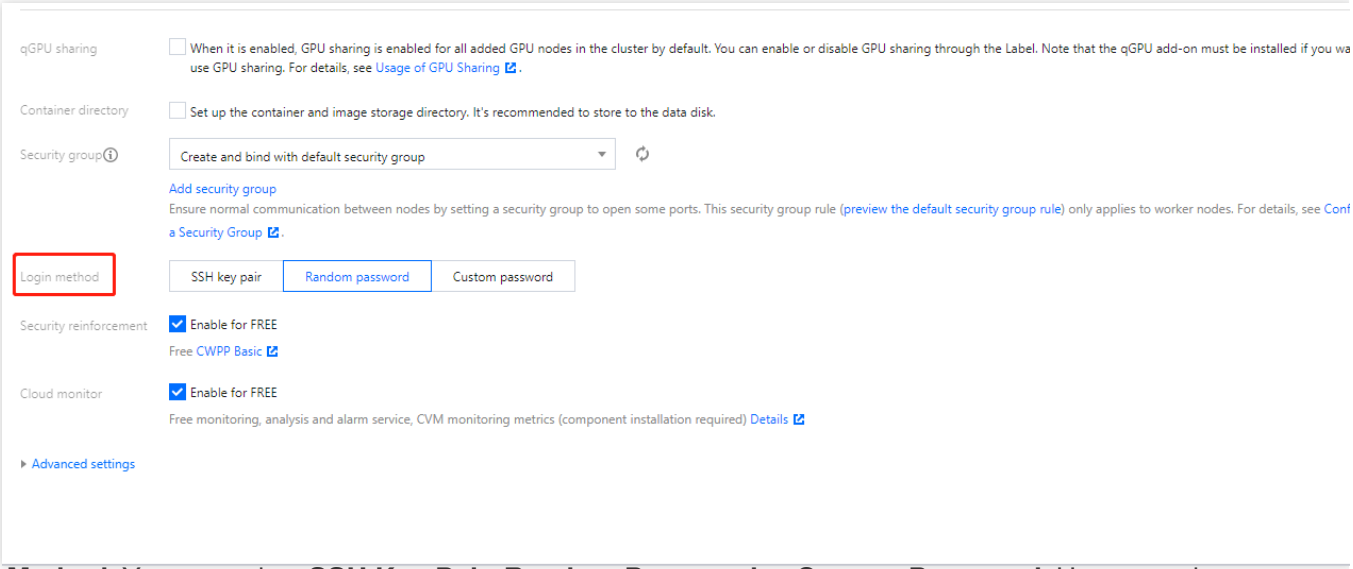
Availability Zone: Here we select **Guangzhou Zone 6**.

Node Network: Here we select the subnet under the current VPC.

Model: Here we select **SA2.MEDIUM2 (Standard SA2, 2-core 2 GB)**.

CVM configuration

On the **CVM configuration** page, select the login method, keep other default settings unchanged, and click **Next**.



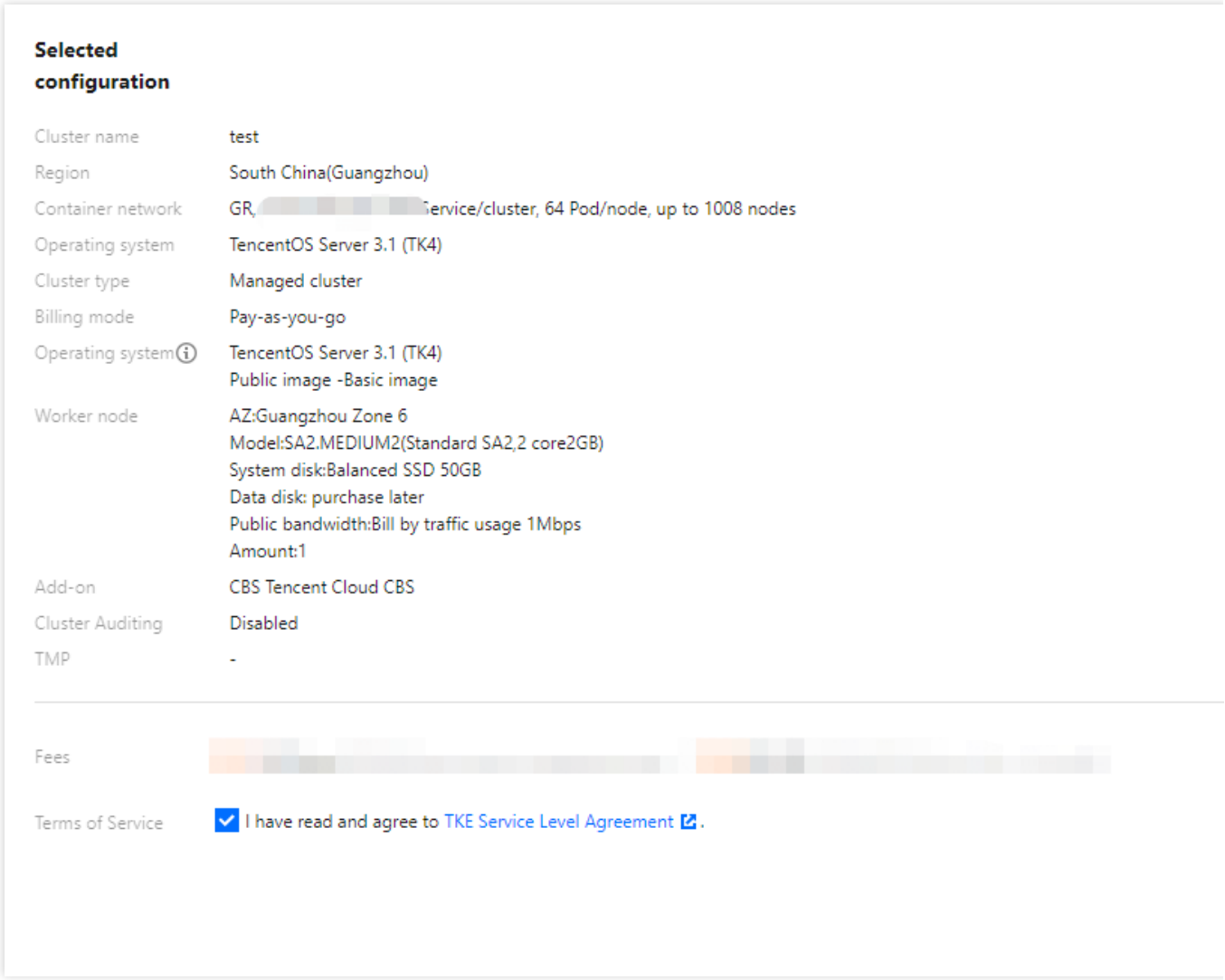
Login Method: You can select **SSH Key Pair**, **Random Password** or **Custom Password**. Here we select "Random Password".

Add-on configurations

On the **Add-on Configurations** page, you can choose the add-ons you need, including storage, monitoring, and image. If you don't need them, click **Next**. Here, we choose not to install add-ons and keep other default settings unchanged.

Information confirmation

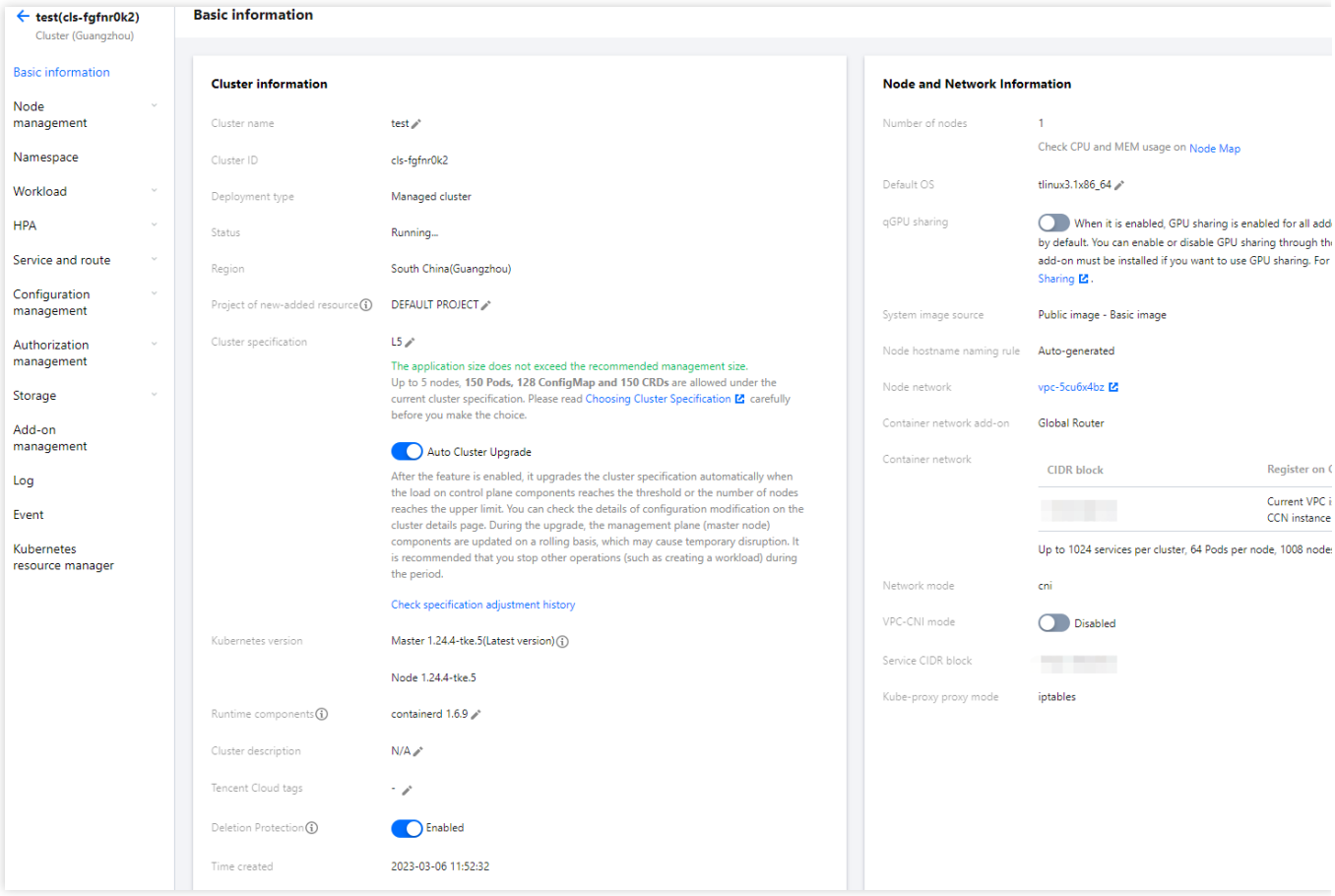
On the **Confirm Info** page, confirm the selected configuration information and billing mode for the cluster, read and indicate your consent to the TKE SLA, and click **Done**.



You can create your first TKE general cluster after making the payment. Then, you can view the created cluster in the TKE console.

Step 5. View the Cluster

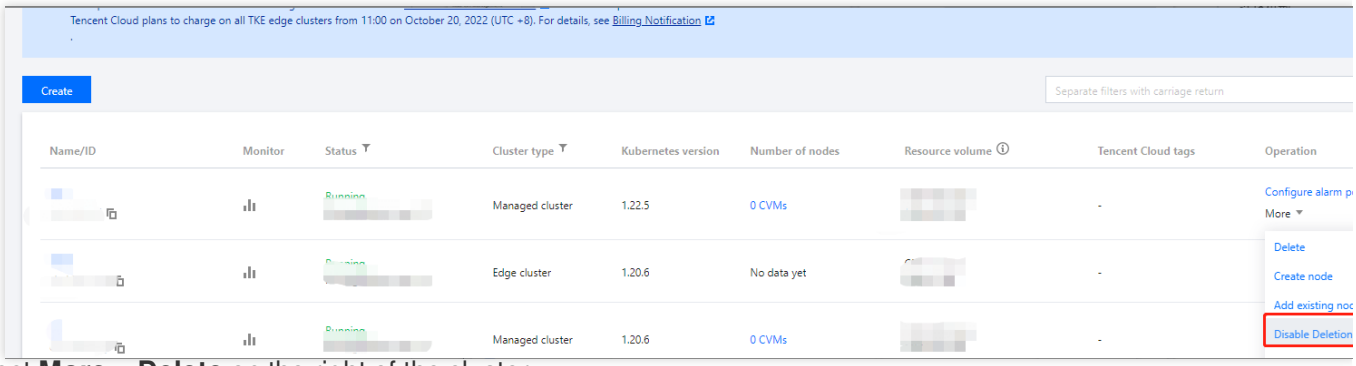
You can view clusters that have been created in the cluster list. You can click the cluster ID to enter the details page, and then view the cluster, node, network, and API Server information on the **Basic Information** page.



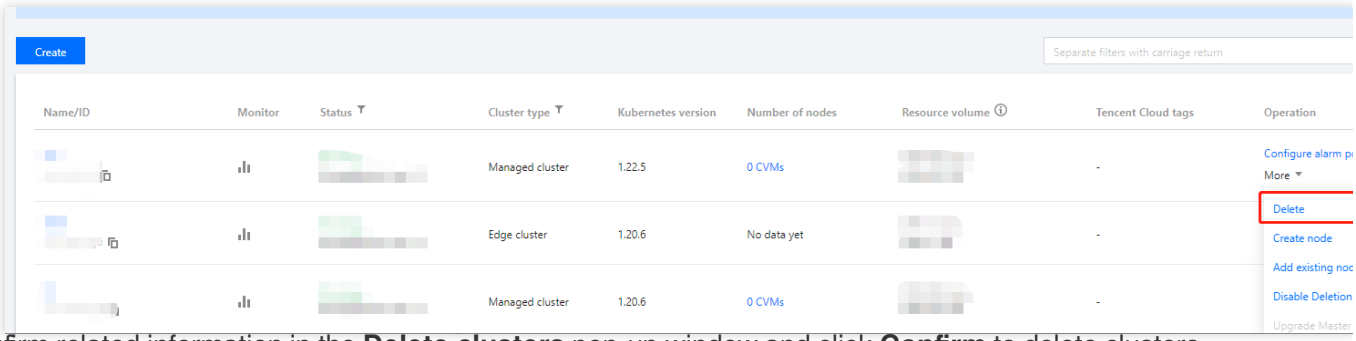
Step 6. Delete Clusters

Once started, clusters will start to consume resources. To avoid unnecessary costs, you can follow the steps below to clear all the resources.

1. Select **Cluster** on the left sidebar. On the **Cluster Management** page, select **More** > **Disable Deletion Protection** on the right of the target cluster.



2. Select **More** > **Delete** on the right of the cluster.



3. Confirm related information in the **Delete clusters** pop-up window and click **Confirm** to delete clusters.

Subsequent Operation: Using a Cluster

Now you know how to create and delete clusters in TKE. You can set workloads and create services in the clusters. Common tasks include:

Creating Simple Nginx Service

WordPress with Single Pod

WordPress Service using TencentDB

Building Hello World Service Manually

Building a Simple Web Application

Troubleshooting

For detailed directions on how to create a general cluster in the TKE console, see Creating a Cluster. If you encounter any problems during the use, contact us for assistance.

Creating a Container Instance

Last updated：2022-12-13 17:16:30

Step 1. Sign up and top up

1. Sign up for a Tencent Cloud account and complete identity verification.
   If you already have a Tencent Cloud account, ignore this step.
2. Top up online.
   TKE serverless cluster provides two billing modes: Pay-as-you-go and reservation. Before purchasing a container instance, you need to top up your account as instructed in Payment Methods.

Step 2. Authorize the service

1. Container instance is in beta test. To try it out, submit a ticket for application.
2. After your account is added to the allowlist, authorize the container instance as prompted in the TKE console. (If you have already authorized the container instance, skip this step.)

Step 3. Create a container instance

Log in to the TKE console and configure the container instance on the **Quick create** page.

Configuration Item	Description
Region	Select a closest region. For example, if you are located in Shenzhen, select "Guangzhou" for the region.
Container network	Assign an IP address within the IP range of the container network to the container instance. Note Subnet determines the availability zone. Each availability zone supports different type of resources, such as AMD, GPU-T4 and GPU-V100. Select a subnet which supports the desired type of resources according to the prompts.
Security Group	Security group has the capability of a firewall and can limit the network communication of

<table>
<tr><td></td><td>the instance. Default value is default.</td></tr>
<tr><td>Instance specification</td><td>For specifications supported by an instance, see Resource Specifications.</td></tr>
<tr><td>Image</td><td>You can select an image from TCR Enterprise Edition, TCR Personal Edition, Docker Hub, or other third-party image repositories.</td></tr>
<tr><td>Image Tag</td><td>If this parameter is left empty, `latest` will be used by default.</td></tr>
<tr><td>Image Repository Credential</td><td>If you select an image from a third-party image repository other than Dockerhub, you must enter the image credential, i.e., access address, username and password of the image repository.</td></tr>
<tr><td>Volume (optional)</td><td>Provides storage for the container. Currently, it supports NFS and CBS. Also, it needs to be mounted to the specified path of the container.
<table>
<tr><th>Volume Type</th><th>Description</th></tr>
<tr><td>Cloud Block Storage (CBS)</td><td>You can mount a Tencent Cloud CBS disk to a specified path of the container. When the container is migrated, the cloud disk will be migrated along with it. CBS volumes are suitable for the persistent storage of data and can be used for stateful services such as MySQL. For a service for which a CBS volume is configured, the maximum number of Pods is 1.</td></tr>
<tr><td>Network File System (NFS)</td><td>You only need to enter the NFS path. You can use a CFS or NFS for file storage. NFS volumes are suitable for the persistent storage of data that is read and written many times. They can also be used in scenarios such as big data analysis, media processing, and content management.</td></tr>
</table></td></tr>
<tr><td>Environment Variable (optional)</td><td>You can configure environment variables for containers.</td></tr>
<tr><td>Number of Instances (optional)</td><td>You can create multiple instances at a time. You can create only one copy if you select CBS as the volume type.</td></tr>
</table>

After configuring the required fields, confirm the resource specification and configuration fees, and click **Create Instance**. Then, you can view the created container instance.

Step 4. View container instance events

- Method 1
- Method 2

1. Log in to the TKE console.
2. On the container instance list page, click **More** > **View events** on the right of the instance for which you want to view the events.

Step 5. View container logs

- Method 1
- Method 2

1. Log in to the TKE console.
2. On the container instance list page, click **Logs** on the right of the instance for which you want to view the events.

Only the standard output logs of the container can be viewed here. For more information on the collection of standard output logs and container file logs, see Enabling Log Collection.

Examples

Creating Simple Nginx Service

Last updated：2023-07-07 17:39:12

Scenarios

This document describes how to quickly create an Nginx service in a container cluster.

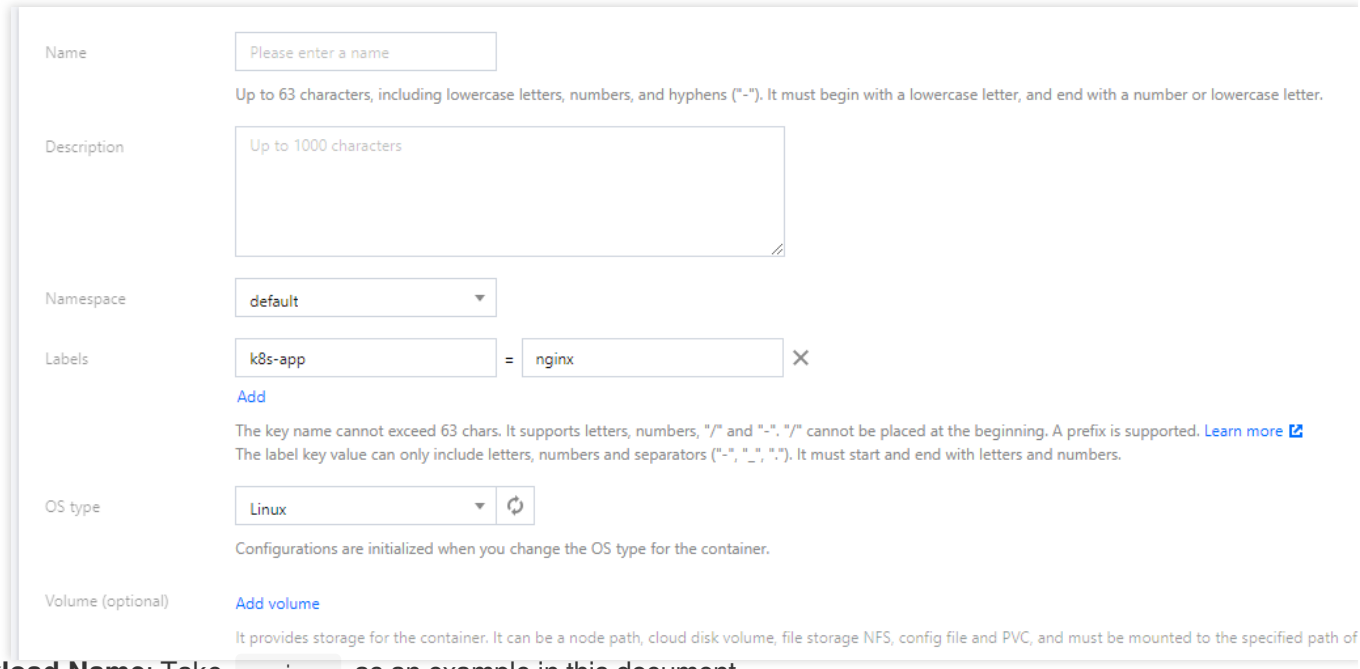
Prerequisites

You have registered a Tencent Cloud account.
You have created a cluster. For operation details, see Creating a Cluster.

Directions

Creating Nginx service

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, click the target cluster ID to enter the **Basic Information** page.
3. On the **Workload** > **Deployment** page, click **Create**. For more information, see Creating a Deployment.
4. On the **Create Deployment** page, specify basic information of the workload as instructed in the figure below.



Workload Name: Take `nginx` as an example in this document.

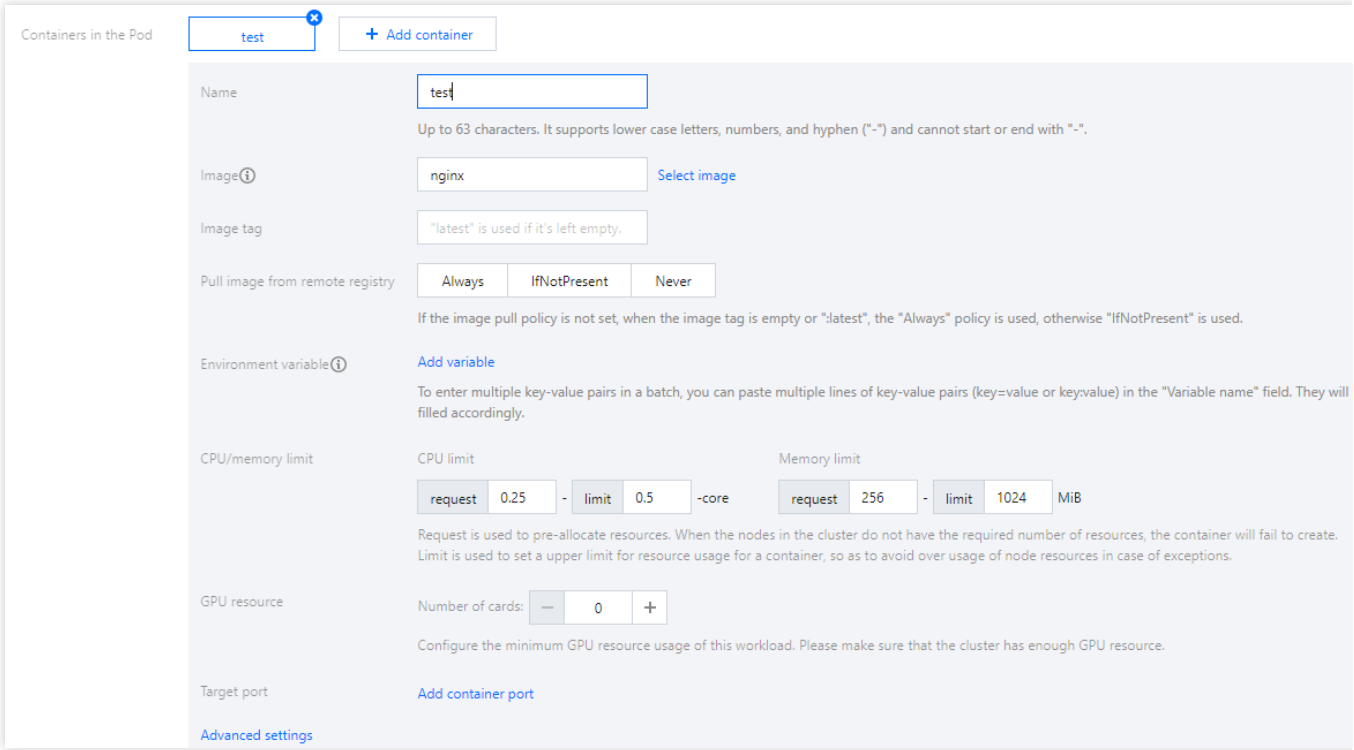
Description: Specify related workload information.

Label: In this example, the default value of the label is `k8s-app = nginx` .

Namespace: Select a namespace as needed. The default value is `default` .

Volume: set the volume to which your workload will be mounted based on your requirements. For more information, see Instructions for Other Storage Volumes.

5. Configure **Containers in the Pod** as instructed in the figure below.



The main parameters are described as follows:

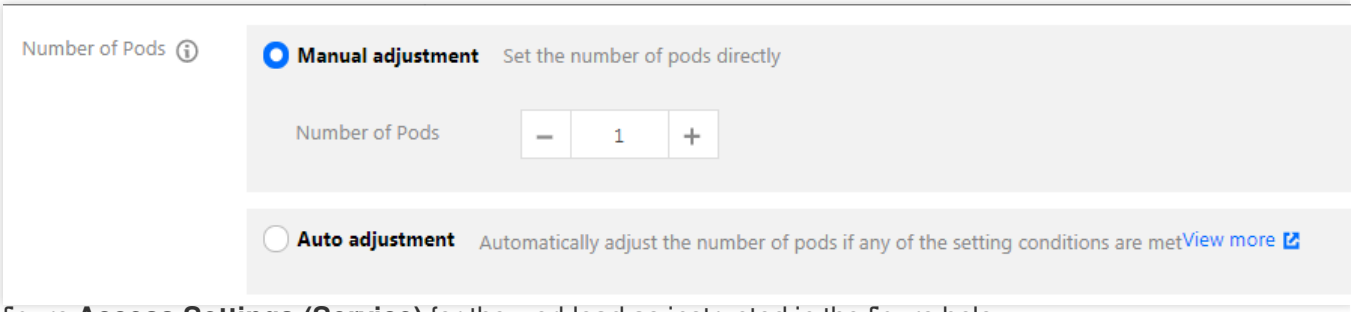
Name: Enter the name of the container in the pod. Here, “test” is used as an example.

Image: Click **Select Image**, select **DockerHub Image** > **Nginx** in the pop-up, and click **OK**.

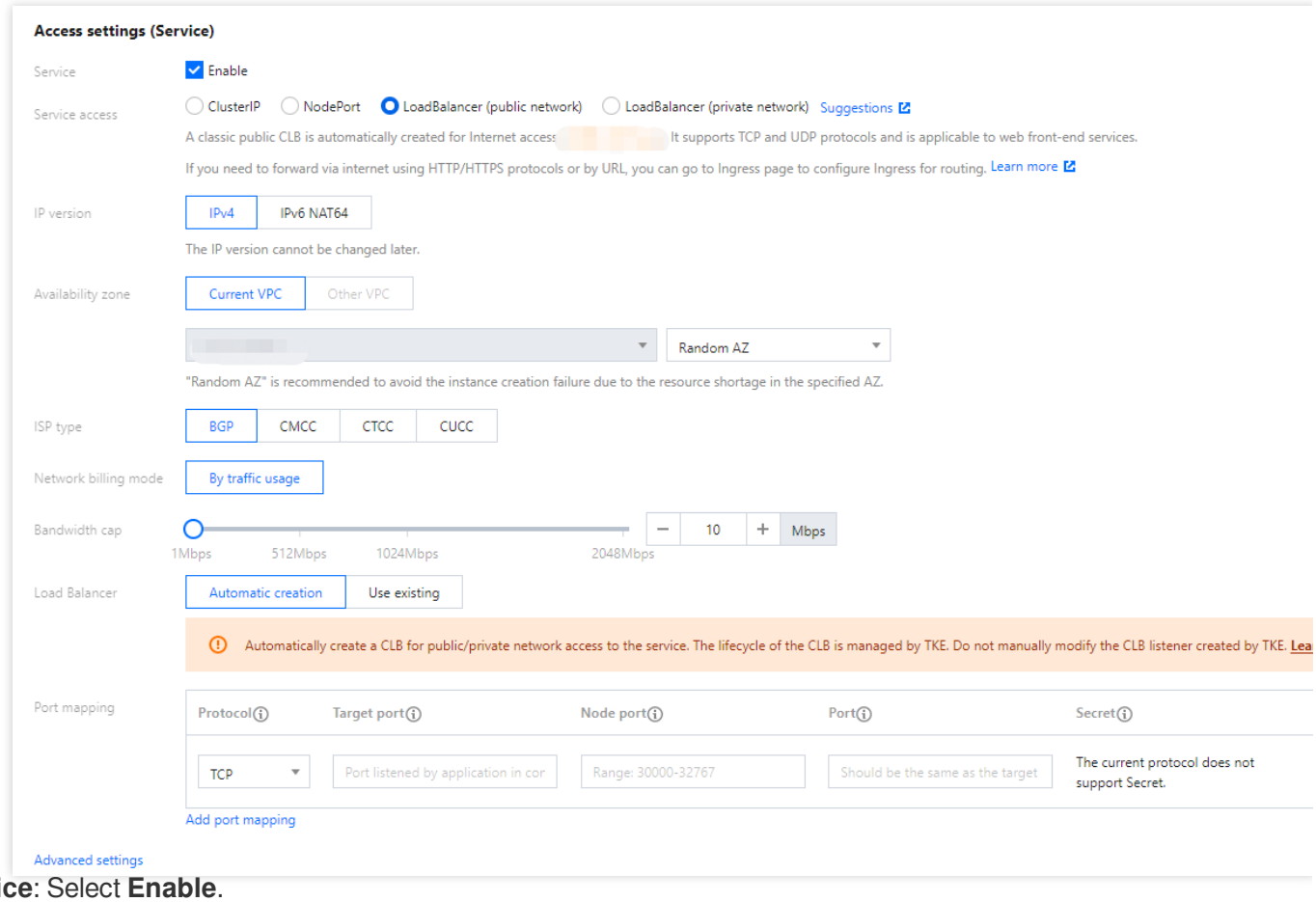
Image Tag: Use the default value `latest` .

Image Pull Policy: Choose from **Always**, **IfNotPresent** and **Never** as needed. In this document, we use the **default policy** as an example.

6. n **Number of Instances**, set the number of instances for the service according to the following information. In this document, we choose **Manual Adjustment** and set the instance number to one. See the figure below:



7. Configure **Access Settings (Service)** for the workload as instructed in the figure below.



Service: Select **Enable**.

Service Access: Select **LoadBalancer (public network)**.

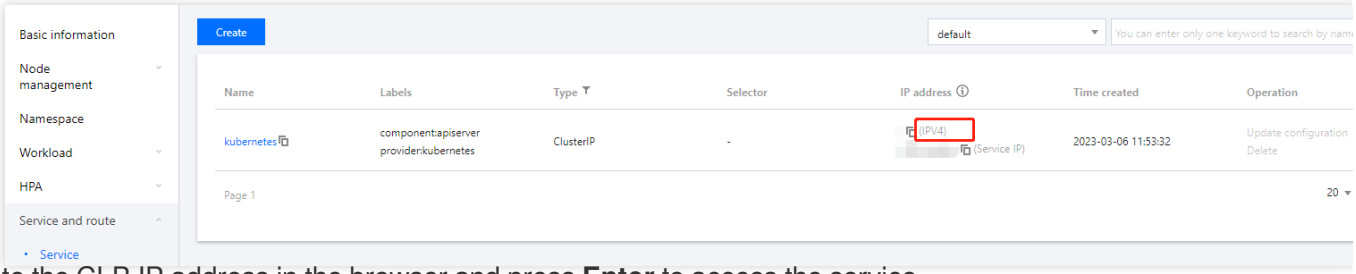
Load Balancer: Select according to your requirements.
Port Mapping: Select TCP, and set both the container port and service port to 80.
Protocol: Select the communication protocol as needed.
Target port: Set the port on which the application in the container listens. The port range is 1 to 65535.
Node Port：The service can be accessed via "CVM IP + host port". The port range is 30000 to 32767. A random port is assigned if it's left empty
Port: A created Service can be accessed from outside the cluster with the "CLB instance domain name or IP + Service port" or from within the cluster with the "Service name + Service port".
Secret: Select a value only when the TCP SSL protocol is used.
Notes
The node network, container network, and ports 30000 to 32768 need to be opened to the internet for the security group of the cluster to which the service belongs. Otherwise, the TKE may be unavailable. For more information, see TKE Security Group Settings.
8. Click **Create workload** to complete the creation of the Nginx service.

Accessing Nginx service

Nginx service can be accessed using the following two methods.

Accessing Nginx service using Cloud Load Balancer IP

1. In the left sidebar, click Cluster to go to the **Cluster Management** page.
2. Click the ID of the cluster to which the Nginx service belongs and select **Services and Routes** > **Service**.
3. On the service list page, copy the CLB IP of the Nginx service as shown below:



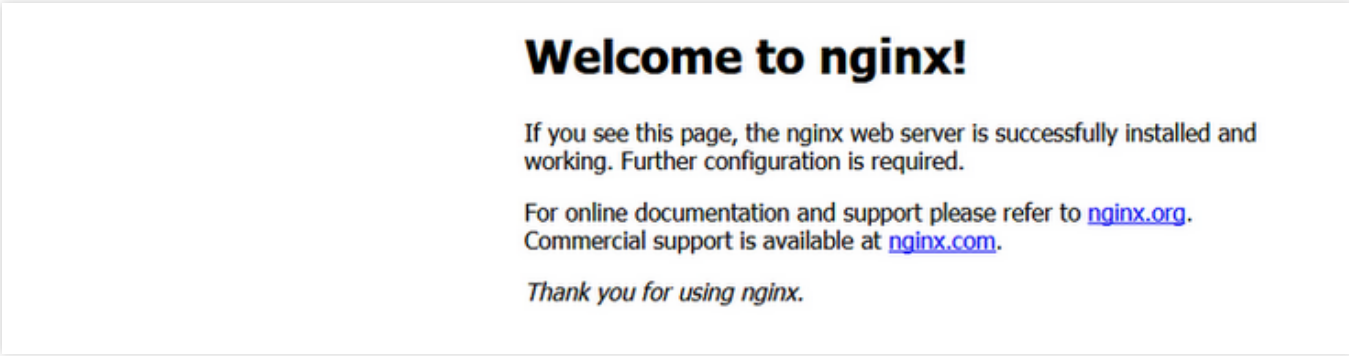
4. Paste the CLB IP address in the browser and press **Enter** to access the service.

Accessing Nginx service using service name

Other services or containers in the cluster can access the WordPress service using the service name.

Verifying Nginx service

When the service is successfully created, you directly enter the Nginx server welcome page when accessing the service. See the figure below:



More Nginx settings

If the container cannot be created, see Event FAQs for a solution.

Building Hello World Service Manually

Last updated：2023-09-26 15:52:20

Scenarios

This document describes how to create a Node.js service, Hello World, in a container cluster. For more information on how to build a Docker image, see How to Build a Docker Image.

Prerequisites

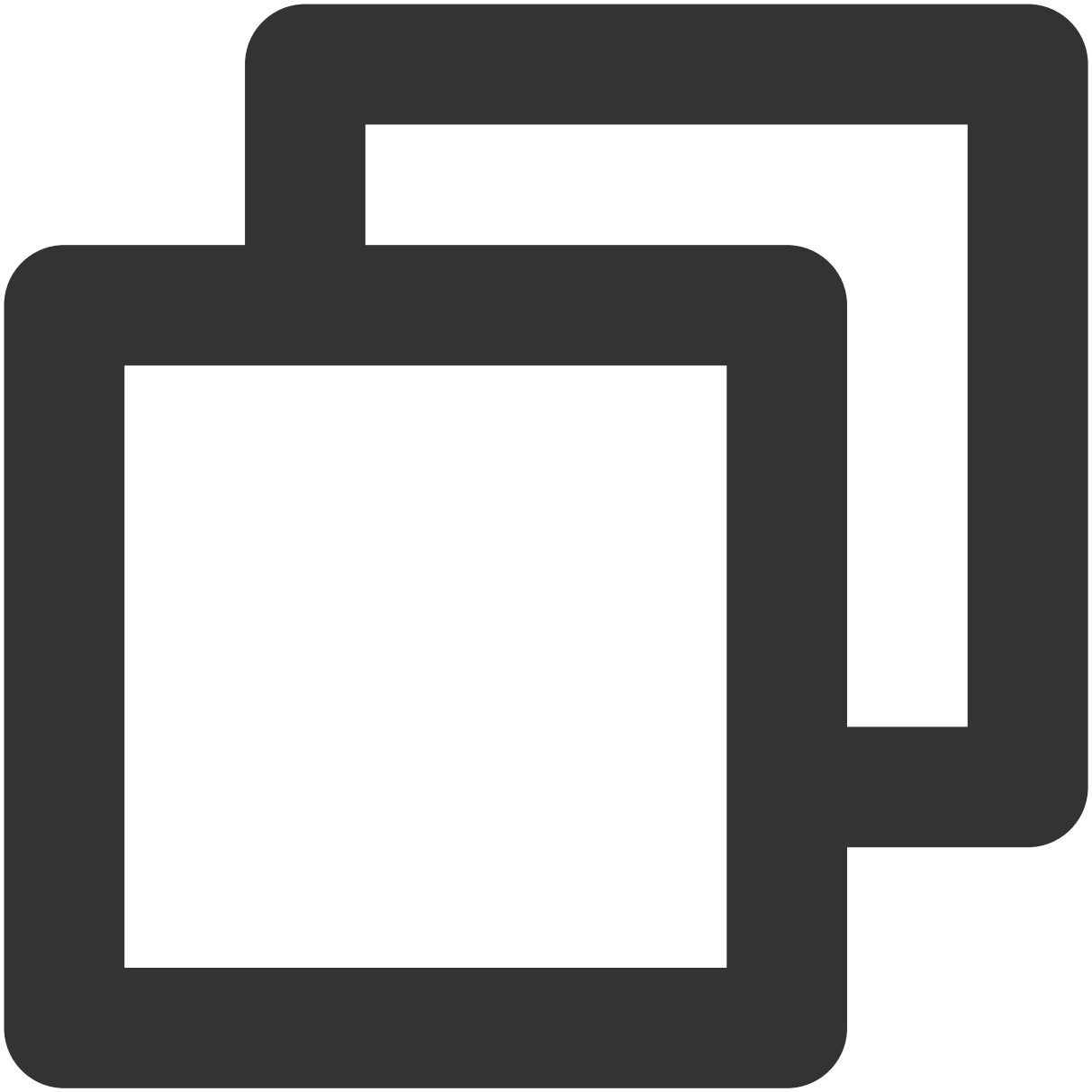
Create a cluster as instructed in Creating a Cluster.
Log in to a node with Node.js installed.

Directions

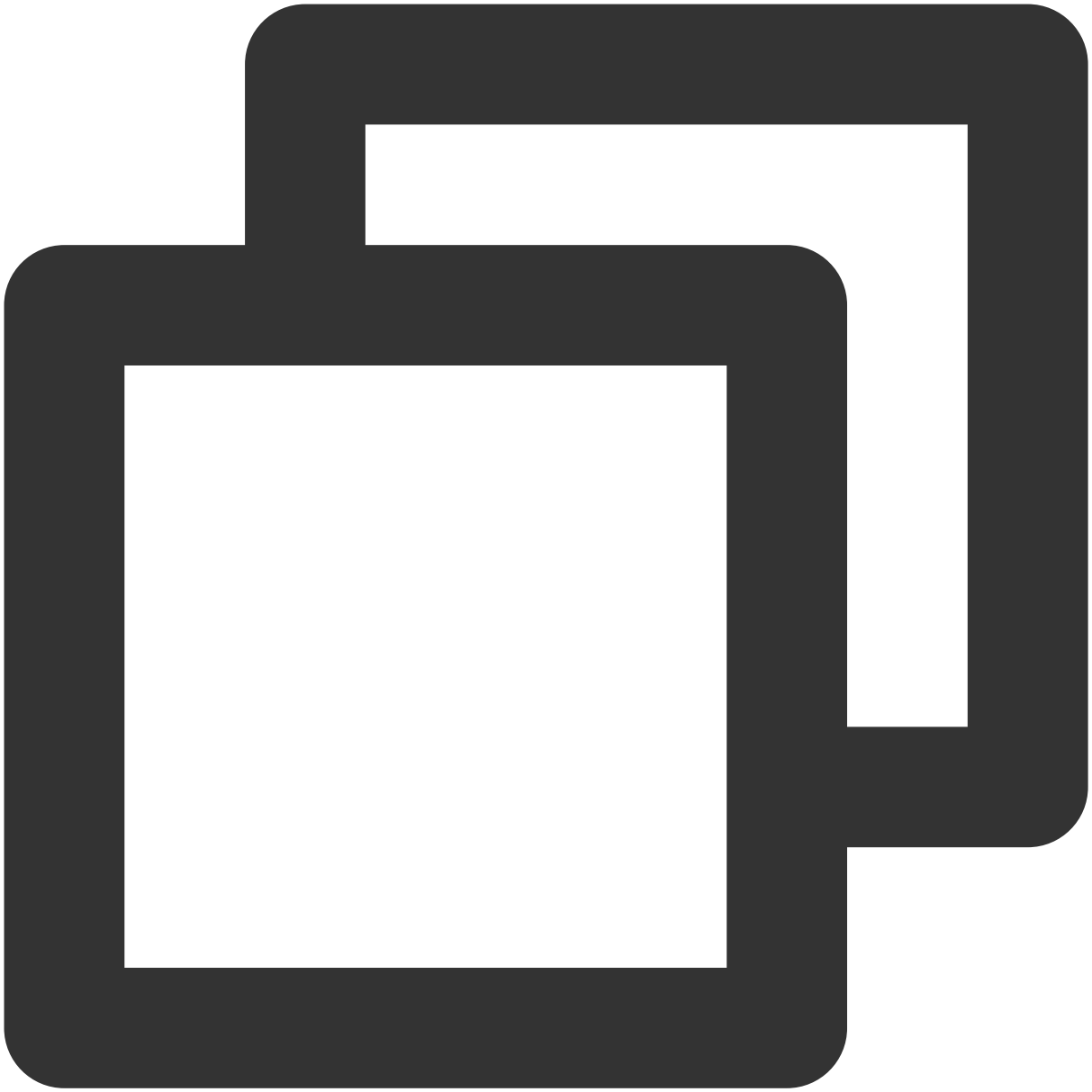
Writing code to create an image

Writing an application

1. Run the following commands in sequence to create and go to the `hellonode` directory:

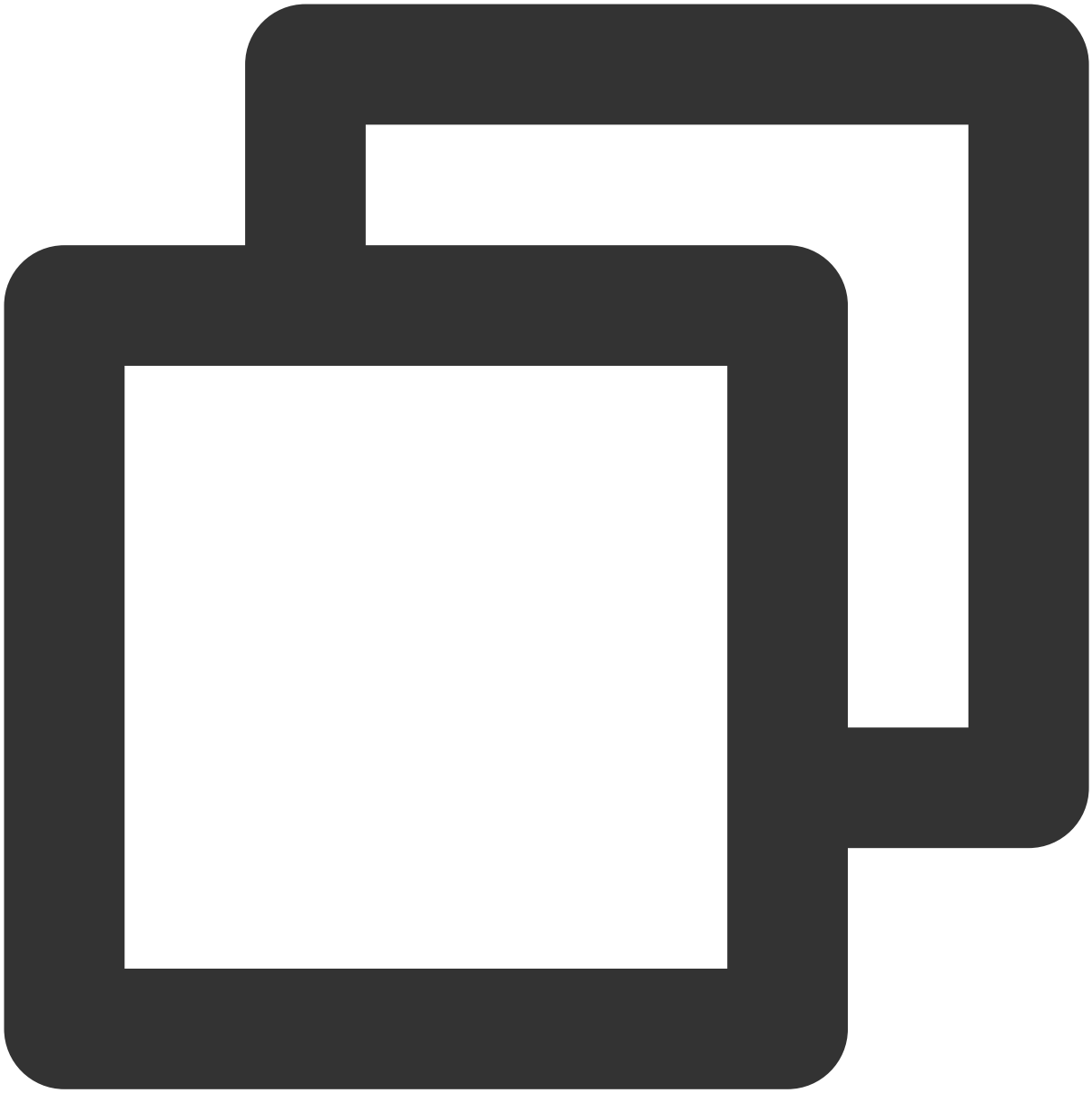


```
mkdir hellonode
```



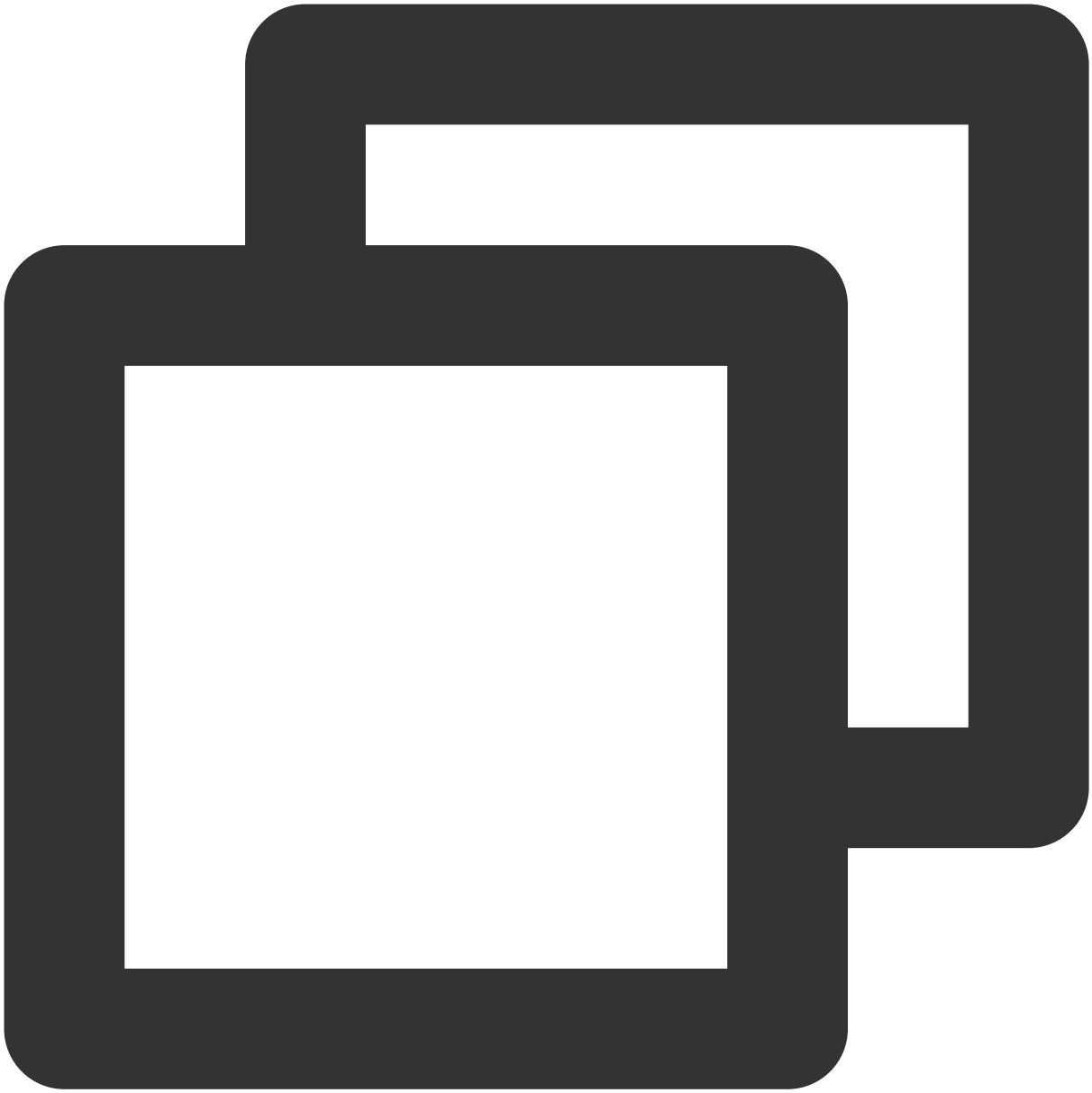
```
cd hellonode/
```

2. Run the following command to create and open the `server.js` file:



```
vim server.js
```

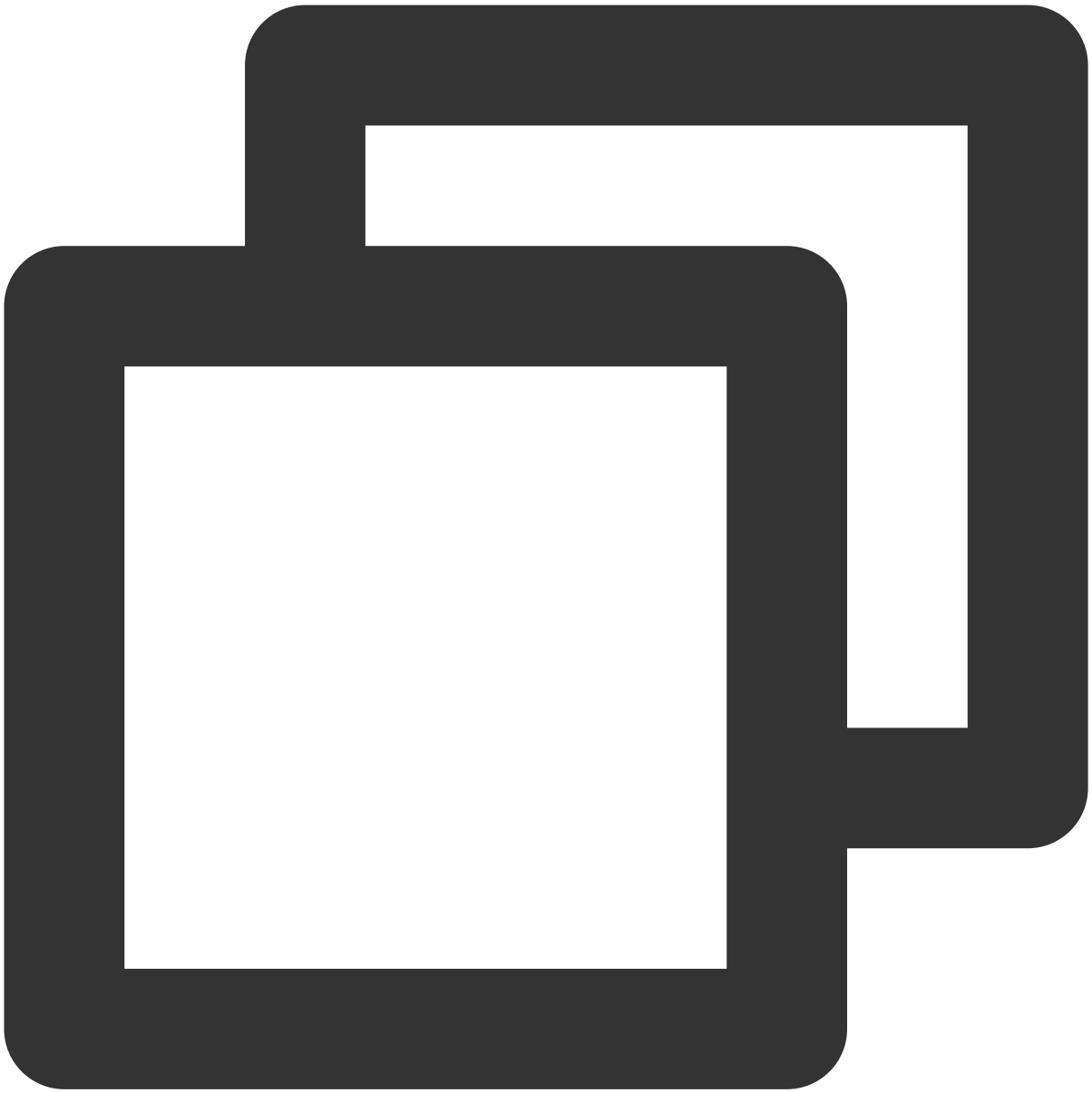
3. Press **i** to switch to the editing mode, and enter the following content in `server.js` :



```
var http = require('http');
var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('Hello World!');
};
var www = http.createServer(handleRequest);
www.listen(80);
```

Press **Esc** and enter `:wq` to save the file and return.

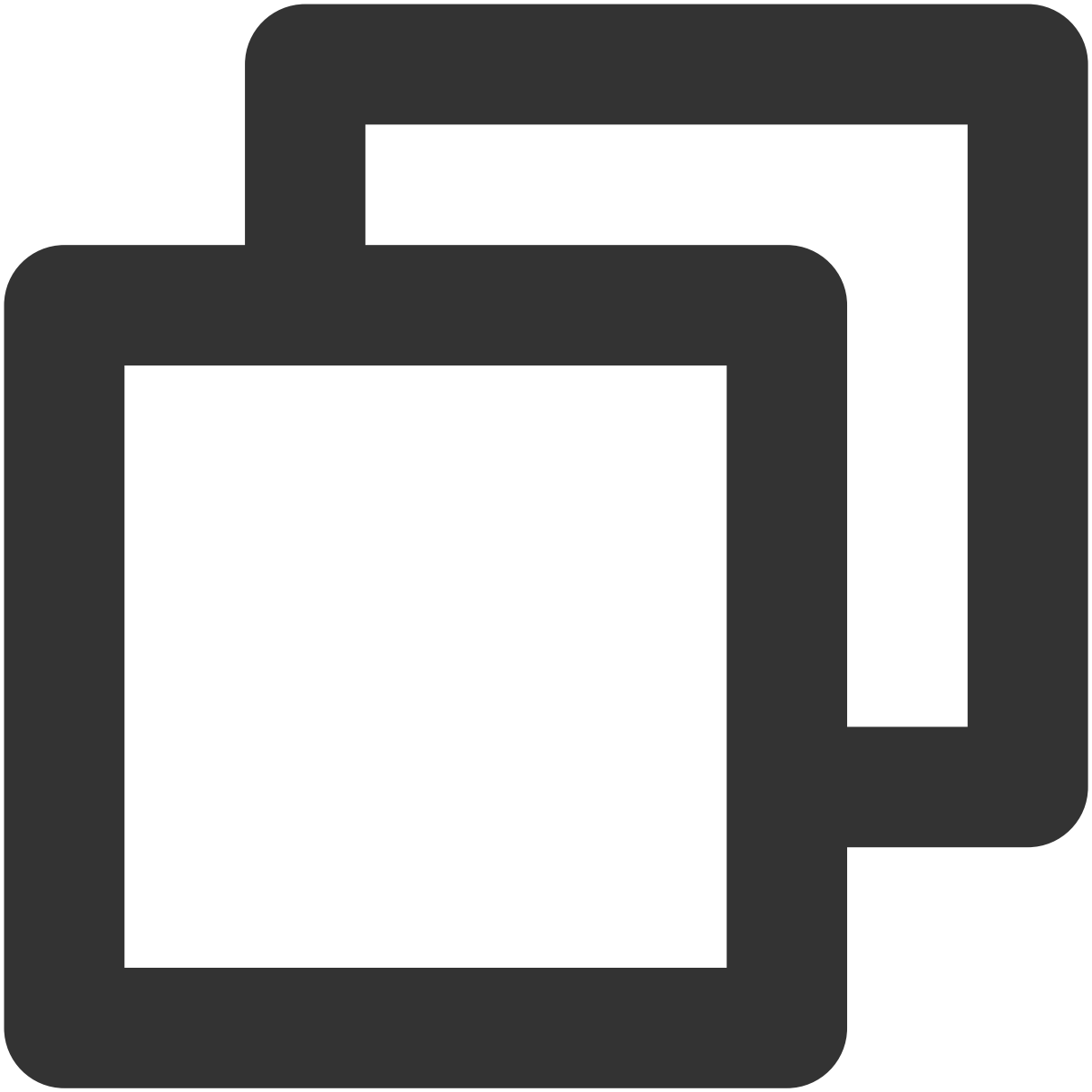
4. Run the following command to execute the `server.js` file:



```
node server.js
```

5. Test the Hello World program.

Method 1. Log in to the node again and run the following command:



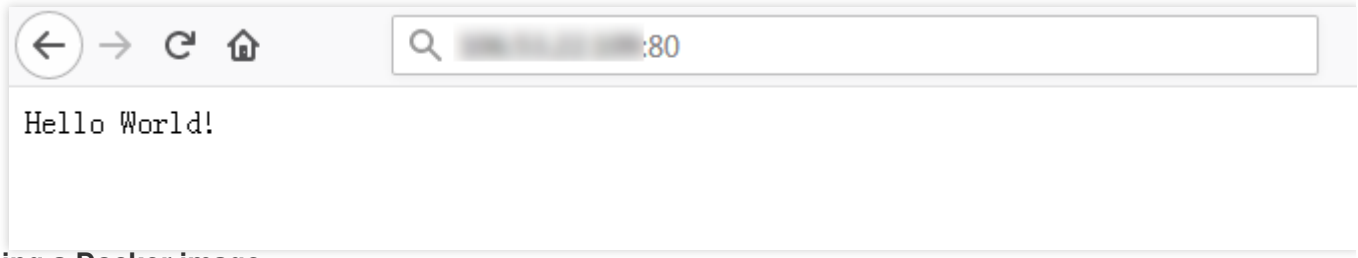
```
curl 127.0.0.1:80
```

If the following information appears, the Hello World program is running successfully.

```
[root@VM_2_5_centos ~]# curl 127.0.0.1:80
Hello World![root@VM_2_5_centos ~]#
```

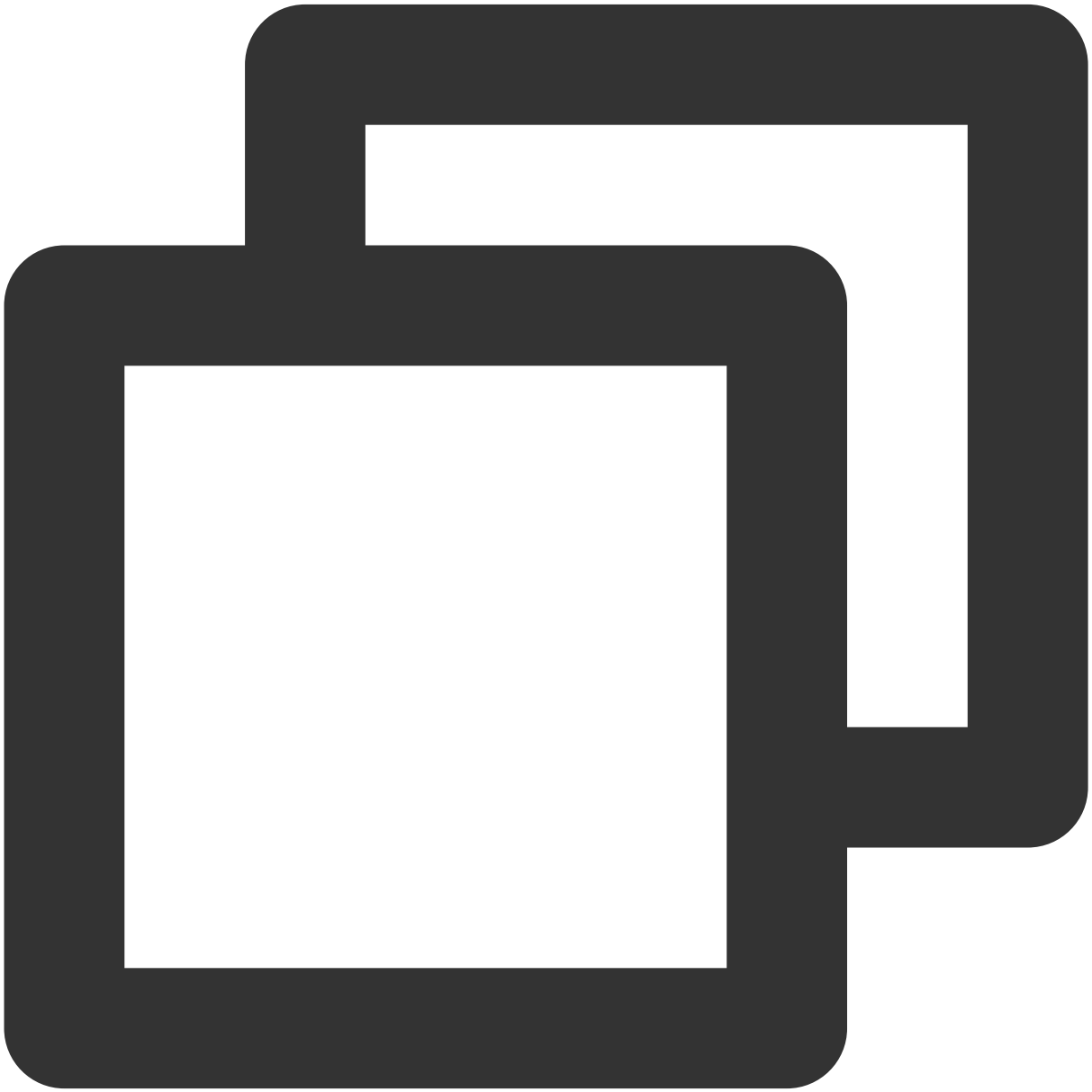
Method 2. Open a local browser and access the program via “[CVM public IP]:[Port]”. Only port 80 is supported.

If the following information appears, the Hello World program is running successfully.

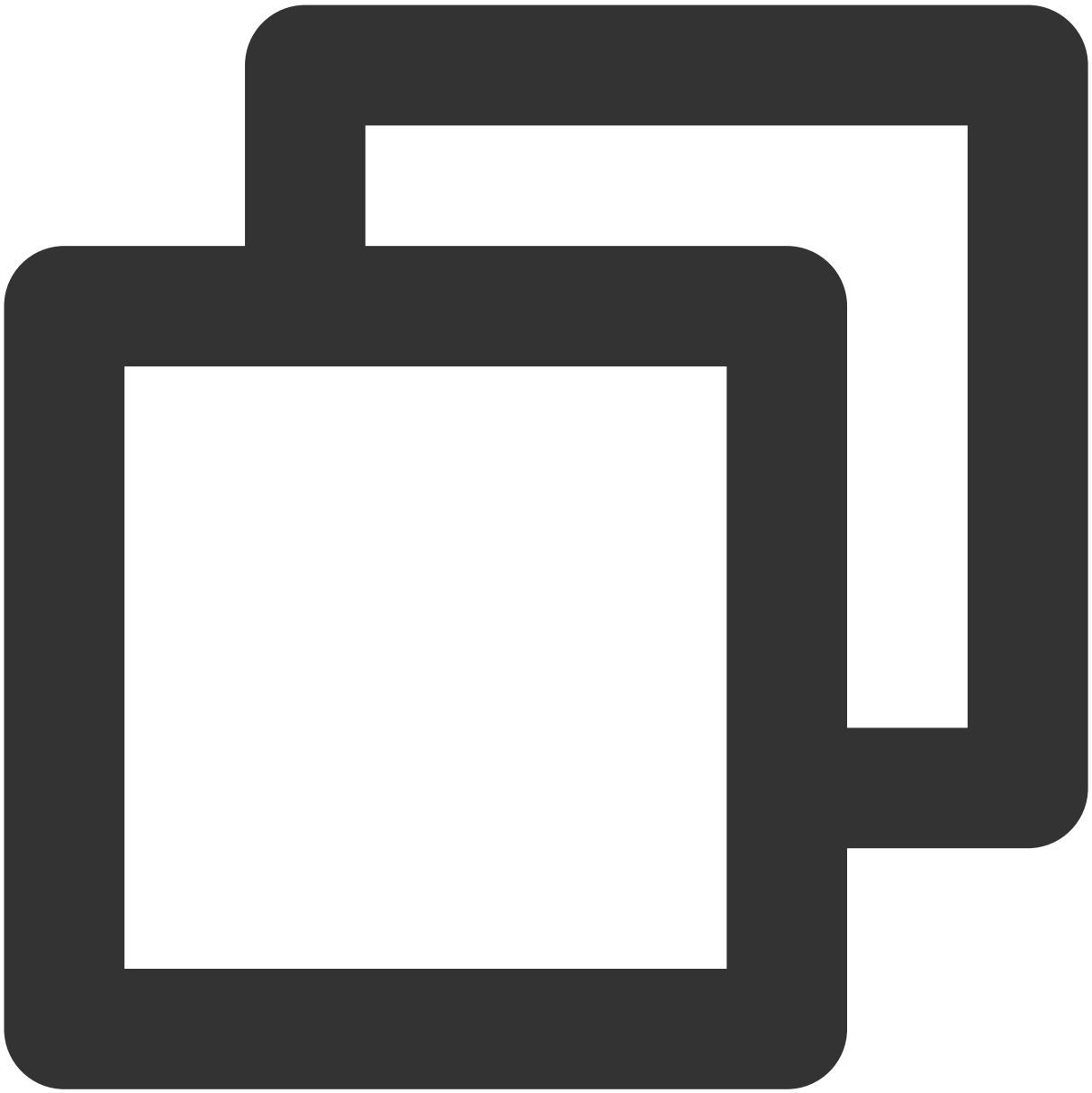


Creating a Docker image

1. Run the following commands in sequence to create a `Dockerfile` file in the `hellonode` directory:

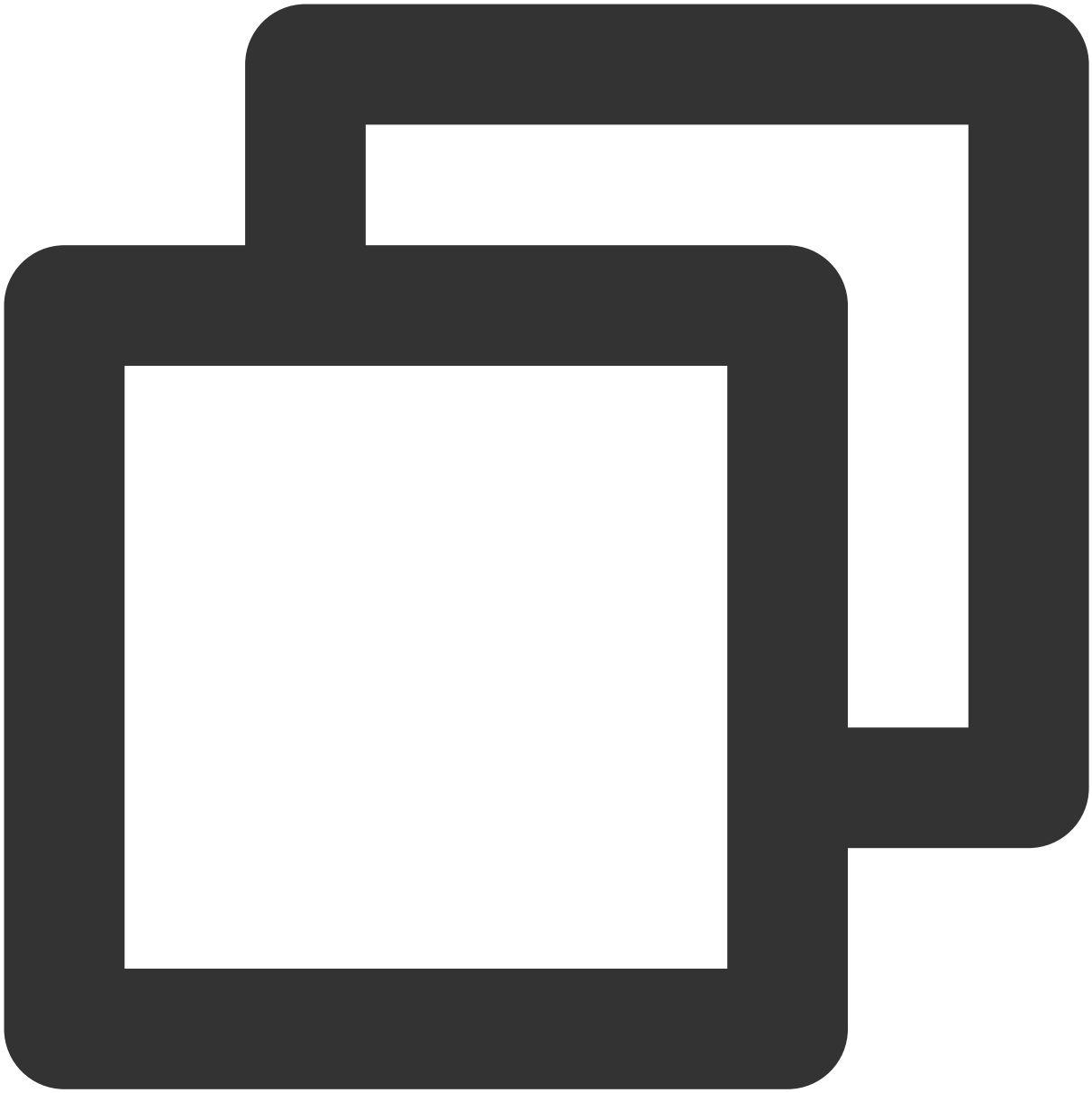


```
cd hellonode
```



```
vim Dockerfile
```

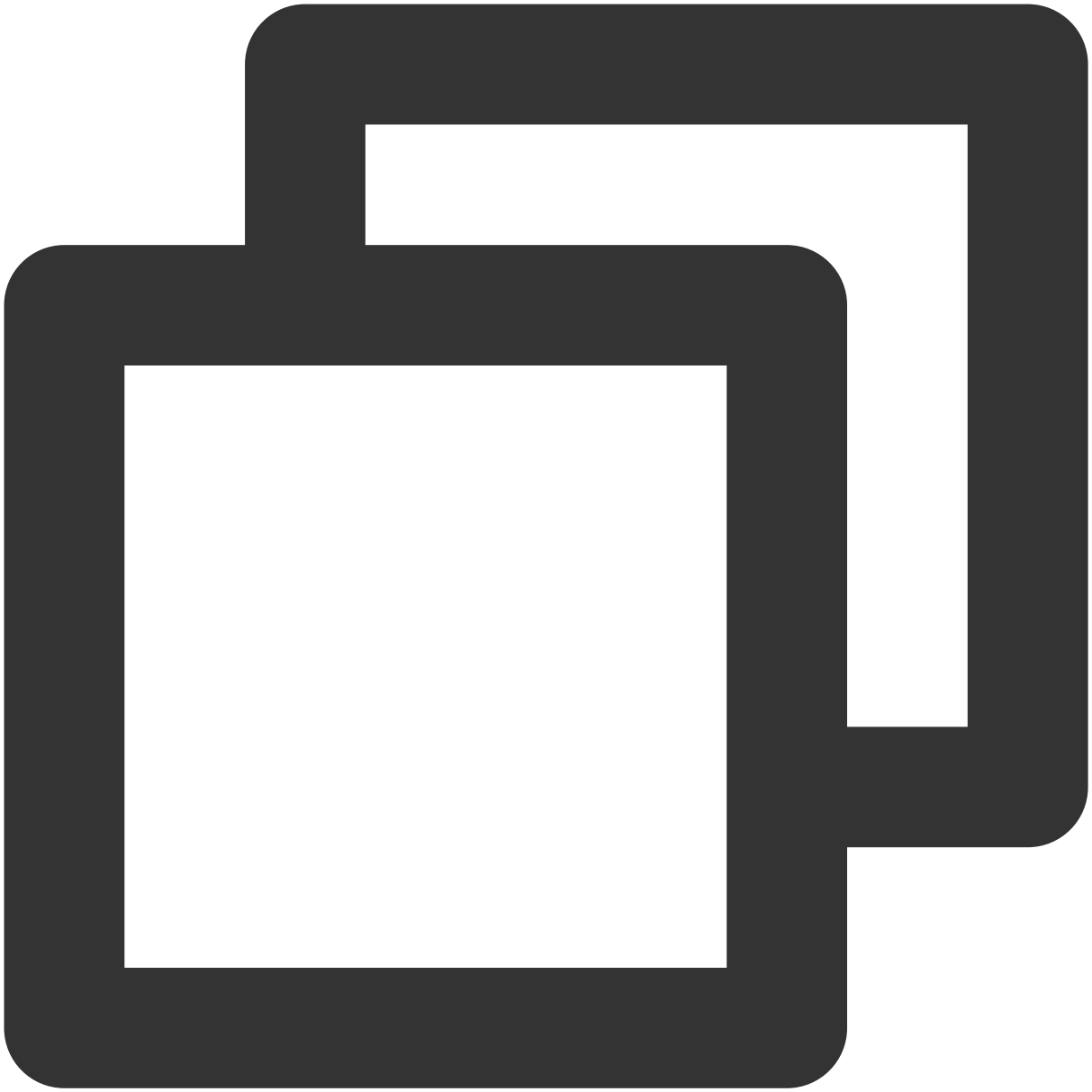
2. Press **i** to switch to the editing mode, and enter the following content in the `Dockerfile` file:



```
FROM node:4.4
EXPOSE 80
COPY server.js .
CMD node server.js
```

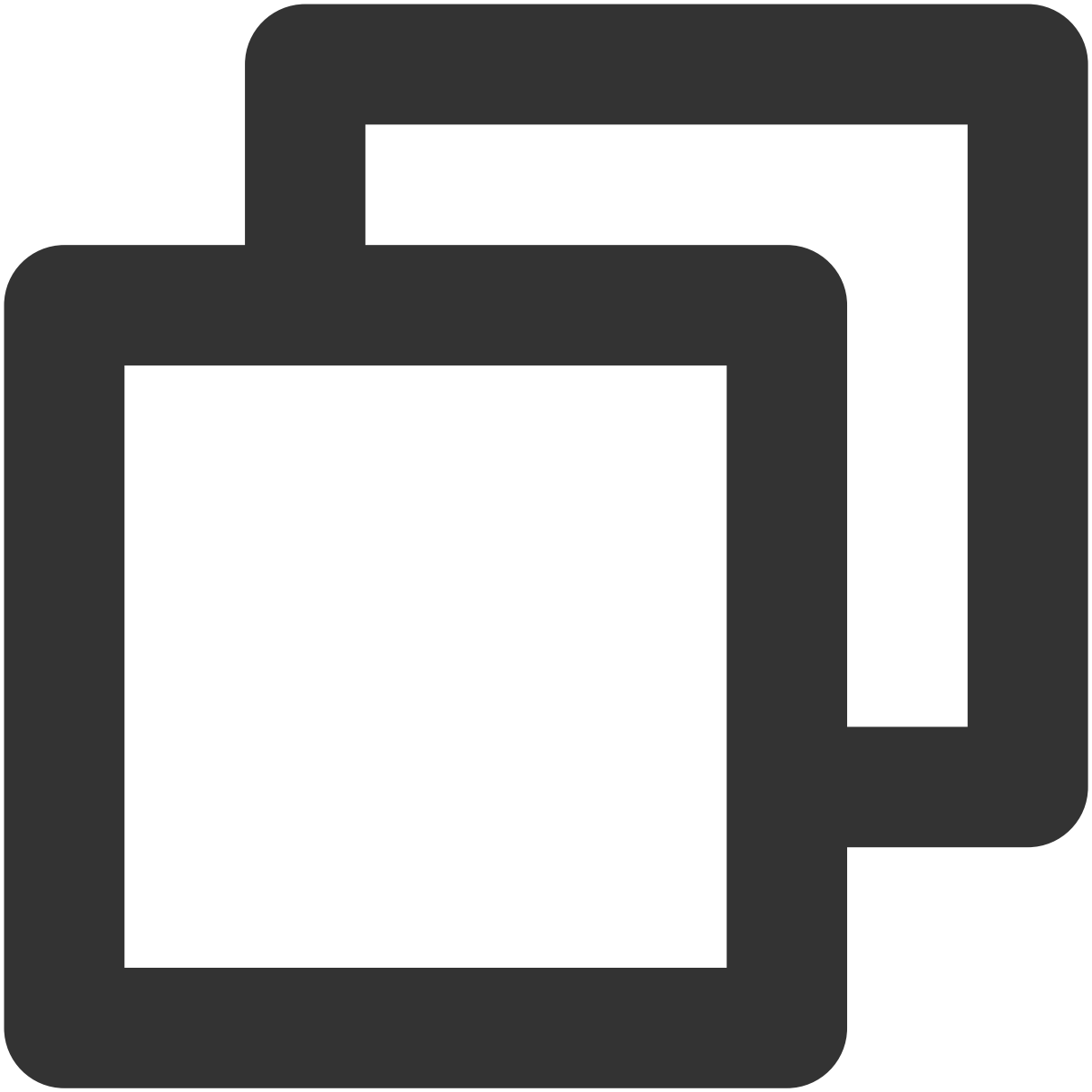
Press **Esc** and enter `:wq` to save the file and return.

3. Install and start Docker on the node.



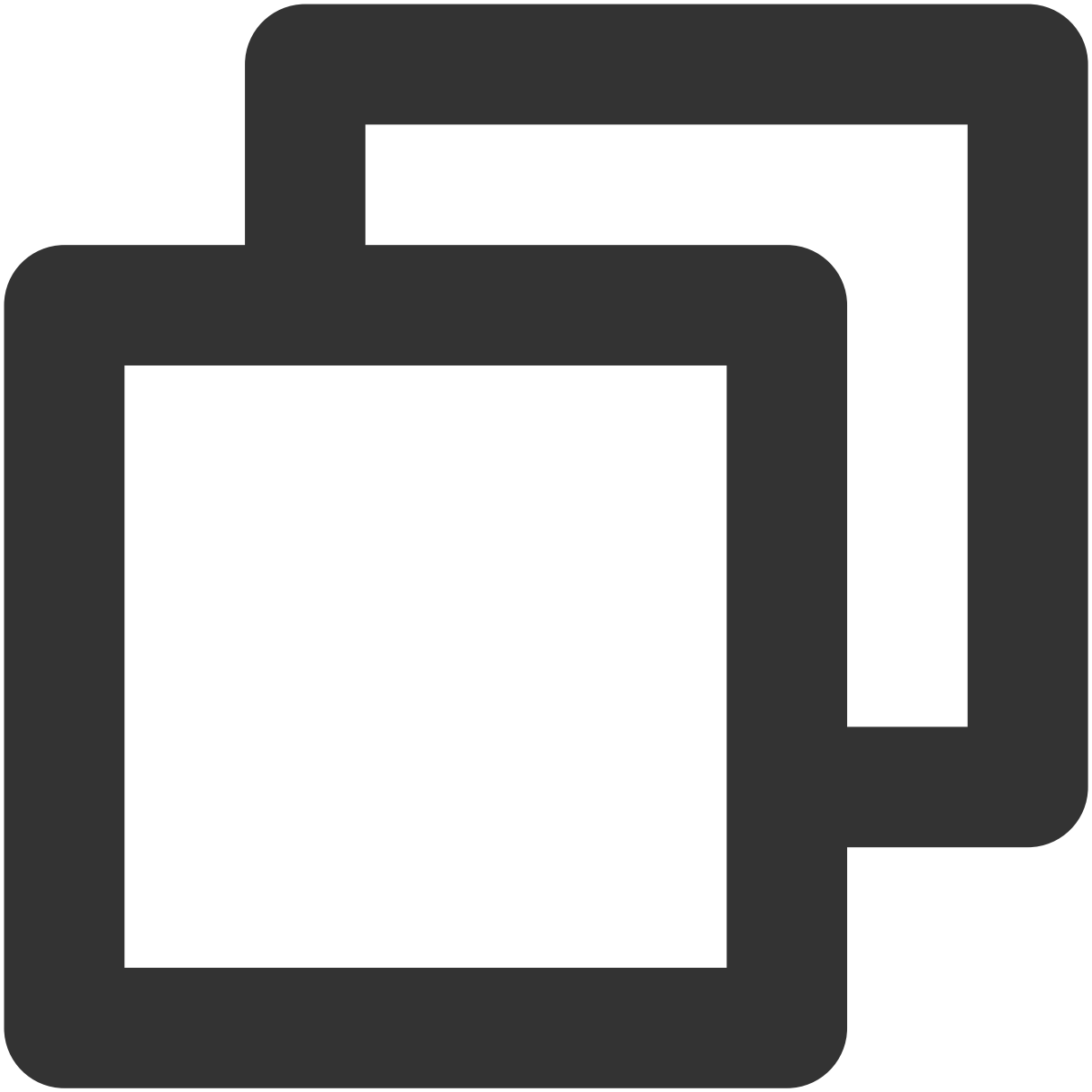
```
yum install -y docker
systemctl start docker
```

4. Run the following command to build an image:



```
docker build -t hello-node:v1 .
```

5. Run the following command to check the built `hello-node` image:

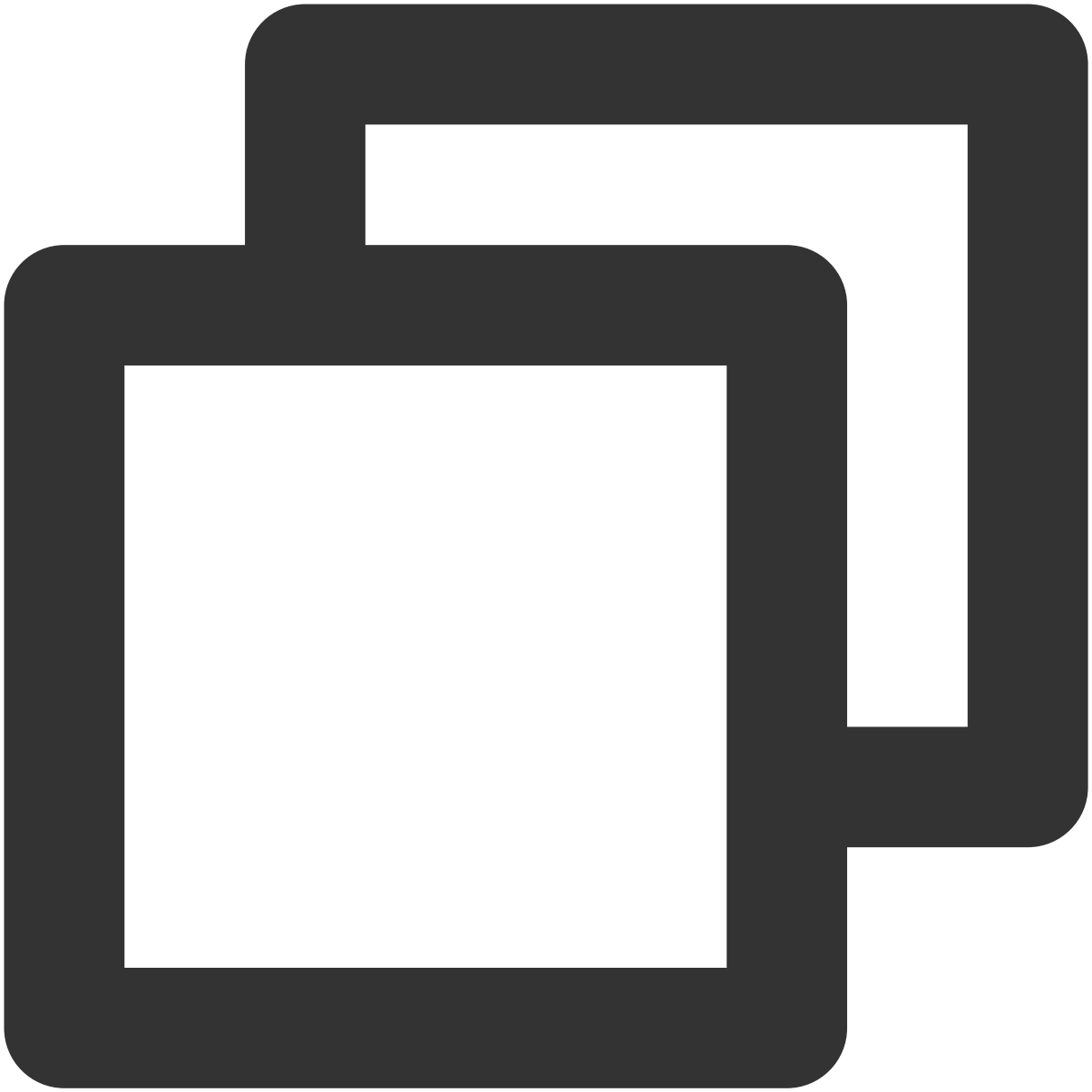


```
docker images
```

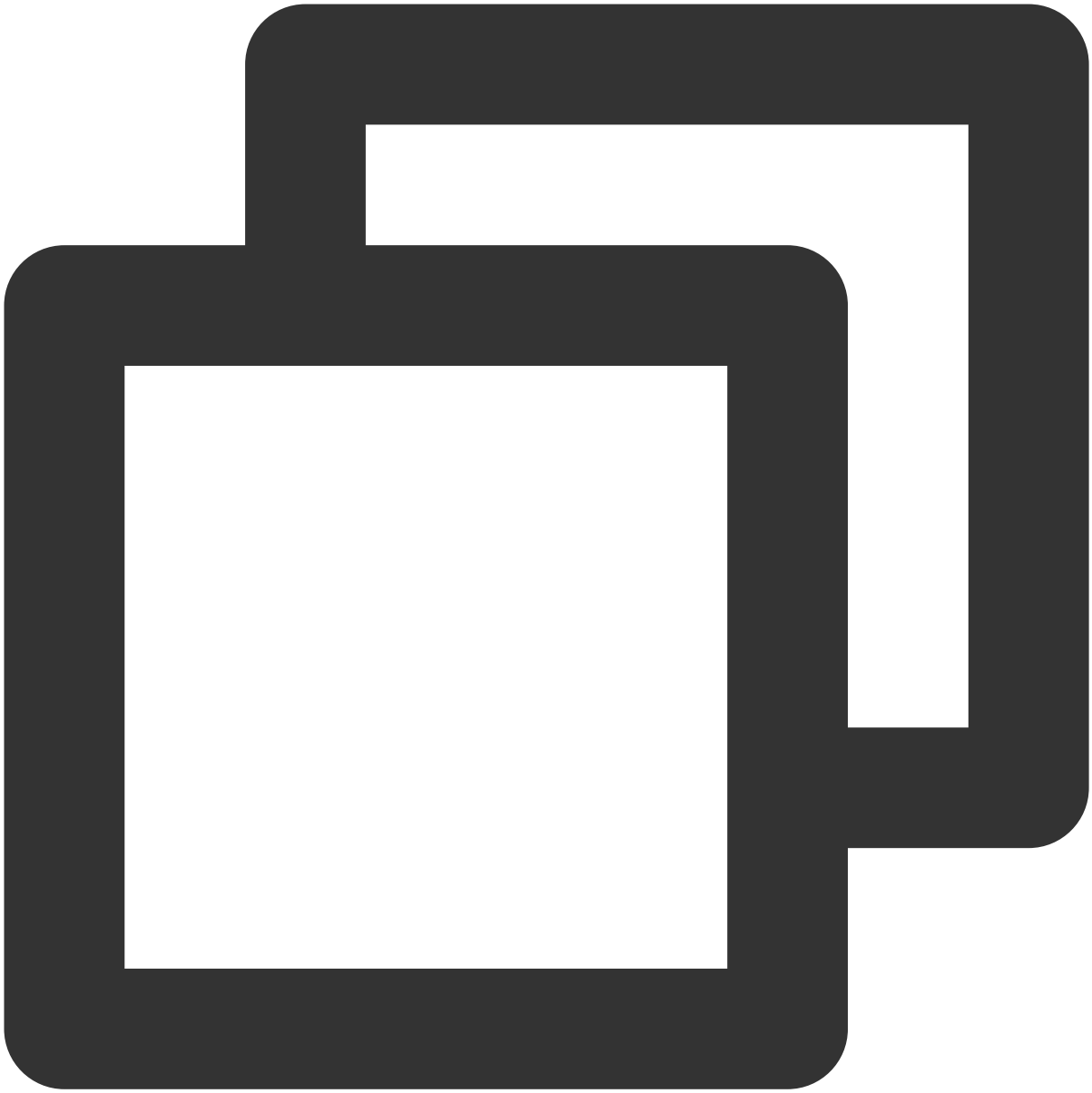
If the following information appears, the hello-node image is successfully built. Take note of the IMAGE ID. See the figure below.

Uploading the image to Tencent Cloud image registry

Run the following commands in sequence to upload the image to the Tencent Cloud image registry.



```
docker tag IMAGEID ccr.ccs.tencentyun.com/Namespace/hello-node:v1
```



```
docker login ccr.ccs.tencentyun.com
docker push ccr.ccs.tencentyun.com/Namespace/hello-node:v1
```

Note

Replace the image ID in the command with the image ID noted down in Step 4.

Replace the namespace in the command with the namespace that you create. If you haven't created a namespace, create one first by referring to Step 4: Creating a Namespace.

If the following information appears, the image is successfully uploaded.

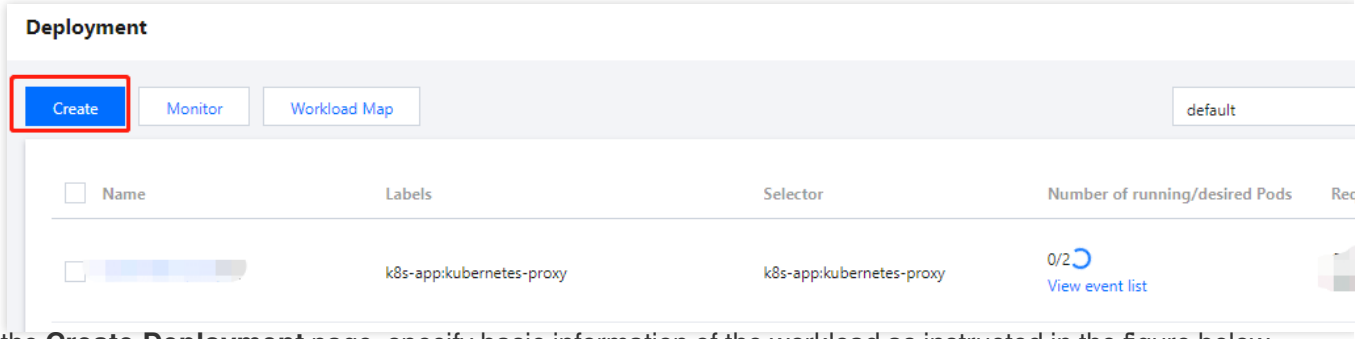
```
[root@VM_2_5_centos hellonode]# sudo docker tag [redacted] ccr.ccs.tencentyun.com/[redacted]
[root@VM_2_5_centos hellonode]# sudo docker push ccr.ccs.tencentyun.com/[redacted]test/hello[redacted]
The push refers to repository [ccr.ccs.tencentyun.com/[redacted]test/helloworld]
7357e3a21b1f: Pushed
20a6f9d228c0: Pushed
80c332ac5101: Pushed
04dc8c446a38: Pushed
1050aff7cfff: Pushed
66d8e5ee400c: Pushed
2f71b45e4e25: Pushed
v1: digest: sha256:9c139ecbb29c49f25e02d7906b9e78c6e2e274827a75603ef48fa5547ff8a620 s[redacted]
```

Creating the Hello World service using the image

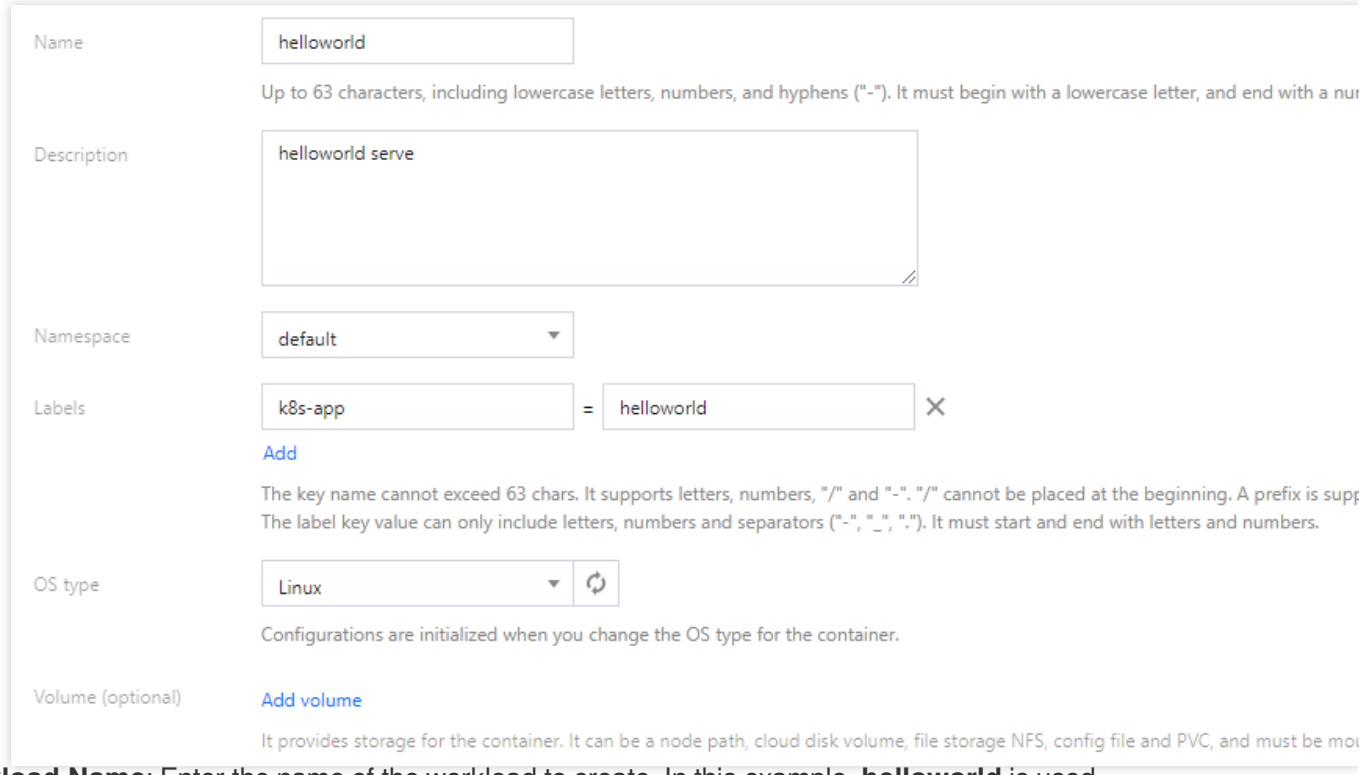
Note

Before creating and using the Hello World service, you must have a cluster. If you do not have a cluster, create one by referring to Creating a Cluster.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. Select **Workload** > **Deployment**. On the **Deployment** page, click **Create**.



4. On the **Create Deployment** page, specify basic information of the workload as instructed in the figure below.



Workload Name: Enter the name of the workload to create. In this example, **helloworld** is used.

Description: Specify related workload information.

Namespace: Select a namespace based on your requirements.

Tag: Specify the key-value pair. The default value is **k8s-app = helloworld** here.

Type: Select a type as required. **Linux** is selected in this example.

Volume: Set up the workload volumes mounted based on your requirements. For more details, see Volume Management.

5. Configure **Containers in Pod** as instructed.

5.1 Enter the name of the container. In this example, `helloworld` is used.

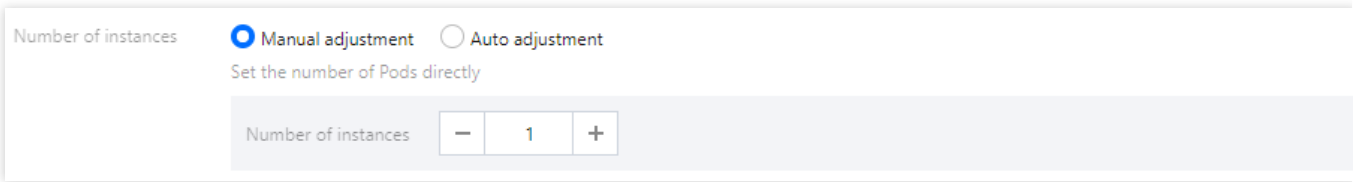
5.2 Click **Select an image**, and click **My Images**in the dialog box that appears. Use the search box to find the helloworld image, and then click **OK**.

The main parameters are described as follows:

Image Tag: Use the default value **latest**.

Image Pull Policy: Choose from **Always**, **IfNotPresent** and **Never** as needed. In this document, we use the default policy.

6. In the **Number of Pods** section, set the number of pods for the service as instructed. See the figure below.



Manual adjustment: Set the number of pods. The number of pods in this example is set to 1. You can click "+" or "-" to change the number of pods.

Auto adjustment: The system automatically adjusts the number of pods if any specified condition is met. For more information, see Automatic Scaling Basic Operations.

7. Configure **Access Settings (Service)** for the workload as instructed below.

Service: Select **Enable**.

Service Access: Select **LoadBalancer (public network)**.

Load Balancer: Select according to your requirements.

Port Mapping: Select TCP, and set both the container port and service port to 80.

Note

The node network, container network, and ports 30000 to 32768 need to be opened to the internet for the security group of the cluster to which the service belongs. Otherwise, the TKE may be unavailable. For more information, see TKE Security Group Settings.

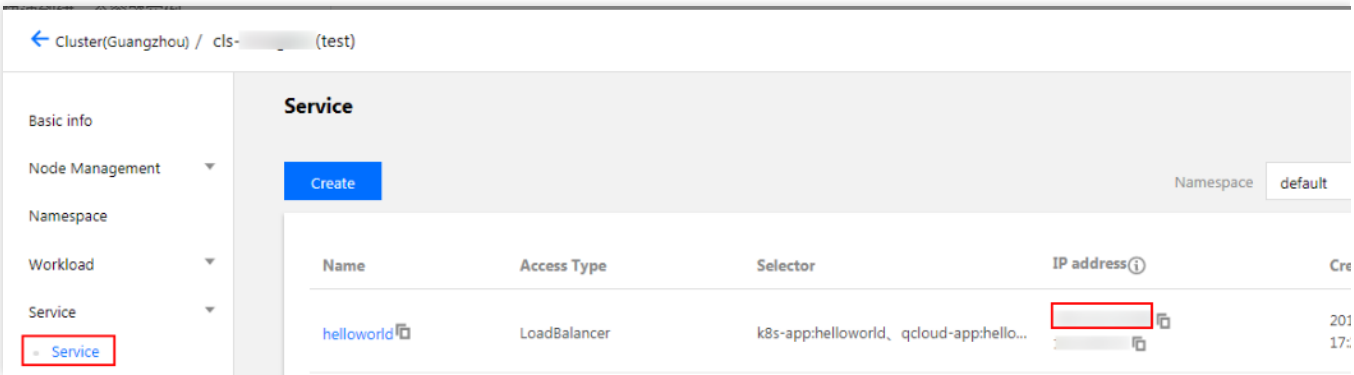
8. Click **Create Deployment** to create the Hello World service.

Accessing the Hello World service

The HelloWorld service can be accessed in either of the following ways.

Access using the CLB IP address

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the cluster to which the Hello World service belongs to go to the cluster details page.
3. Select **Service and route** > **Service** to go to the **Service** page.
4. On the service management page, copy the CLB IP address of the Hello World service, as shown in the following figure:



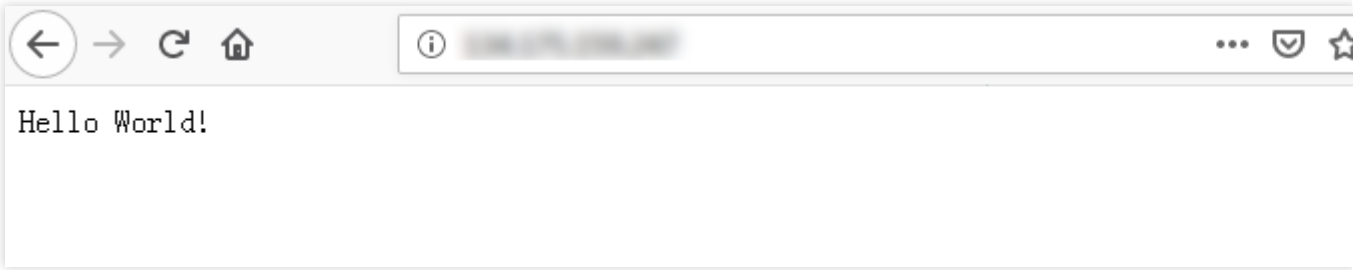
5. Paste the CLB IP address for the Hello World service in your browser.

Access using the service name

Other services or containers in the cluster can access the WordPress service using the **service name**.

Verifying the Hello World service

If the following information appears when you access the service, the Hello World service is successfully created.



If the container cannot be created, see Event FAQs for a solution.

WordPress with Single Pod

Last updated：2023-02-02 17:05:22

Overview

WordPress is a blogging platform developed with PHP. You can use it as a content management system, or use it to create websites on services that support PHP and MySQL databases.

This document describes how to use the official `wordpress` image on Docker Hub to create a publicly accessible WordPress website.

Prerequisites

Note：

- The `wordpress` image contains all operating environments for WordPress, allowing you to pull and create the service directly.
- WordPress with a single Pod is used for testing purposes only, and therefore cannot ensure persistent data storage. It is recommended that you use a self-built MySQL or TencentDB to store your data. For more information, see WordPress Using TencentDB.

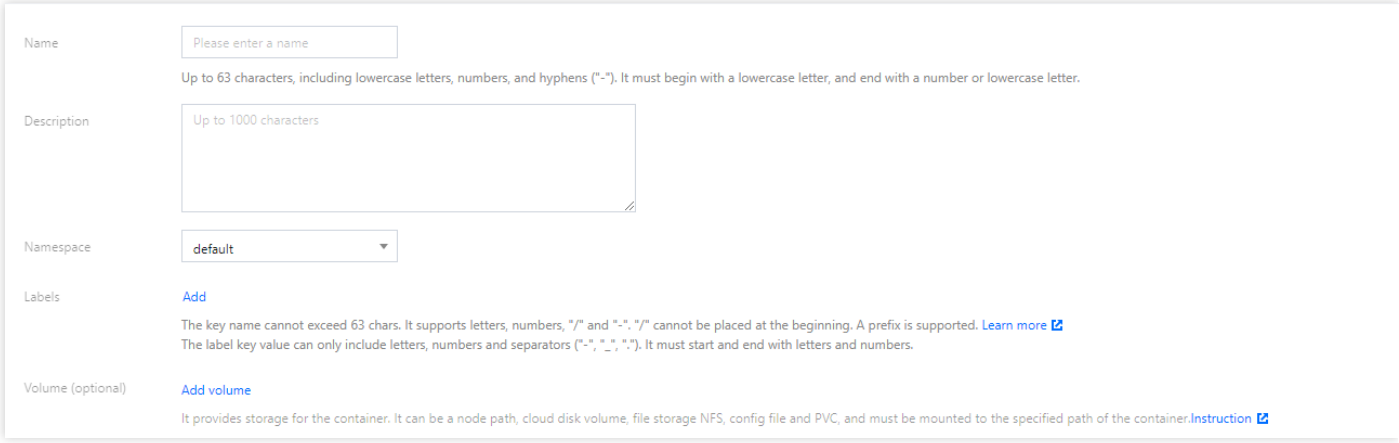
- You have registered a Tencent Cloud account.
- You have created a standard TKE cluster. For more information, see Creating a Cluster.

Directions

Creating a WordPress service

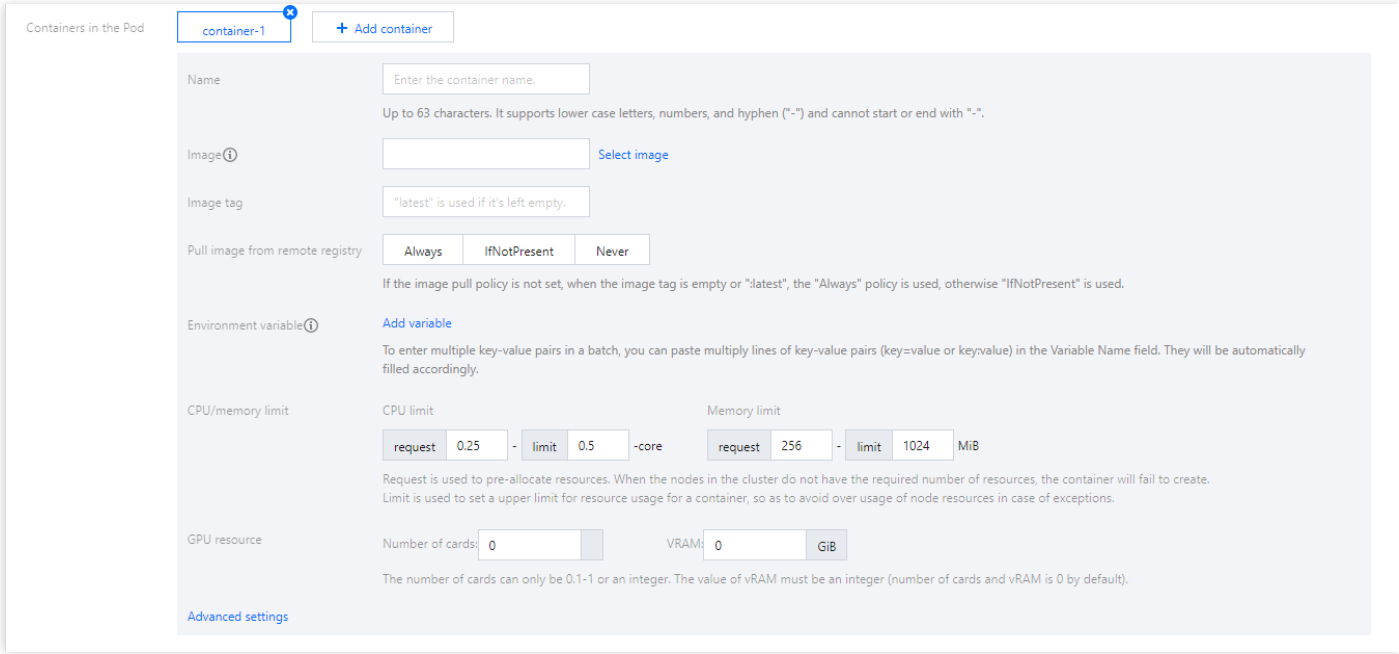
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the **Workload** > **Deployment** page, click **Create**. For more information, see Creating a Deployment.

4. On the **Create Deployment** page, specify basic information of the workload as instructed in the figure below.



- **Workload Name**: enter the name of the workload to create. In this example, `wordpress` is used.
- **Description**: specify related workload information.
- **Labels**: the default value is `k8s-app = wordpress` in this example.
- **Namespace**: select a namespace based on your requirements.
- **Volume**: set the volume to which the workload is mounted based on your requirements. For more information, see Volume Management.

5. Configure "Containers in the Pod" as instructed. See the figure below:

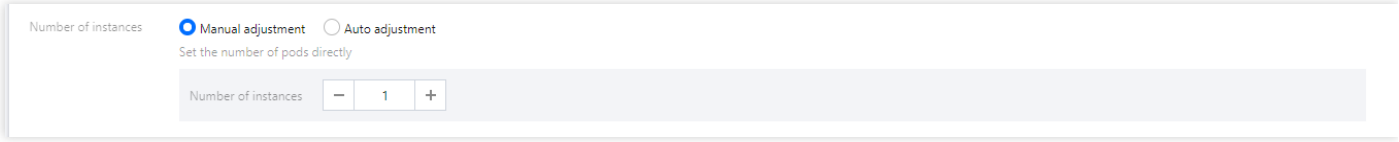


The main parameters are described as follows:

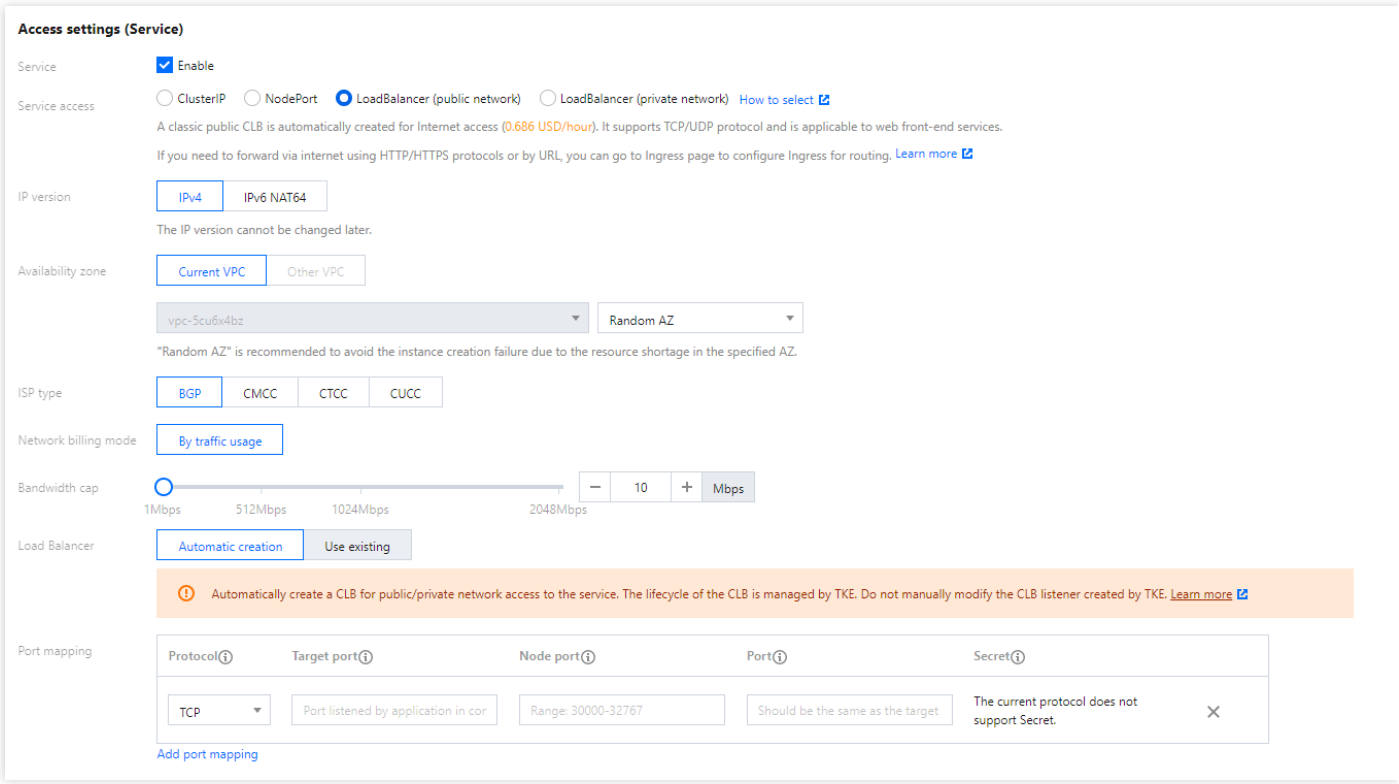
- **Name**: enter the name of the container in the pod. Here, “test” is used as an example.

- **Image**: click **Select Image**, select **DockerHub Image** > **wordpress** in the pop-up window, and click **OK**.
- **Image Tag**: use the default value `latest` .
- **Image Pull Policy**: choose from **Always**, **IfNotPresent** and **Never** as needed. In this document, we use the **default policy** as an example.

6. In **Number of Instances**, set the number of instances for the service according to the following information. In this document, we choose **Manual Adjustment** and set the instance number to one. See the figure below:



7. Specify the access mode of the workload, as shown in the following figure.



- **Service**: select **Enable**.
- **Service Access**: select **LoadBalancer (public network)**.
- **Load Balancer**: select according to your requirements.
- **Port Mapping**: select TCP, and set both the container port and service port to 80.

Note：

> The security group of the service's cluster must open the node network and container network to the Internet. It is also required to open ports 30000 to 32768 to the Internet. Otherwise, the problem of TKE being unusable could occur. For more details, see TKE Security Group Settings.

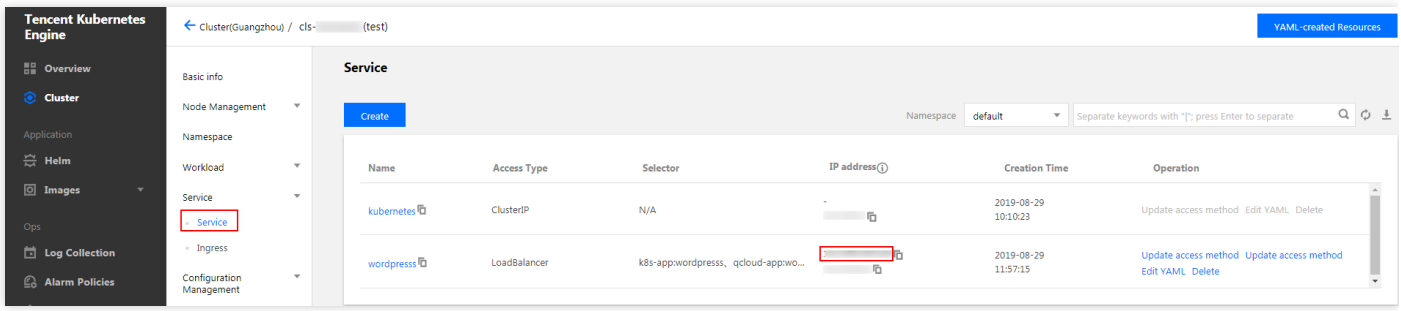
8. Click **Create Deployment**.

Accessing the WordPress service

You can access the WordPress service using either of the following two methods.

Access using the CLB IP address

1. In the left sidebar, click **Cluster** to go to the **Cluster Management** page.
2. Click the ID of the cluster to which the WordPress service belongs and choose **Services and Routes** > **Service**.
3. On the service list page, copy the CLB IP address of the WordPress service, as shown in the figure below.



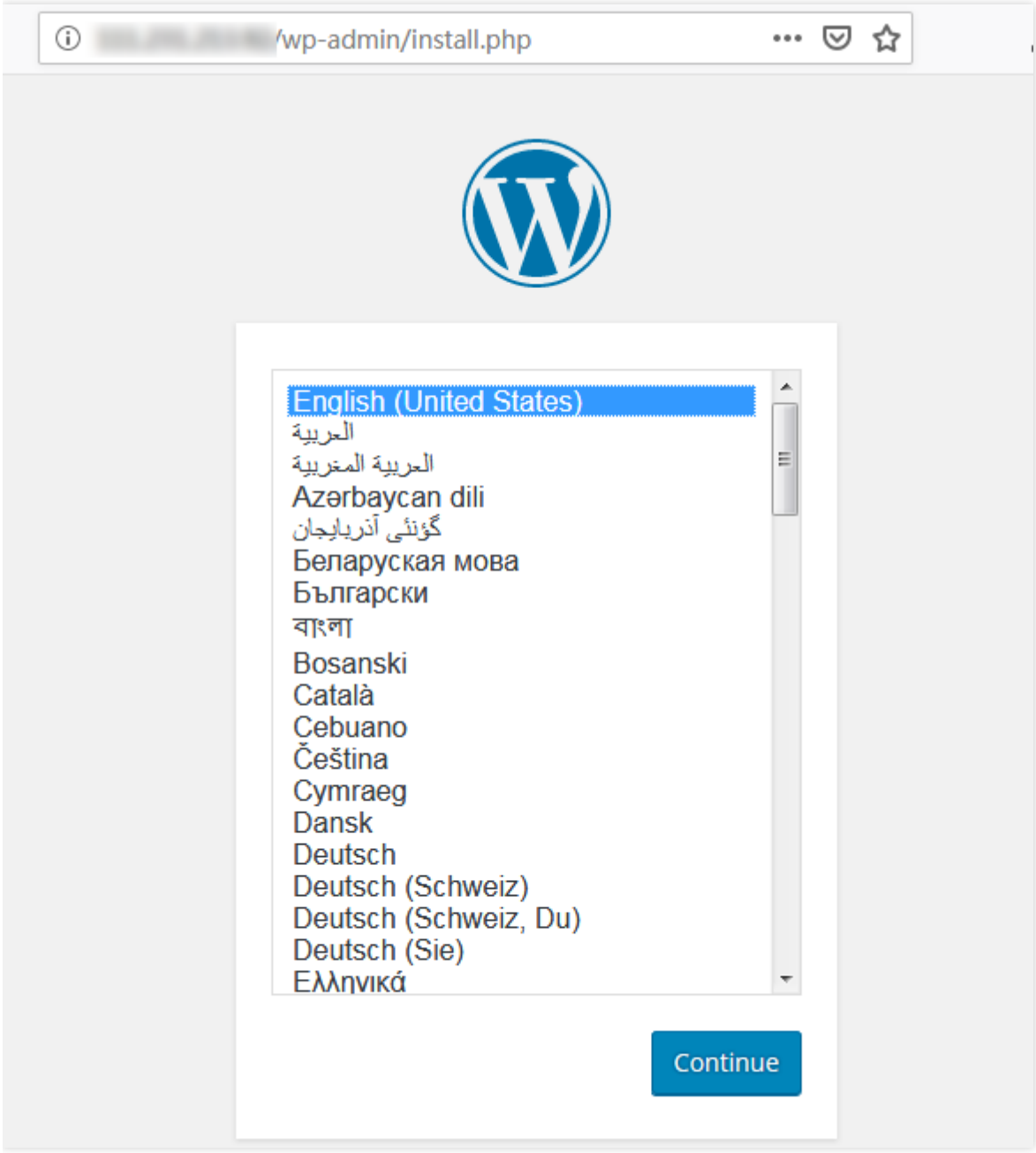
4. Paste the CLB IP address in the browser and press **Enter** to access the service.

Accessing the WordPress service using the service name

Other services or containers in the cluster can access the WordPress service using the service name.

Verifying the WordPress service

After the service is created, the WordPress server configuration page is displayed when you access the service, as shown in the figure below.



More WordPress settings

If the container fails to be created, you can view Event FAQs to locate the causes.

WordPress Service using TencentDB

Last updated：2023-02-02 17:05:22

Overview

To learn about how to quickly create WordPress services, you can refer to Creating a WordPress Service. WordPress services created in this way have the following features:

- Data is written to the MySQL databases running on the same container.
- Services can be quickly launched.
- Databases and storage-type files will be lost if the container is stopped for certain reasons.

Using MySQL databases can ensure permanent storage of data. The databases will continue to run when the pod/container restarts. This document explains how to configure the MySQL database using TencentDB and how to create a WordPress service that uses TencentDB.

Prerequisites

- You have registered a Tencent Cloud account.
- You have created a standard TKE cluster. For more information, see Creating a Cluster.

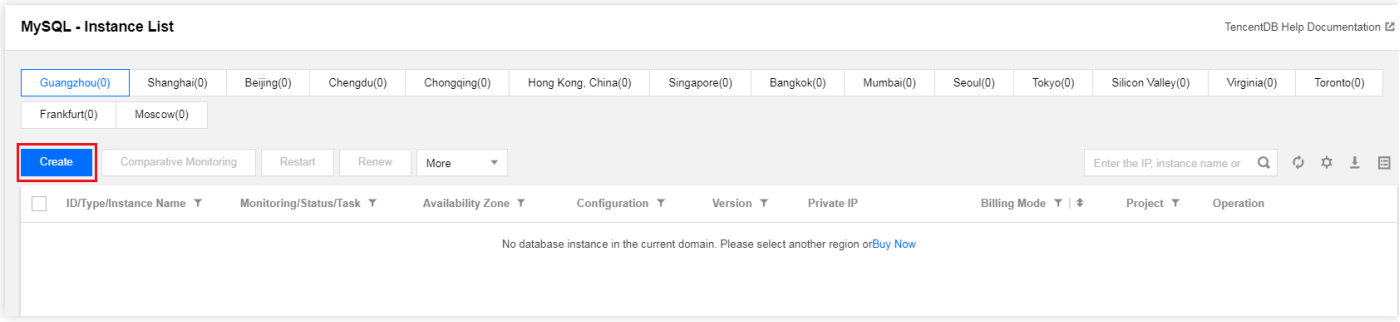
> Note：
> The database used in the document is TencentDB for MySQL instance.

Directions

Creating a WordPress service

Creating a TencentDB instance

1. Log in to the TencentDB for MySQL console, and click **Create** in the database instance list, as shown in the figure below.



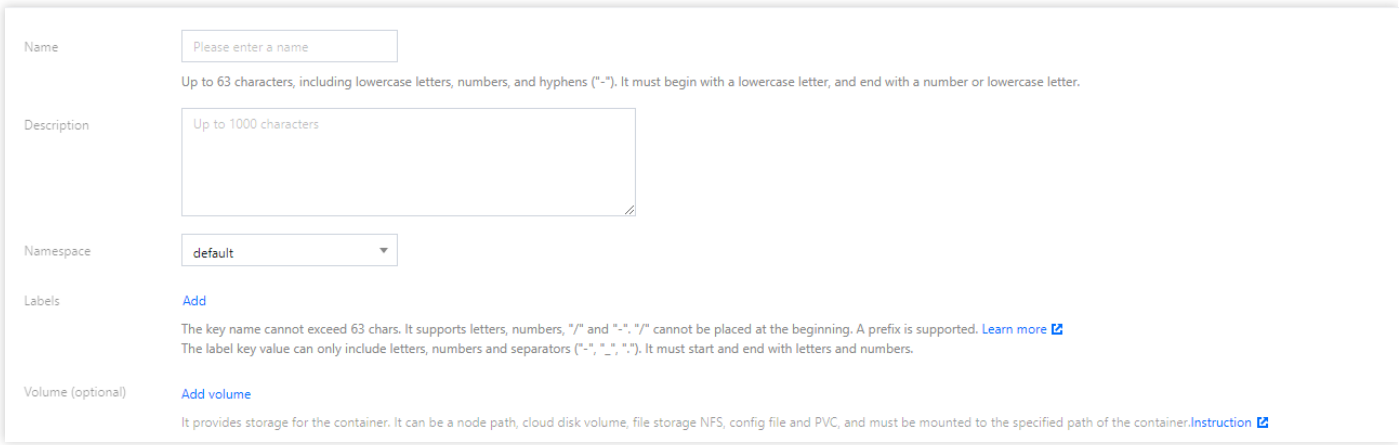
2. Select the configuration to purchase. For more information, see Overview.

> Note：
> The database must be in the same region as that of the cluster. Otherwise, you will be unable to connect to the database.

3. After creating the database, you can view it in the MySQL instance list.
4. Initialize the database. For details, see Initializing MySQL database.

Creating a WordPress service that uses Tencent DB

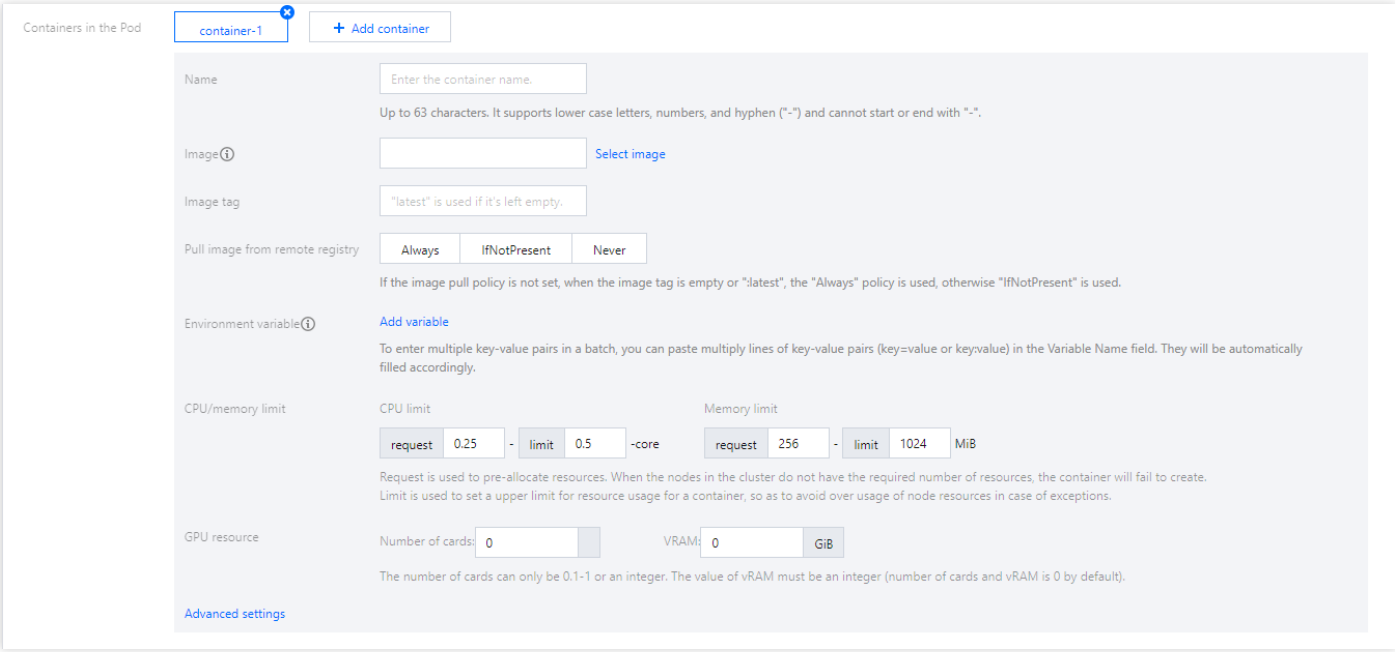
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the **Workload** > **Deployment** page, click **Create**. For more information, see Creating a Deployment.
4. On the **Create Deployment** page, specify basic information of the workload as instructed in the figure below.



- **Workload Name**: enter the name of the workload to create. In this example, `wordpress` is used.
- **Description**: specify related workload information.
- **Labels**: the default value is `k8s-app = wordpress` in this example.

- **Namespace**: select a namespace based on your requirements.
- **Volume**: set the volume to which the workload is mounted based on your requirements. For more information, see Volume Management.

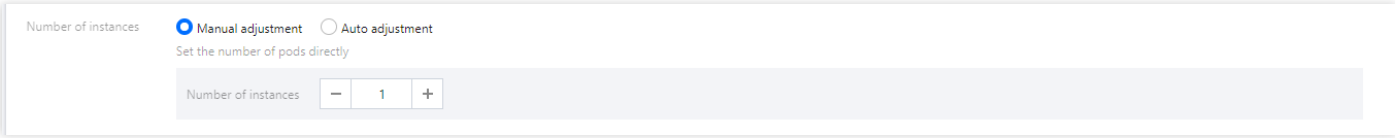
5. Configure "Containers in the Pod" as instructed. See the figure below:



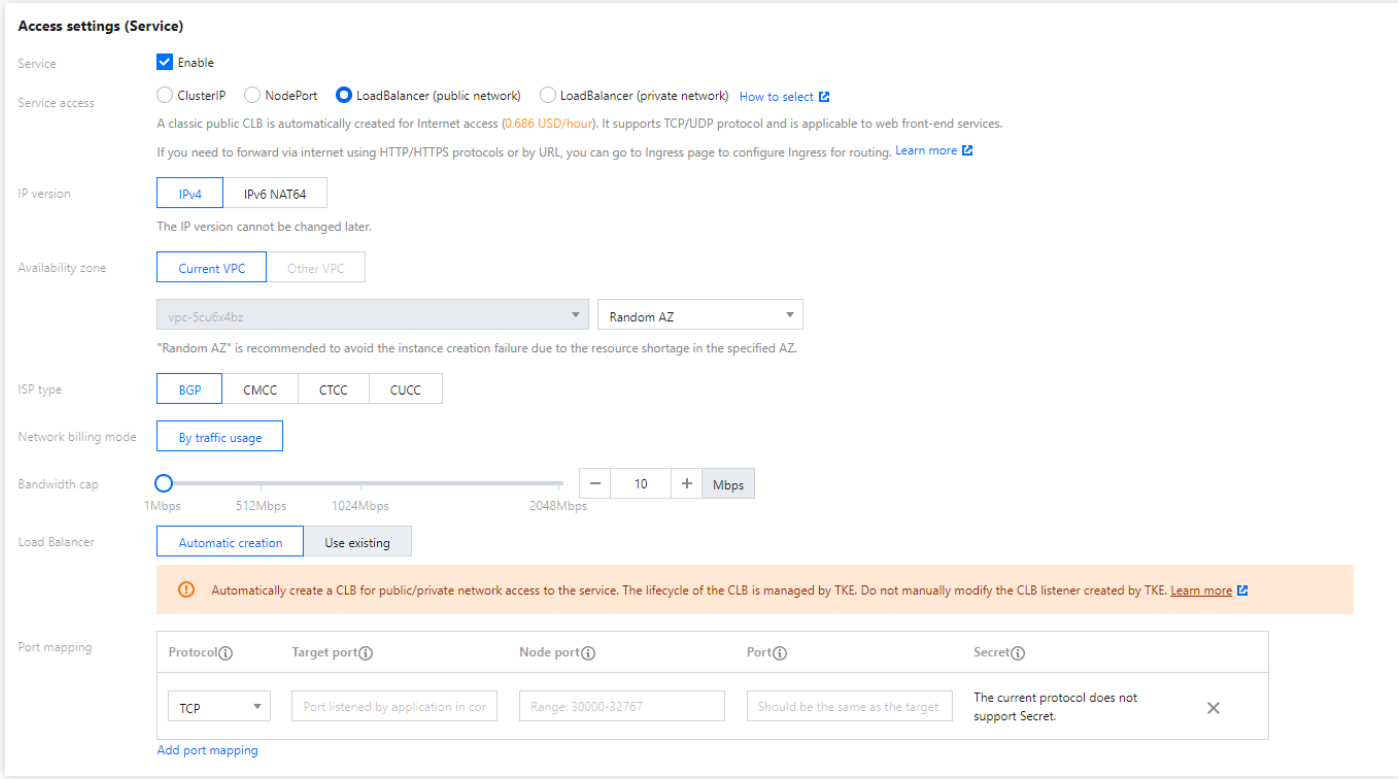
The main parameters are described as follows:

- **Name**: enter the name of the container in the pod. Here, “test” is used as an example.
- **Image**: click **Select Image**, select **DockerHub Image** > **wordpress** in the pop-up window, and click **OK**.
- **Image Tag**: use the default value `latest` .
- **Image Pull Policy**: choose from **Always**, **IfNotPresent** and **Never** as needed. In this document, we use the **default policy** as an example.
- **Environment variable**: enter the following configuration information in sequence.
  WORDPRESS_DB_HOST = Private IP address of TencentDB for MySQL
  WORDPRESS_DB_PASSWORD = Password entered during initialization

6. In **Number of Instances**, set the number of instances for the service according to the following information. In this document, we choose **Manual Adjustment** and set the instance number to one. See the figure below:



7. Specify the access mode of the workload, as shown in the following figure.



- **Service**: select **Enable**.
- **Service Access**: select **LoadBalancer (public network)**.
- **Load Balancer**: select according to your requirements.
- **Port Mapping**: select TCP, and set both the container port and service port to 80.

> Note：
> The security group of the service's cluster must open the node network and container network to the Internet. It is also required to open ports 30000 to 32768 to the Internet. Otherwise, the problem of TKE being unusable could occur. For more details, see TKE Security Group Settings.

8. Click **Create Deployment**.

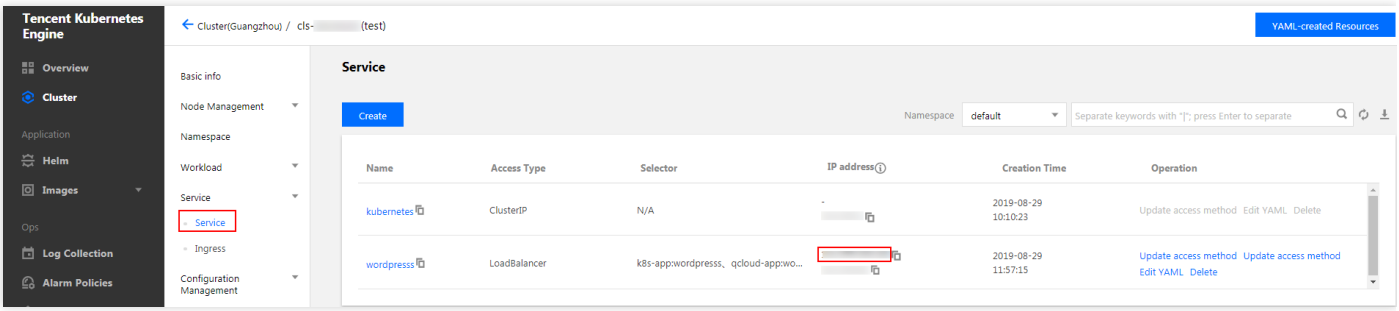
Accessing the WordPress service

You can access the WordPress service using either of the following two methods.

Accessing the WordPress service using the CLB IP address

1. In the left sidebar, click **Cluster** to go to the **Cluster Management** page.

2. Click the ID of the cluster to which the WordPress service belongs and choose **Services and Routes** > **Service**.
3. On the service list page, copy the CLB IP address of the WordPress service, as shown in the figure below.



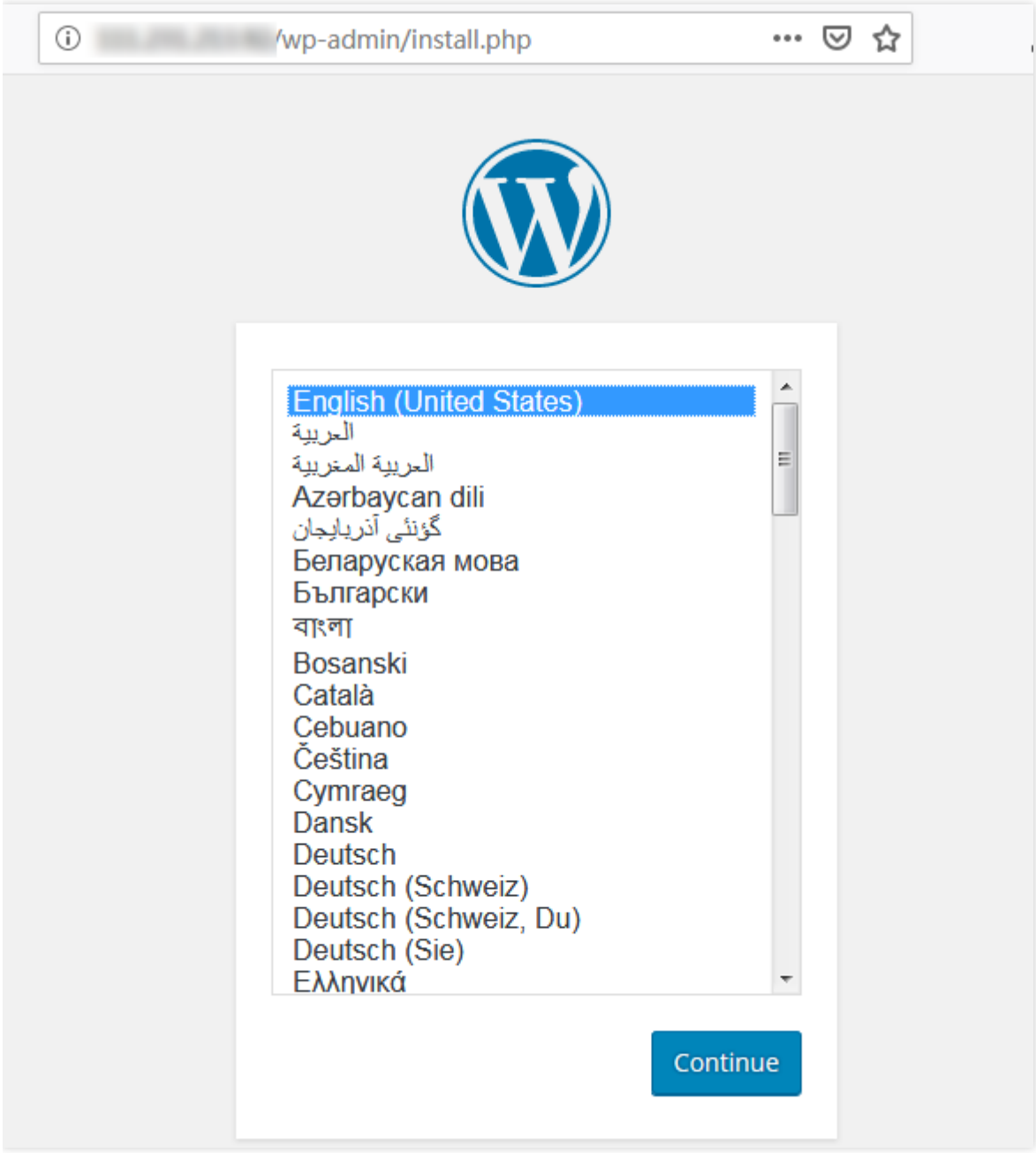
4. Paste the CLB IP address in the browser and press **Enter** to access the service.

Accessing the WordPress service using the service name

Other services or containers in the cluster can access the WordPress service using the service name.

Verifying the WordPress service

After the service is created, the WordPress server configuration page is displayed when you access the service, as shown in the figure below.



More WordPress settings

If the container fails to be created, you can view Event FAQs to locate the causes.

Building a Simple Web Application

Last updated：2023-08-10 15:55:46

Overview

This document describes how to create a simple web application through TKE.

A web application is divided into the following parts:

- The first part is the frontend service, which is used to handle query and write requests from clients.
- The other part is the data storage service, which uses Redis to store written data to redis-master and read data from redis-slave. Data is synchronized between redis-master and redis-slave through master-slave replication.

This application is a sample application supplied with the Kubernetes project. For more information, see the Guestbook App.

Prerequisites

- You have registered a Tencent Cloud account.
- You have created a cluster. For more information, see Creating a Cluster.

Directions

Creating a redis-master service

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster for which the application is to be created. On the cluster details page that appears, select **Workload** > **Deployment** and click **Create**.
3. On the **Create Deployment** page, configure basic information of the workload. The main parameters are as follows. Retain the default settings for other parameters.

- **Workload name**: **redis-master** is used as an example.

> Note
> For more information about the Deployment parameters, see Creating a Deployment.

4. Configure **Containers in Pod** as instructed. The main parameters are as follows. Retain the default settings for other parameters.

- **Name**: enter the name of the container in the pod. In this example, the name is **master**.
- **Image**: enter `ccr.ccs.tencentyun.com/library/redis`.
- **Image Tag**: enter **latest**.
- **Image Pull Policy**: in this example, you do not need to specify this field, but simply use the default policy.

5. Configure **Access Settings (Service)** for the workload as instructed below.

- **Service**: select **Enable**.
- **Service Access**: select **ClusterIP**.
- **Port Mapping**: select **TCP** from the **Protocol** drop-down list and set both **Port** and **Target Port** to **6379**. In this way, other services can access the master container by using the “redis-master” service name and the 6379 port number.

6. Click **Create Deployment**.

Creating a redis-slave service

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster for which the application is to be created. On the cluster details page that appears, select **Workload** > **Deployment** and click **Create**.
3. On the **Create Deployment** page, configure basic information of the workload. The main parameters are as follows. Retain the default settings for other parameters.

- **Workload Name**: indicates the name of the workload to be created. In this example, the name is **redis-slave**.

4. Configure **Containers in Pod** as instructed. The main parameters are as follows. Retain the default settings for other parameters.

- **Name**: enter the name of the container in the pod. In this example, the name is **slave**.
- **Image**: enter `ccr.ccs.tencentyun.com/library/gb-redisslave`.
- **Image Tag**: enter **latest**.
- **Image Pull Policy**: select the value as required. In this example, you do not need to specify this field, but simply use the default policy.
- **Environment Variable**: enter `GET_HOSTS_FROM = dns`.

5. Configure **Access Settings (Service)** for the workload as instructed below.

- **Service**: select **Enable**.
- **Service Access**: select **ClusterIP**.
- **Port Mapping**: select **TCP** from the **Protocol** drop-down list and set both **Port** and **Target Port** to **6379**. In this way, other services can access the master container by using the “redis-master” service name and the 6379 port number.

7. Click **Create Deployment**.

Creating a frontend service

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster for which the application is to be created. On the cluster details page that appears, select **Workload** > **Deployment** and click **Create**.
3. On the **Create Deployment** page, configure basic information of the workload. The main parameters are as follows. Retain the default settings for other parameters.

- **Workload Name**: the name of the workload to be created. In this example, the name is **frontend**.

4. Configure **Containers in Pod** as instructed. The main parameters are as follows. Retain the default settings for other parameters.

- **Name**: enter the name of the container in the pod. In this example, the name is **frontend**.
- **Image**: enter `ccr.ccs.tencentyun.com/library/gb-frontend` .
- **Image Tag**: enter **latest**.
- **Image Pull Policy**: select the value as required. In this example, you do not need to specify this field, but simply use the default policy.
- **Environment Variable**: enter `GET_HOSTS_FROM = dns` .

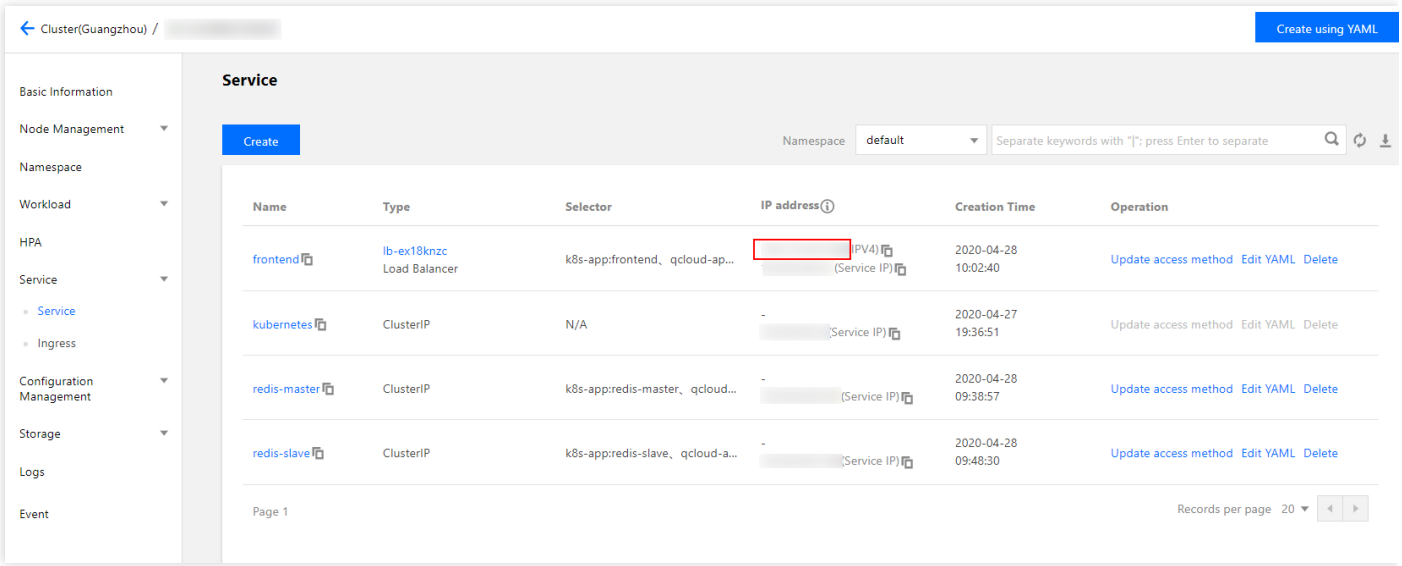
5. Configure **Access Settings (Service)** for the workload as instructed below.

- **Service**: select **Enable**.
- **Service Access**: select **Via Internet**.
- **Port Mapping**: select **TCP** from the **Protocol** drop-down list, and set both **Port** and **Target Port** to **80**. In this way, users can access the frontend container in a browser by using the load balancer IP address.

8. Click **Create Deployment**.

Verifying the web application

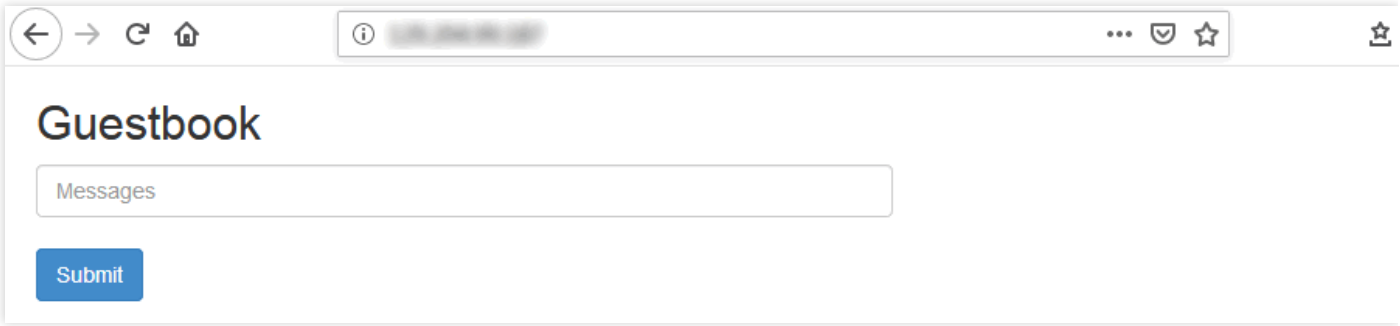
1. Go to the cluster details page, and choose **Services and Routes** > **Service** in the left sidebar.
2. On the **Service** page, copy the load balancer IP address of the frontend service.



Note：

- You have selected the *ClusterIP* access mode when creating the redis-master and redis-slave services. Therefore, these services have only one private IP address and can only be accessed by other services within the cluster.
- You have selected the **Via Internet** access mode when creating the frontend service. Therefore, this service has a load balancer IP address (which is a public IP address) and a private IP address. As a result, this service not only can be accessed by other services in the cluster, but also can be accessed through the Internet.

3. Access the load balancer IP address of the frontend service in a browser. If the following page appears, the frontend service can be normally accessed.



4. Enter a random string in the field and click **Submit**. You will see that the entered string is saved and displayed at the bottom of the page.
   Refresh the browser page to access the IP address of the service again. The previously entered string remains on the page, indicating that the string has been stored in Redis.

Development practices

The following sample code is the complete code for the frontend service of the Guestbook app. When receiving an HTTP request, the frontend service determines whether it is a set command:

- If yes, the frontend service obtains the key and value from the parameters, connects to the redis-master service, and applies the key and value to the redis-master service.
- If no, the frontend service connects to the redis-slave service, obtains the value of the key parameter, and returns the value to the client to display.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);
require 'Predis/Autoloader.php';
Predis\Autoloader::register();
if (isset($_GET['cmd']) === true) {
$host = 'redis-master';
if (getenv('GET_HOSTS_FROM') == 'env') {
$host = getenv('REDIS_MASTER_SERVICE_HOST');
}
header('Content-Type: application/json');
if ($_GET['cmd'] == 'set') {
$client = new Predis\Client([
'scheme' => 'tcp',
'host' => $host,
'port' => 6379,
]);
$client->set($_GET['key'], $_GET['value']);
print('{"message": "Updated"}');
} else {
$host = 'redis-slave';
if (getenv('GET_HOSTS_FROM') == 'env') {
$host = getenv('REDIS_SLAVE_SERVICE_HOST');
}
$client = new Predis\Client([
'scheme' => 'tcp',
'host' => $host,
'port' => 6379,
]);
```

```
$value = $client->get($_GET['key']);
print('{"data": "' . $value . '"}');
}
} else {
phpinfo();
} ?>
```

Notes

- When the frontend service accesses the redis-master and redis-slave services, it connects to the **Service name and port**. The cluster comes with the DNS service, which resolves the service name into the corresponding service IP address and performs load balancing according to this IP address.
  Assume that the redis-slave service manages three pods. If the frontend service is directly connected to the redis-slave service and port 6379, the DNS service automatically resolves the redis-slave service into an IP address of the redis-slave service, which is a floating IP address similar to a load balancer IP address. Then, the DNS service automatically performs load balancing based on the IP address of the redis-slave service and sends a request to the corresponding pod of the redis-slave service.
- Note the following when specifying **Environment Variable** for the container:
- **Default setting (recommended)**: during runtime, the frontend container reads the preset environment variable GET_HOSTS_FROM = dns and then directly connects to a service by using the service name.
- **Other settings**: to obtain the domain name of the redis-master or redis-slave service, you must specify another environment variable.