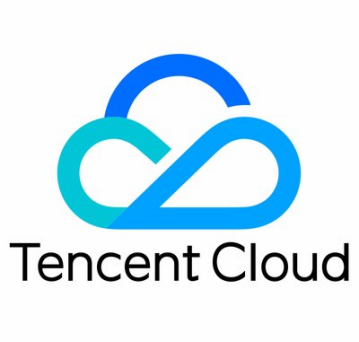
Tencent Kubernetes Engine

Getting Started



Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Getting Started
- Beginner's Guide
- Quickly Creating a Standard Cluster
- Examples
  - Creating Simple Nginx Service
  - Building Hello World Service Manually
  - WordPress with Single Pod
  - WordPress Service using TencentDB
  - Building a Simple Web Application

Getting Started
Beginner's Guide

Last updated: 2023-09-26 17:42:21

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

1. What Is TKE?

Tencent Kubernetes Engine (TKE) offers a container-centric, highly scalable, and high-performance container management service based on native Kubernetes. It is closely integrated with Tencent Cloud IaaS products, enabling customers to rapidly implement business containerization. For more information, see Product Overview.
TKE allows you to manipulate clusters and services in the TKE console or through TencentCloud APIs.

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

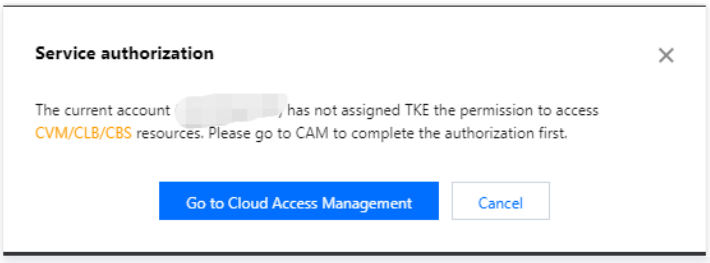
3.1 Register on Tencent Cloud

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

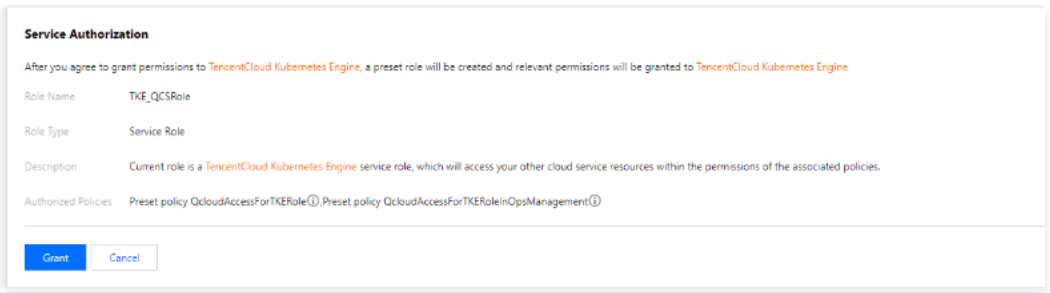
3.2 Role Authorization

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. Review the information in the "Service authorization" pop-up window and click **Go to Cloud Access Management**. As shown in the image below:



2. On the "Service authorization" page, carefully read the role-related information, as shown in the following image:



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

You can refer to the Quickly Create a Standard Cluster document to learn how to create a standard managed cluster. For a complete process, see the Create a Cluster document. If you need to use more types of clusters, please refer to the Create a

Serverless Cluster, Create a Container Instance, and Create an Edge Cluster documents.

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.

Desired Operation	Reference Document
To connect to a TKE cluster from a local client using Kubectl, the Kubernetes command line tool	Connecting to a Cluster
To upgrade a running Kubernetes cluster	Upgrading a Cluster
To add a pod to the created Kubernetes cluster	Adding a Node
To manage nodes in a Kubernetes cluster	Creating a Node Pool
To operate native Kubernetes objects on the console	Kubernetes Object Management
To provide a fixed access entry for a set of containers through service	Basic Features
To configure different forwarding rules through Ingress resources	Ingress Management
To leverage TKE's storage capability	Storage Management Overview
To assign the IP addresses within the container network address range to containers in the cluster	Container Network Overview
To store and analyze service logs in Kubernetes clusters	Log Collection
To monitor clusters	TPS Instance Management
To 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 a 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 Methods.
- **How does a container access the public network?**
  If the host where the container is located has a public IP and bandwidth, the container can directly access the public network. If the host does not have a public IP and bandwidth, the container can access the public network through a NAT Gateway.

- **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, container images, 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 numerous 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.

- If you find any issues with the product documentation, such as links, content, or API errors, you can click **Send Feedback** on the right side of the document page or highlight the problematic content to provide feedback.
- If you have any questions about the product, please seek assistance through online support.
- If you have other questions, go to the Tencent Cloud developer community to ask questions.

Quickly Creating a Standard Cluster

Last updated: 2023-09-15 14:32:12

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

Step 1. Register a Tencent Cloud account

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

Step 2: Topping 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. Authorizing TKE

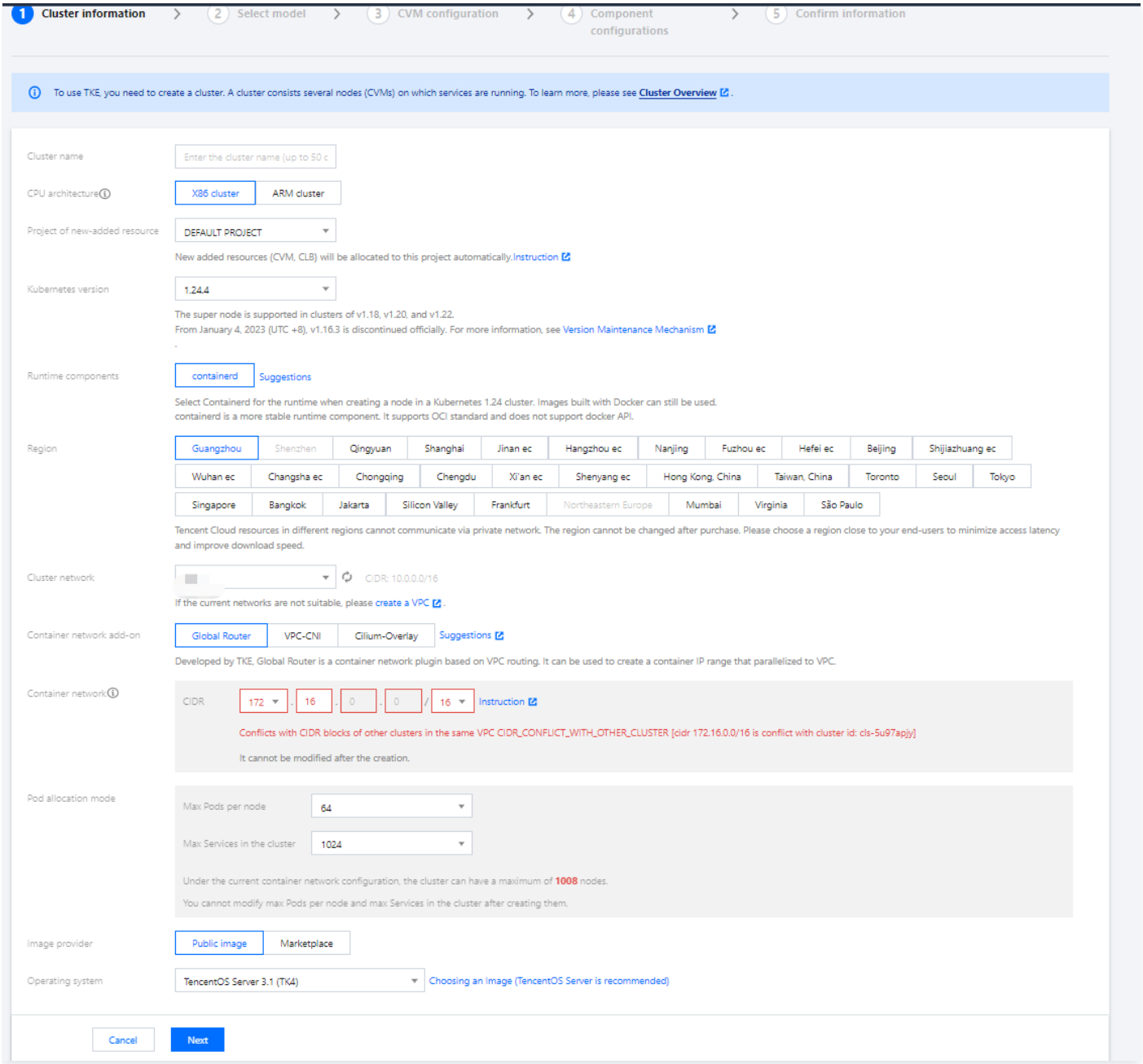
Log in to the Tencent Cloud console, select **Tencent Cloud services** > **Tencent Kubernetes Engine** to enter the TKE console and authorize TKE according to the prompts. If you have already authorized TKE, skip this step.

Step 4. Create a cluster

Log in to the TKE console to create a cluster.

Cluster information

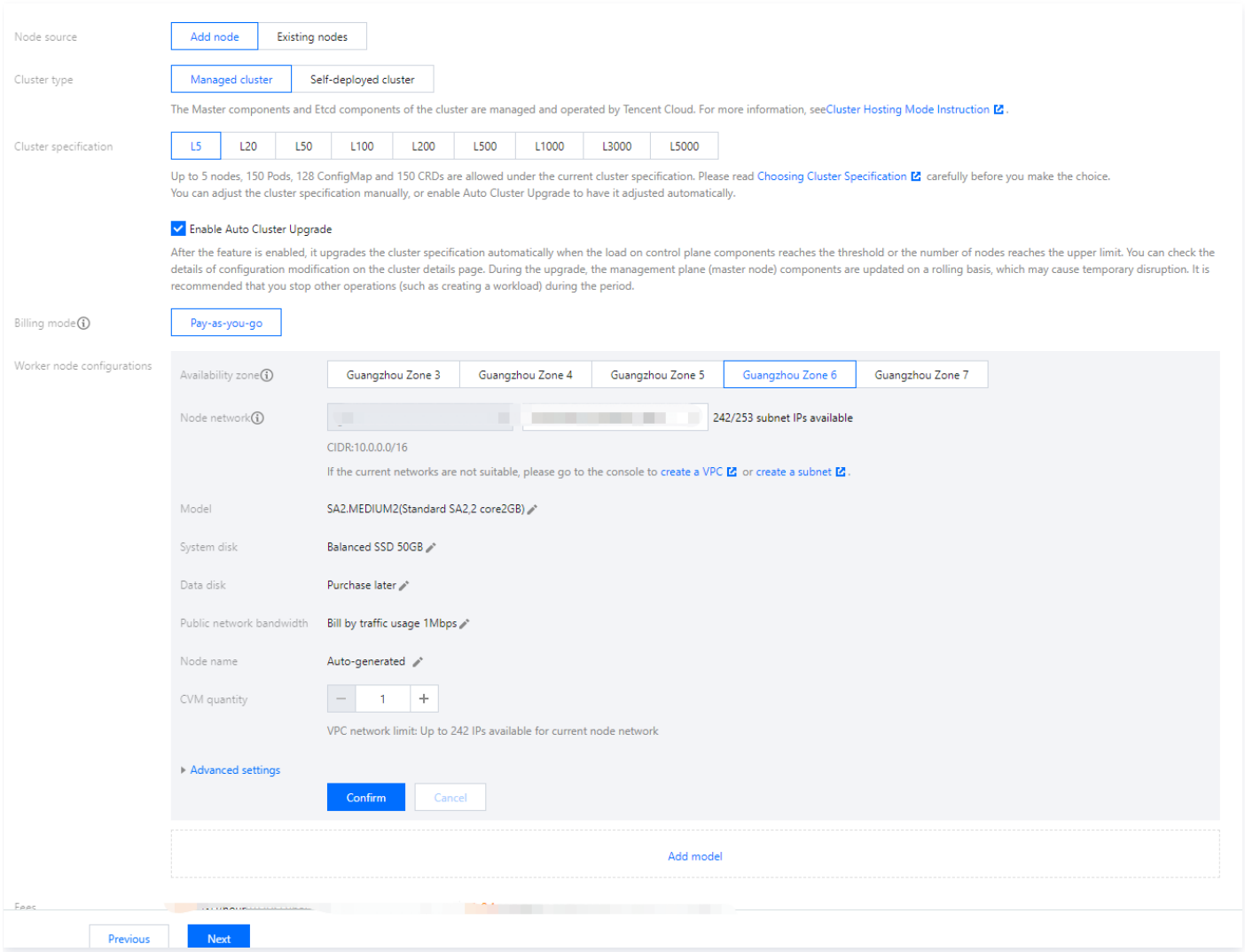
On the **Cluster Information** page, enter a **Cluster name**, select the **Region**, choose the **Cluster network** and **Container network**, keep the other default options, and click **Next**. See the figure below:



- **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.

Select a model

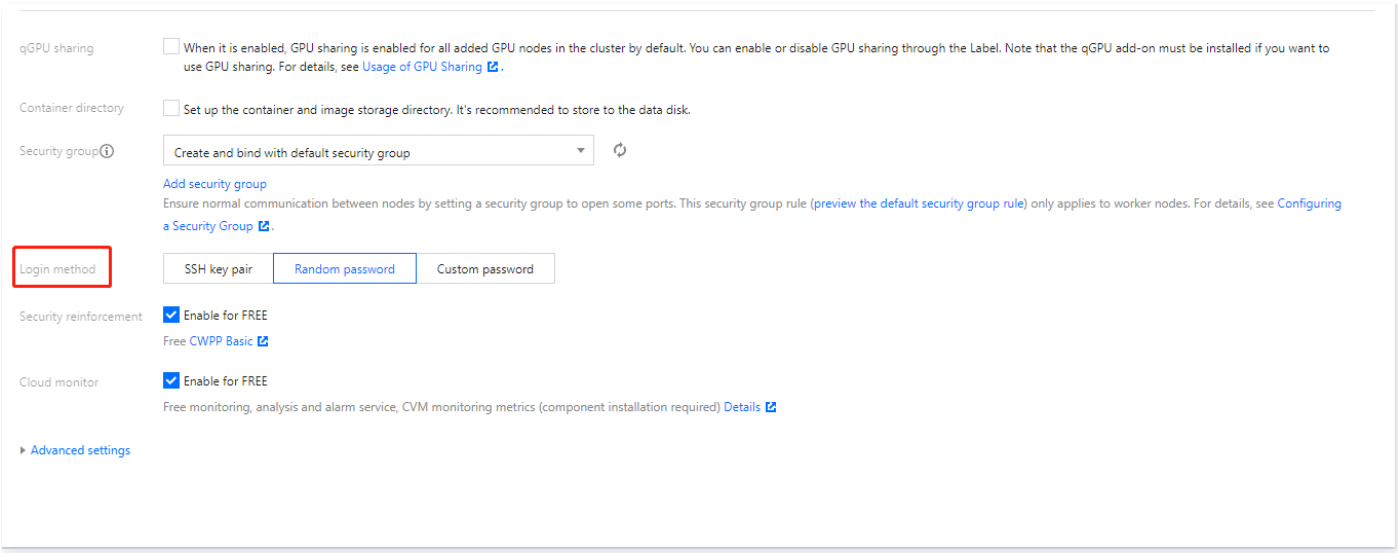
On the **Select Instance Type** page, confirm the **Billing mode**, select the **Availability zone** and **corresponding subnet**, **confirm the node instance type**, and click **Next**. As shown in the following figure:



- **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 the latter.
- **Cluster specification**: Various cluster specifications are available. In this case, we will select "L5".
- **Billing mode**: Offers both **Pay-as-you-go** and **Monthly Subscription** billing options. In this case, we choose "Pay-as-you-go".
- **Worker node 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)**.

Configure CVM

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



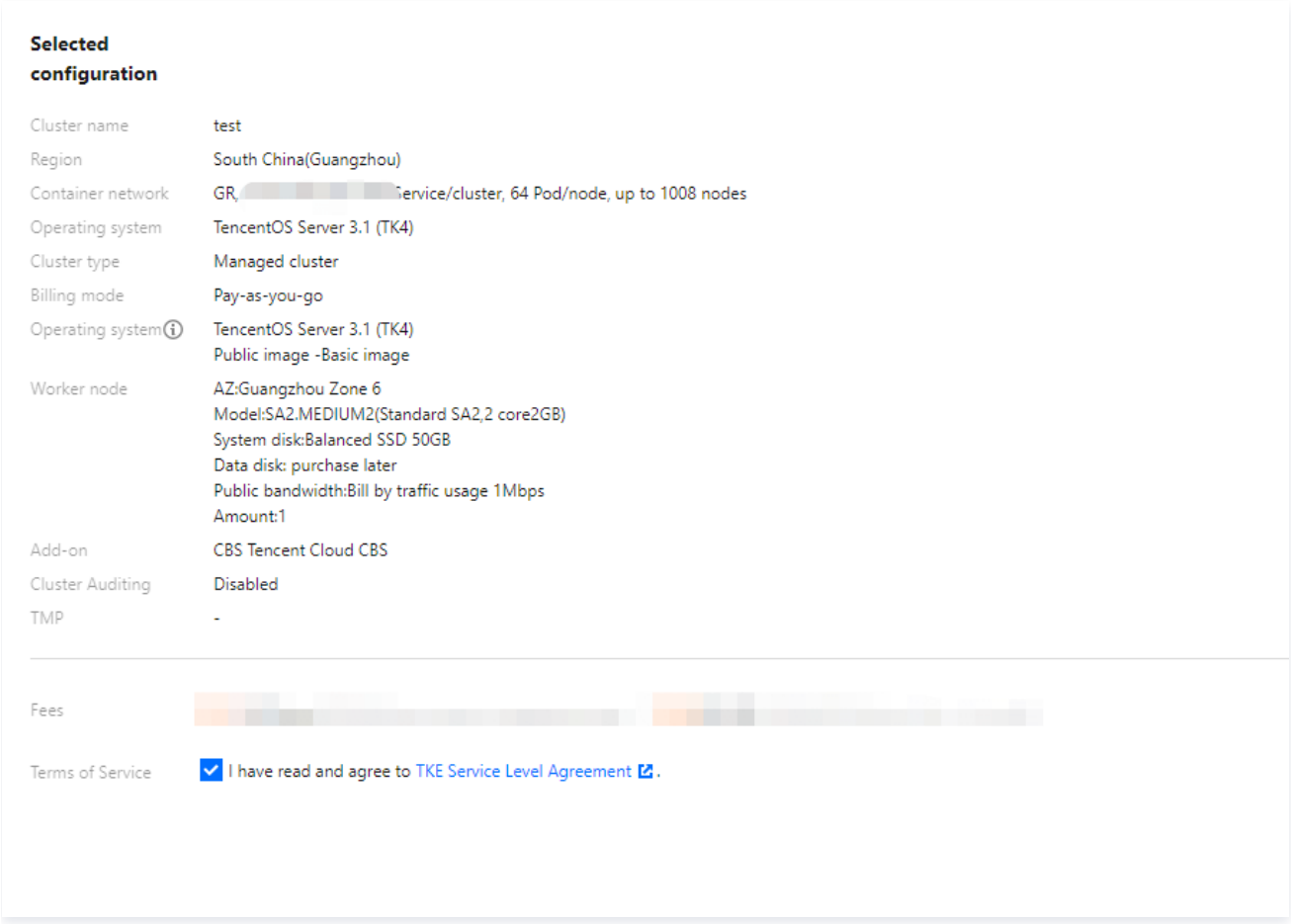
- **Login method**: You can select **SSH key pair**, **Random password** or **Custom password**. Here we select "Random password".

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

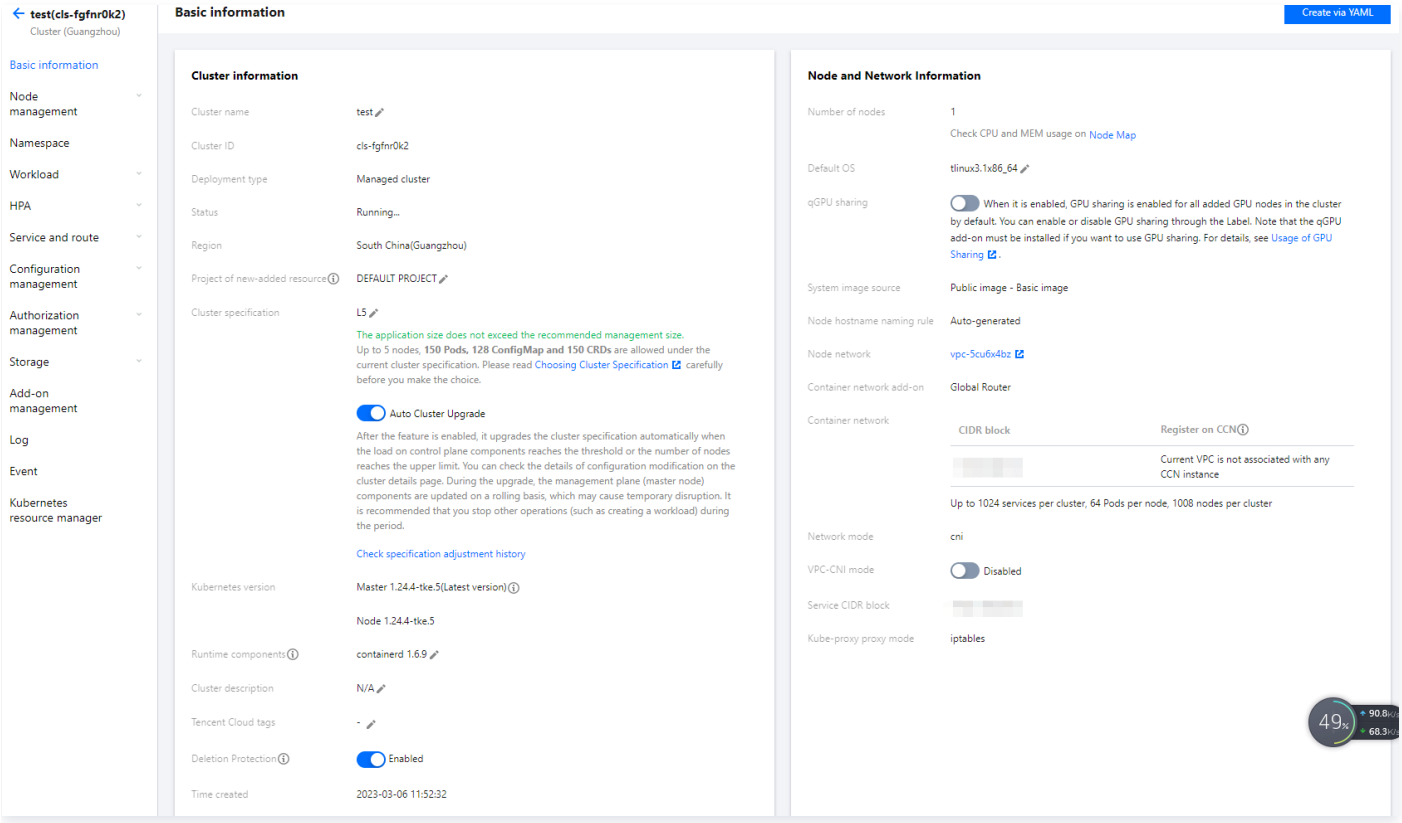
On the "Selected configuration" page, review the selected cluster configuration and costs. After reading and agreeing to the TKE Service Level Agreement, click **Finish**. As shown in the image below:



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: Viewing the Cluster

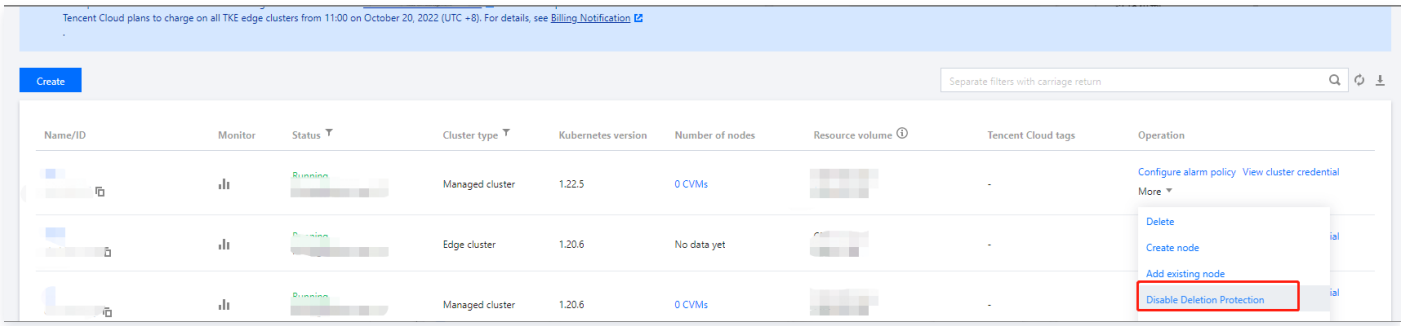
You can view clusters that have been created in the cluster list. You can click the cluster ID to enter the details page, and then view the cluster, node, network, and API Server information on the **Basic information** page.



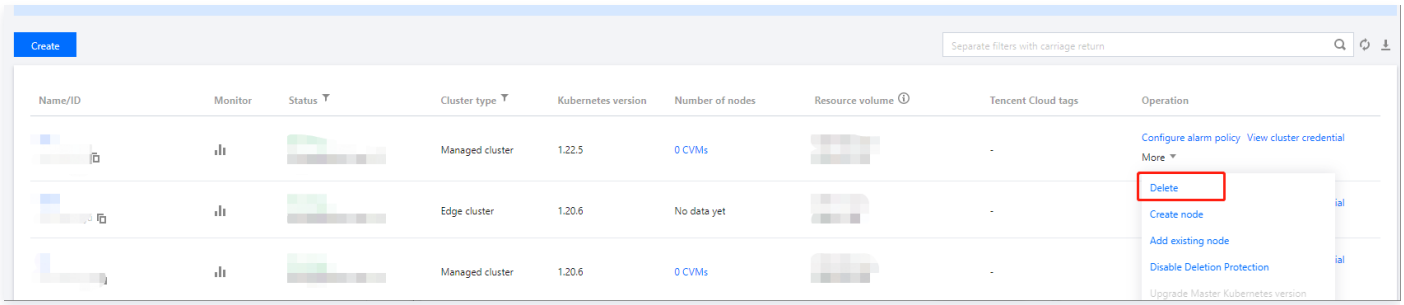
Step 6: Deleting Clusters

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 **Clusters** in the left navigation pane, and on the "Cluster Management" page, choose **More > Disable Deletion Protection** in the row where the cluster to be deleted is located, as shown below:



2. Select **More > Delete** on the right side of the cluster row, as shown below:



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.

Examples
Creating Simple Nginx Service

Last updated: 2023-09-15 14:41:04

Scenario

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

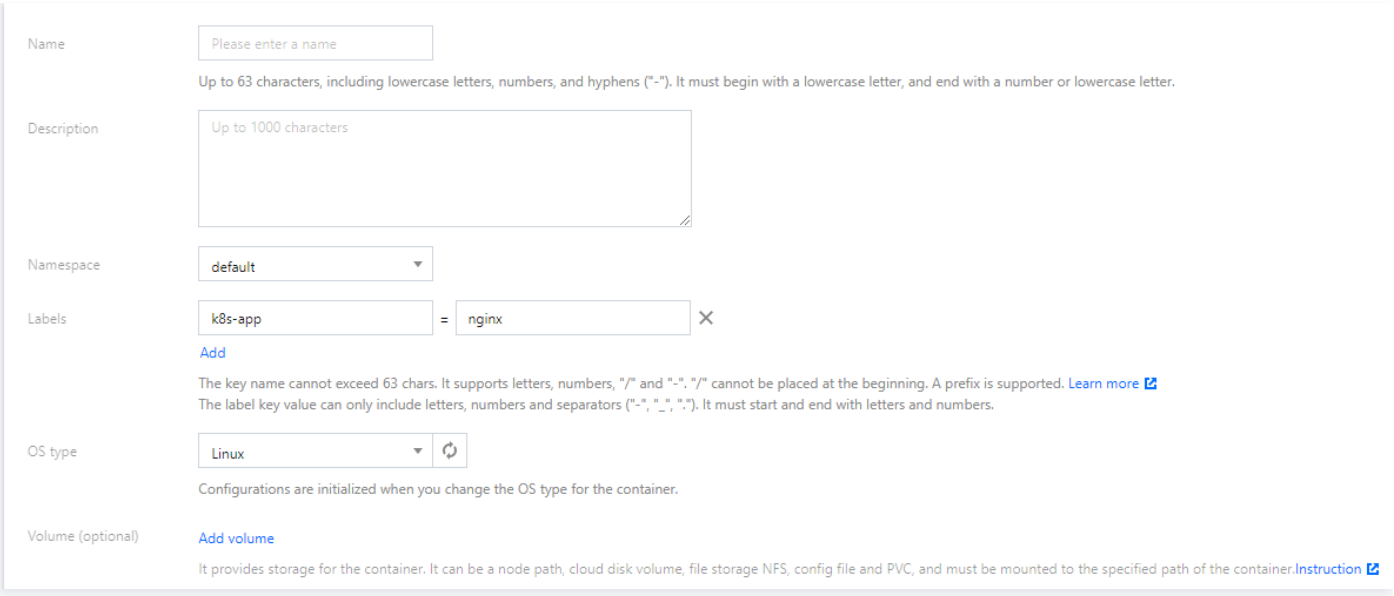
Preparations

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

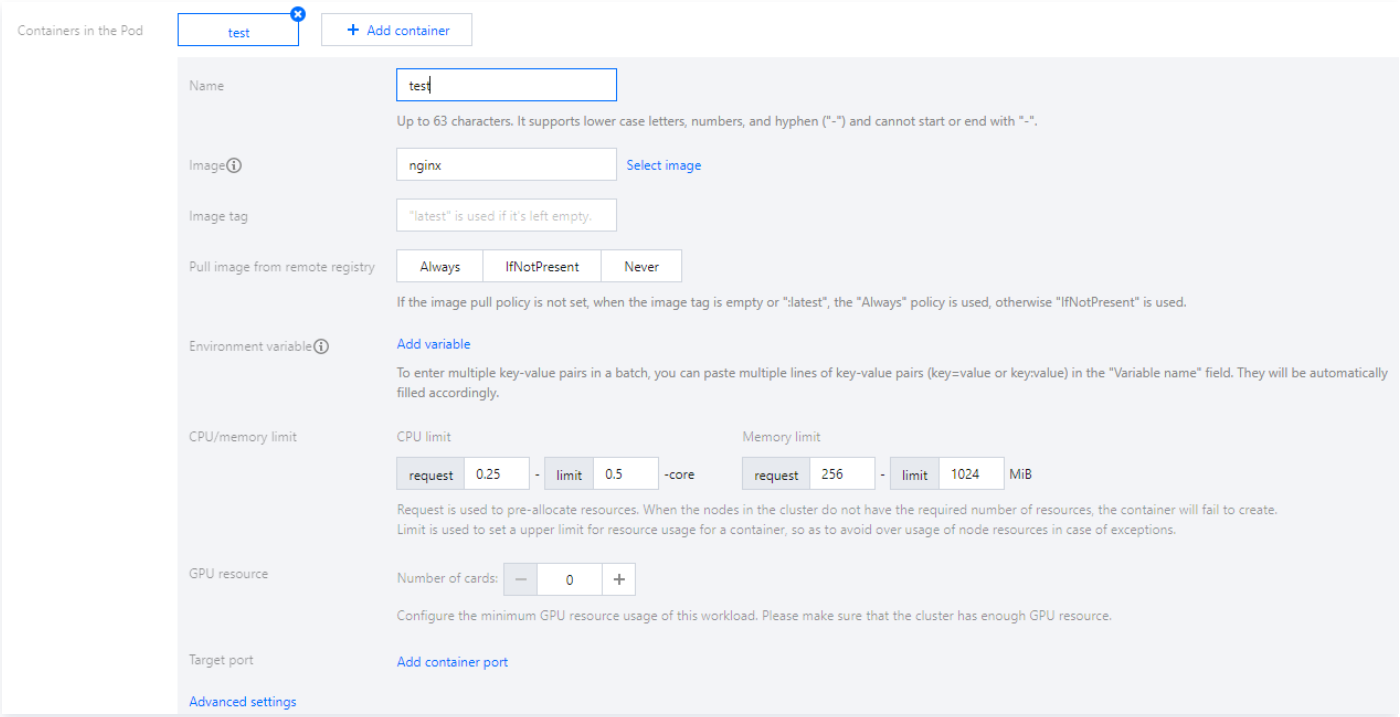
Instructions

Creating a Nginx service

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, select the cluster ID for which you want to create a service, and enter the basic information page of the cluster.
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.



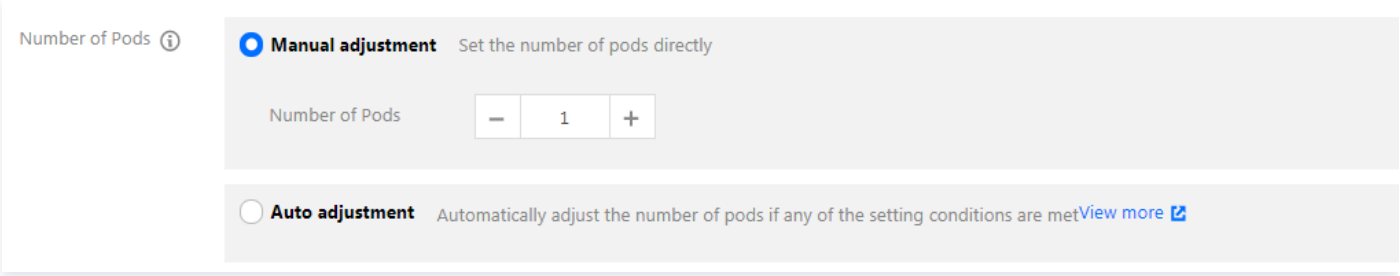
   - **Name**: In this example, we use Nginx.
   - **Description**: specify related workload information.
   - **Labels**: In this example, the default value of the label is `k8s-app = nginx`.
   - **Namespace**: Choose according to your actual requirements. The default is "default".
   - **Volume**: set the volume to which your workload will be mounted based on your requirements. For more information, see Instructions for Other Storage Volumes.
5. Configure the "Instance Container" according to the following information, as shown in the image below:



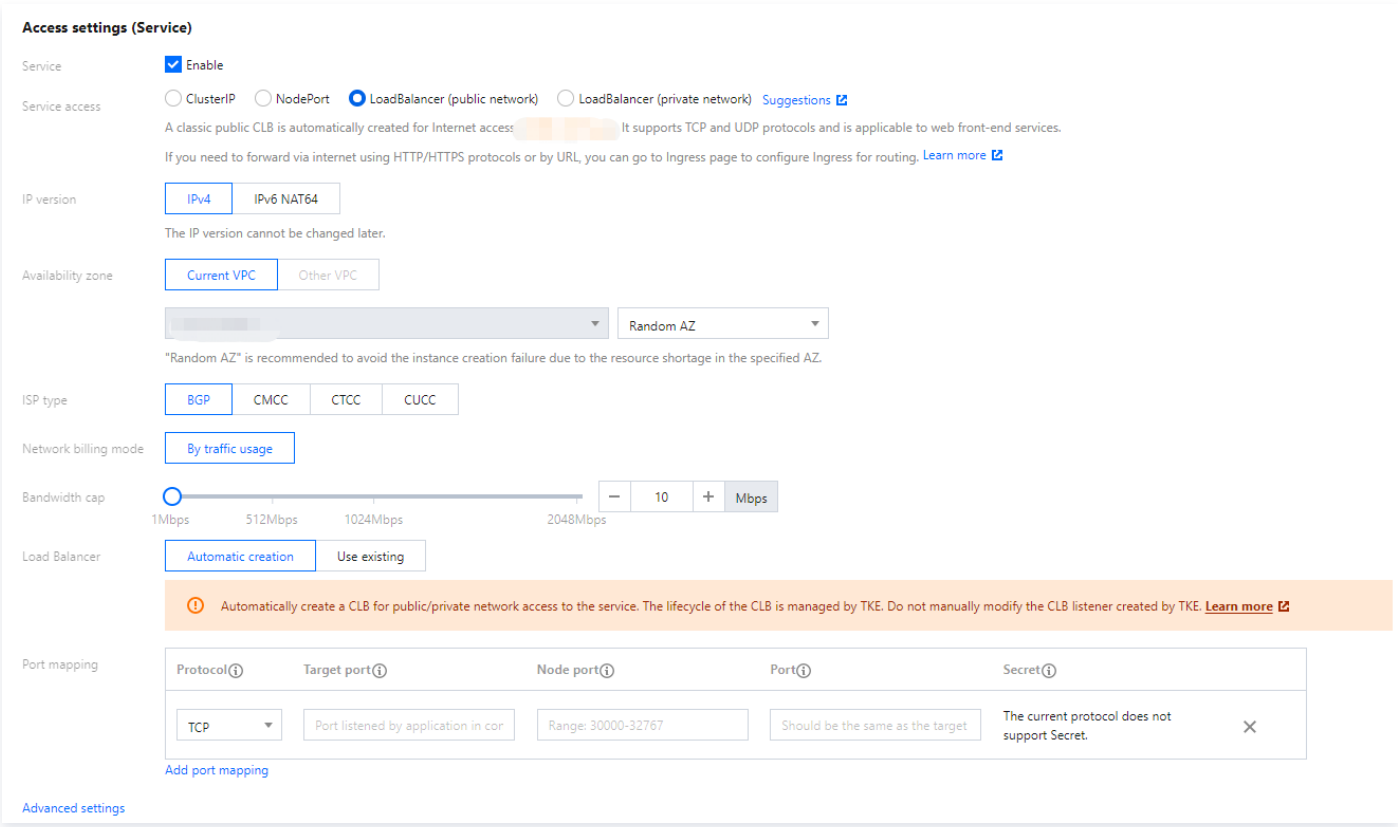
The main parameter information is as follows:

- **Name**: Enter the name of the container in the pod. Here, “test” is used as an example.
- **Image**: Enter "nginx" to use the official Nginx image from DockerHub by default.
- **Image tag**: Use the default value `latest`.
- **Pull image from remote registry**: Three policies (`Always`, `IfNotPresent`, and `Never`) are available. Select a policy as required. In this example, **the default policy is used**.

6. In "Instance Quantity," set the number of service instances based on the following information. This document uses **manual adjustment** as an example, with the instance quantity set to 1. As shown in the image below:



7. Configure the workload access settings according to the following instructions, as shown in the image 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.

> **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.

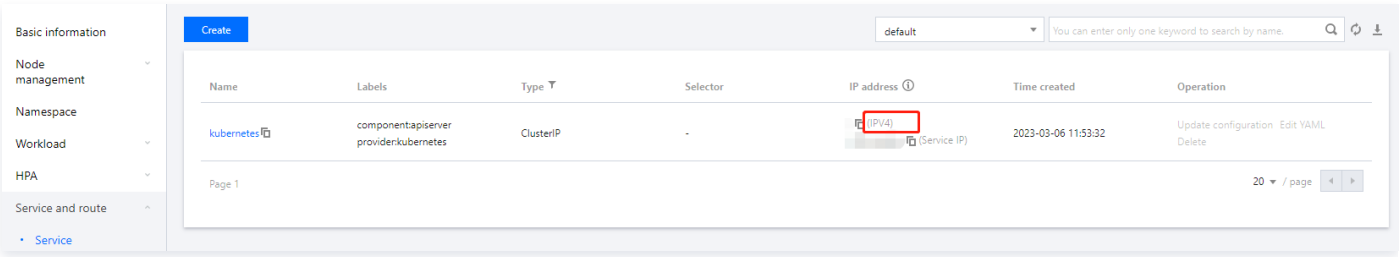
8. Click **Create workload** to complete the creation of the Nginx service.

Accessing the Nginx service

Nginx service can be accessed using the following two methods.

Accessing the Nginx service using the load balancer IP

1. Click Cluster in the left navigation bar to access the "Cluster Management" page.
2. Click the ID of the cluster to which the Nginx service belongs and select **Service and Route** > **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 the Nginx service using the service name

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

Verifying Nginx service

Upon successful creation of the service, accessing it will directly lead to the default welcome page of the Nginx server. As illustrated in the figure below:

More Nginx settings

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

Building Hello World Service Manually

Last updated: 2023-09-15 14:57:41

Scenario

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

Preparations

- You have created a cluster. For more information, see Creating a Cluster.
- You have logged in to a node with Node.js installed. For more information, see Deploying a Node.js Environment with an Image.

Instructions

Writing Codes to Create an Image

Write 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 the server.js file.

```
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:

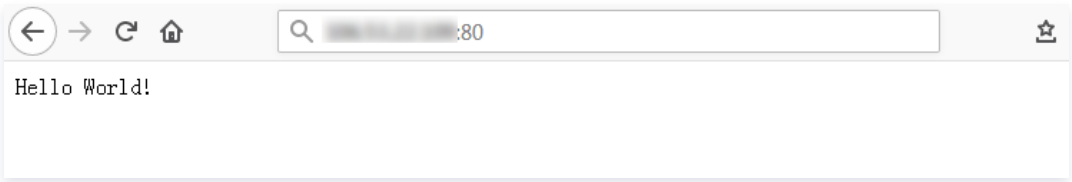
```
curl 127.0.0.1:80
```

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 URL in the format `Public IP address of the CVM instance: Configured port number`, with the port number set to 80.

If the following is displayed, it indicates that the Hello World program is running successfully.



Creating 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 edit mode and enter the following content into the Dockerfile.

```
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 the image.

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

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

```
docker images
```

   If the following result is displayed, it indicates that the hello-node image has been successfully built. Note down its IMAGE ID, as shown below:

Uploading Image to Tencent Cloud Image Registry

You have created a Tencent Container Registry (TCR) Individual Edition instance. For more information, see Getting Started with TCR Individual Edition.
Execute 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 **IMAGEID** in the command with the IMAGEID noted down in Viewing Images.
- Please replace **namespace** in the command with the namespace you have created. If you have not created a namespace yet, please create one first. For more information, see Creating a Namespace.

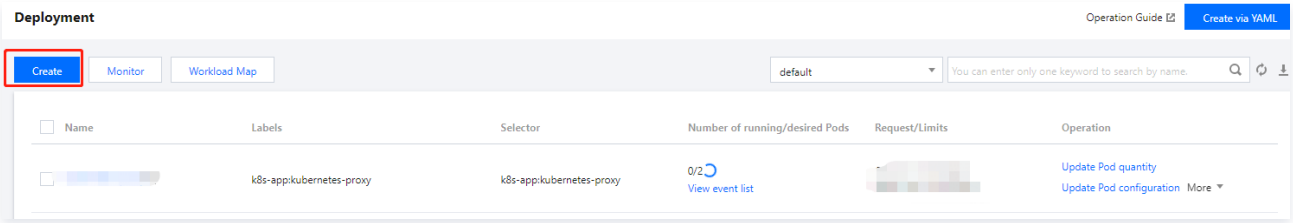
If the following information appears, the image is successfully uploaded.

```
[root@VM_2_5_centos hellonode]# sudo docker tag  ccr.ccs.tencentyun.com/  test/helloworld:v1
[root@VM_2_5_centos hellonode]# sudo docker push ccr.ccs.tencentyun.com/  test/helloworld:v1
The push refers to repository [ccr.ccs.tencentyun.com/  test/helloworld]
7357e3a21b1f: Pushed
20a6f9d228c0: Pushed
80c332ac5101: Pushed
04dc8c446a38: Pushed
1050aff7cfff: Pushed
66d8e5ee400c: Pushed
2f71b45e4e25: Pushed
v1: digest: sha256:9c139ecbb29c49f25e02d7906b9e78c6e2e274827a75603ef48fa5547ff8a620 size: 1794
```

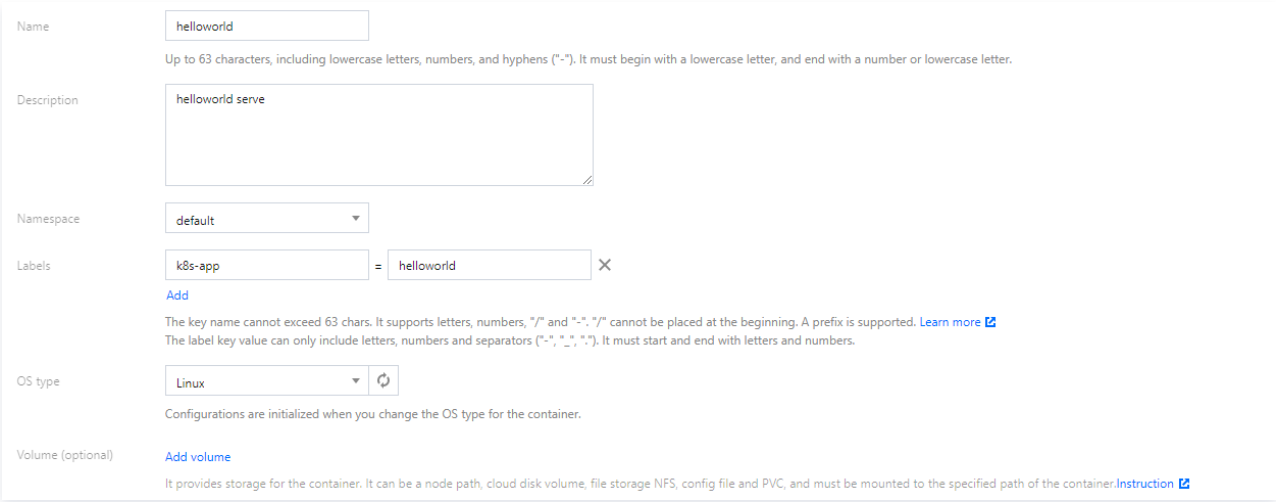
Creating Hello World Service Using This Image

> **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, select the cluster ID for which you want to create a service, and enter the **Cluster Details** page.
3. Select **Workloads > Deployment**, and click **Create** on the Deployment page, as shown in the following image:



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



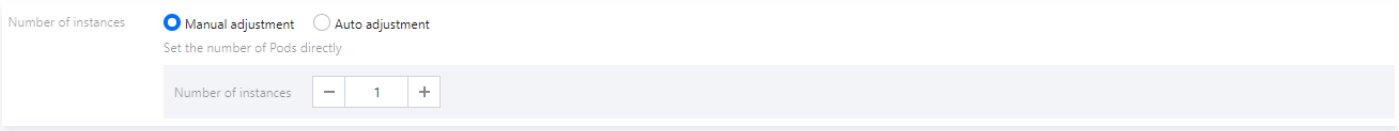
   - **Name**: Enter the name of the workload to create. In this example, **helloworld** is used.
   - **Description**: specify related workload information.
   - **Namespace**: select a namespace based on your actual requirements.
   - **Labels**: Specify the key–value pair. The default value is **k8s–app = helloworld** here.
   - **OS 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 the pod. In this document, **helloworld** is used as an example.

5.2 Click **Select Image**. In the pop-up, select **My Images**, find the helloworld image using the search function, and click **Confirm**. The main parameter information is as follows:

- **Image Tag**: Use the default value **latest**.
- **Image Pull Policy**: Offers three policies: **Always, IfNotPresent, and Never**. Choose according to your needs. This document uses the default policy as an example, without any specific settings.

6. In "Number of instances", set the number of service instances based on the following information, as shown in the image 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 number of pods is automatically adjusted if any of the setting conditions are 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 Method**: Select "Public Network CLB Access".
   - **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.

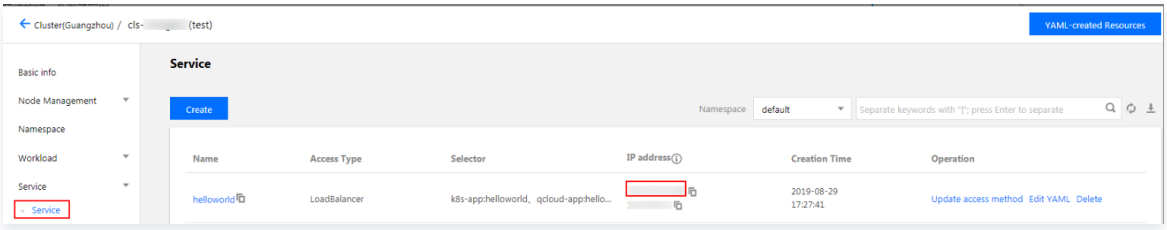
8. Click **Create Deployment** to create the Hello World service.

Accessing HelloWorld Service

HelloWorld service can be accessed using the following two methods.

Accessing via Load Balancer IP

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the cluster ID where the Hello World service is located to enter 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:



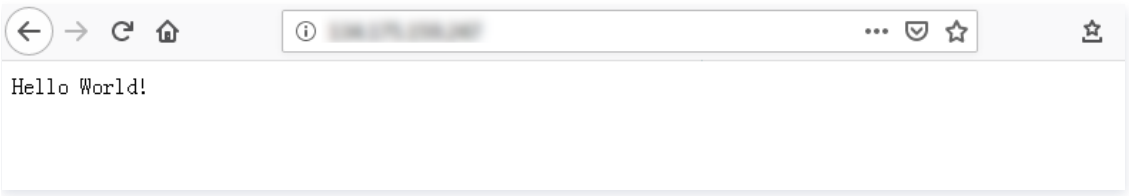
5. Paste the CLB IP address for the Hello World service in your browser.

Access Using Service Name

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

Verifying Hello World Service

If the following is displayed when accessing the service, the Hello World service has been successfully created.



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

WordPress with Single Pod

Last updated: 2023-09-15 15:05:15

Scenario

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.

Preparations

Note

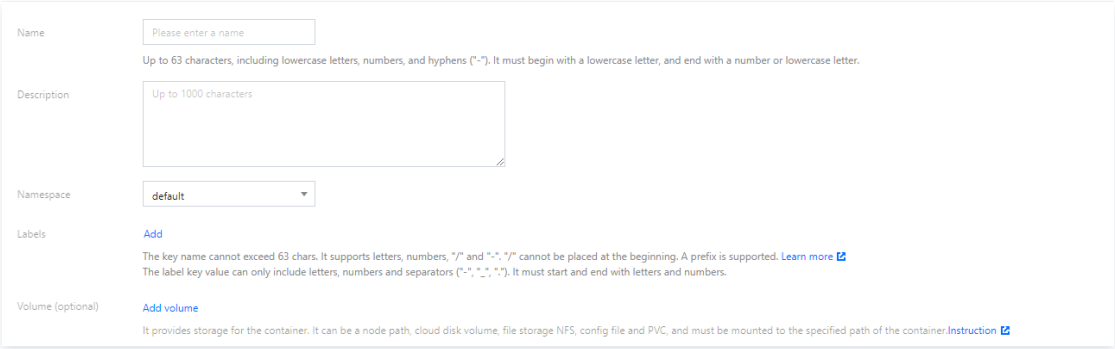
- `wordpress` This image contains all the runtime environments required for WordPress, and you can create a service by directly pulling the image.
- WordPress with a single Pod is used for testing purposes only, and therefore cannot ensure persistent data storage. It is recommended that you use a self-built 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.

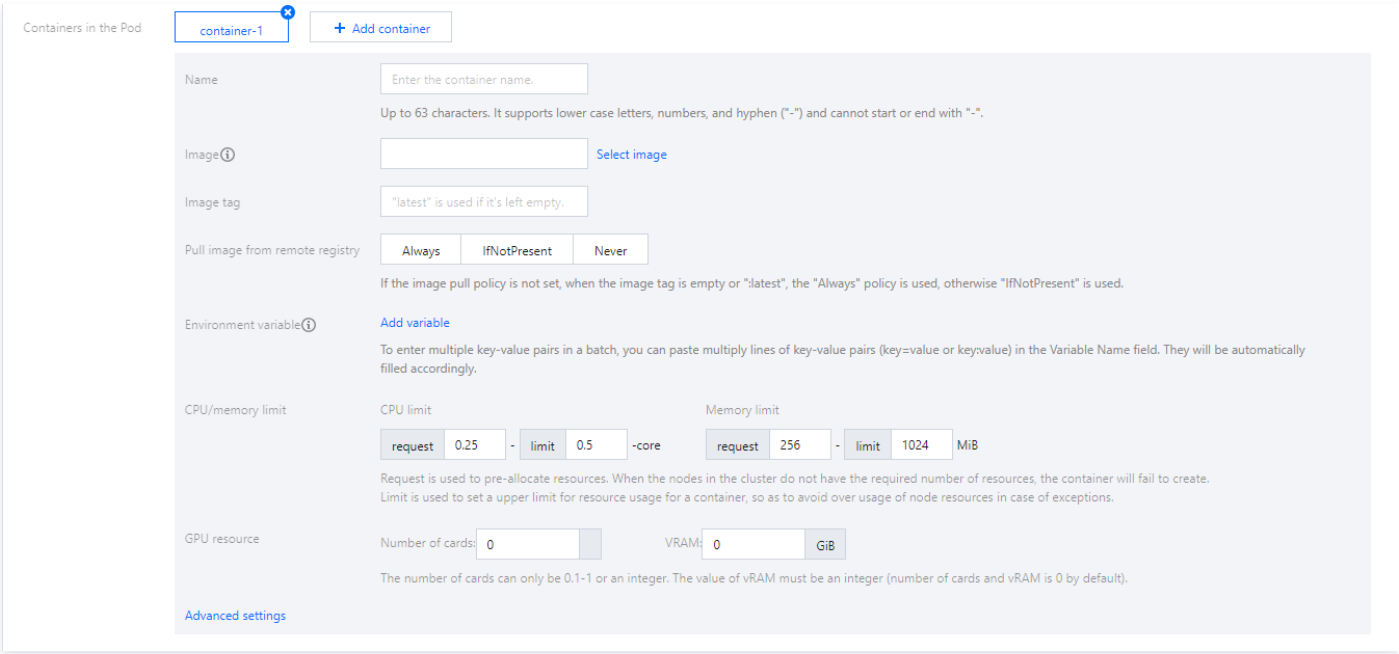
Instructions

Creating a WordPress service

1. Log in to the TKE console and click 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.



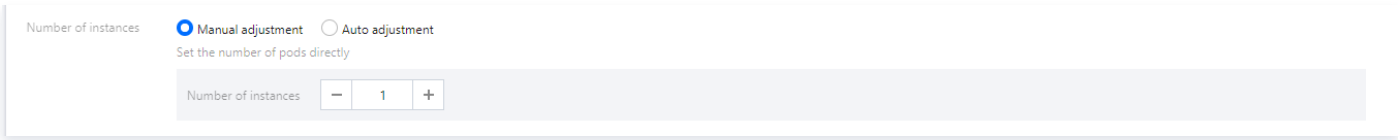
   - **Name**: Enter the name of the workload to be created, using "wordpress" as an example in this document.
   - **Description**: specify related workload information.
   - **Labels**: the default value is `k8s-app = wordpress` in this example.
   - **Namespace**: select a namespace based on your actual requirements.
   - **Volume**: Set up the workload volumes mounted based on your requirements. For more details, see Volume Management.
5. Configure the "Container in the pod" according to the following information, as shown in the image below:



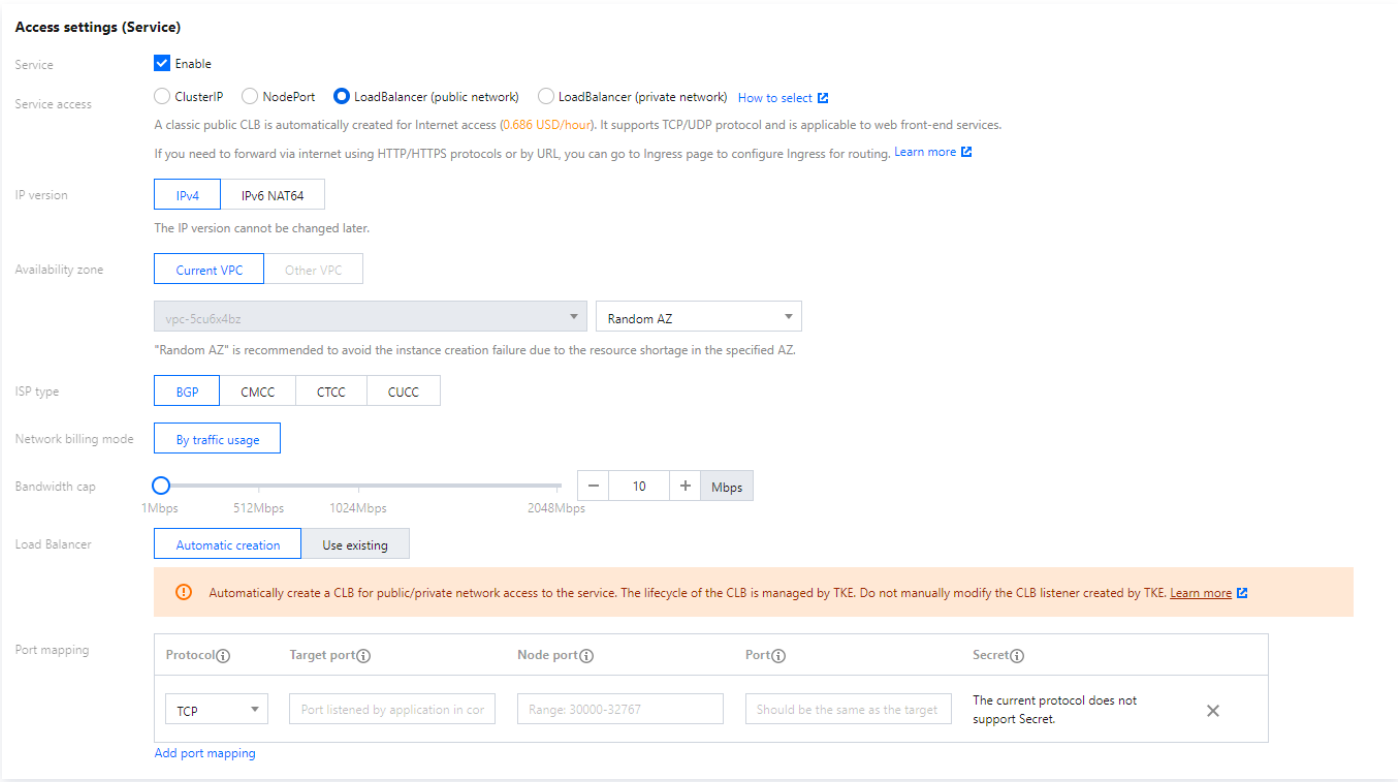
The main parameter information is as follows:

- **Name**: Enter the name of the container in the pod. Here, “test” is used as an example.
- **Image**: enter `wordpress`.
- **Image tag**: Use the default value `latest`.
- **Pull image from remote registry**: Three policies (`Always`, `IfNotPresent`, and `Never`) are available. Select a policy as required. In this example, **the default policy is used**.

6. In "Number of Instances", set the number of service instances based on the following information. This document uses **Manual adjustment** as an example, with the instance quantity set to 1. As shown in the image below:



7. Configure the workload access settings according to the following instructions, as shown in the image 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.

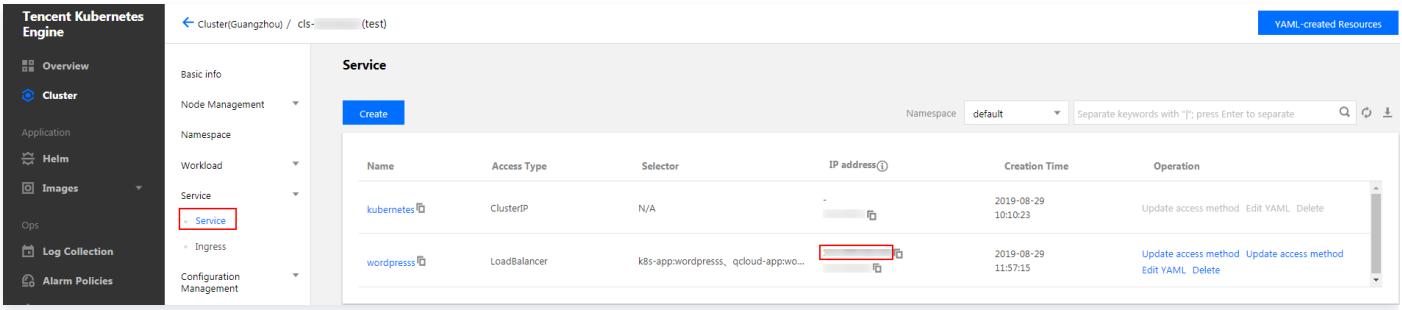
8. Click **Create Deployment**.

Accessing the WordPress service

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

Access using the CLB IP address

1. Click on Clusters in the left navigation bar to access 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 of the WordPress service as shown below:



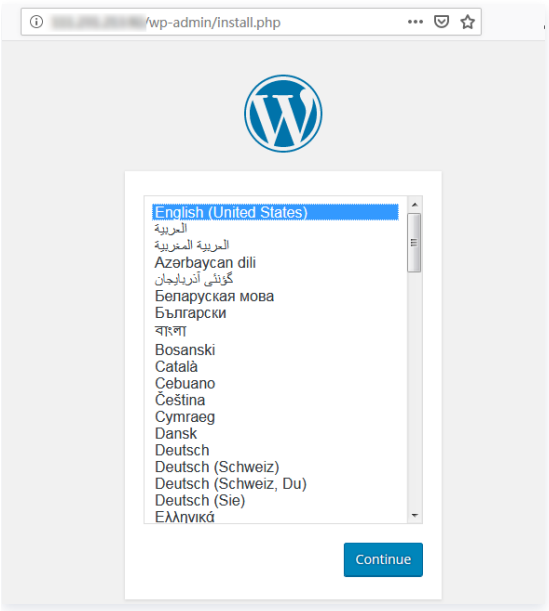
4. Paste the CLB IP address in the browser and press **Enter** to access the service.

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

Upon successful creation of the service, accessing it will take you directly to the WordPress server configuration page, 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: 2023-09-15 15:09:08

Scenario

WordPress with Single Pod demonstrates how to quickly create a WordPress service. The characteristics of the WordPress service created in this manner are as follows:

- 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.

Preparations

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

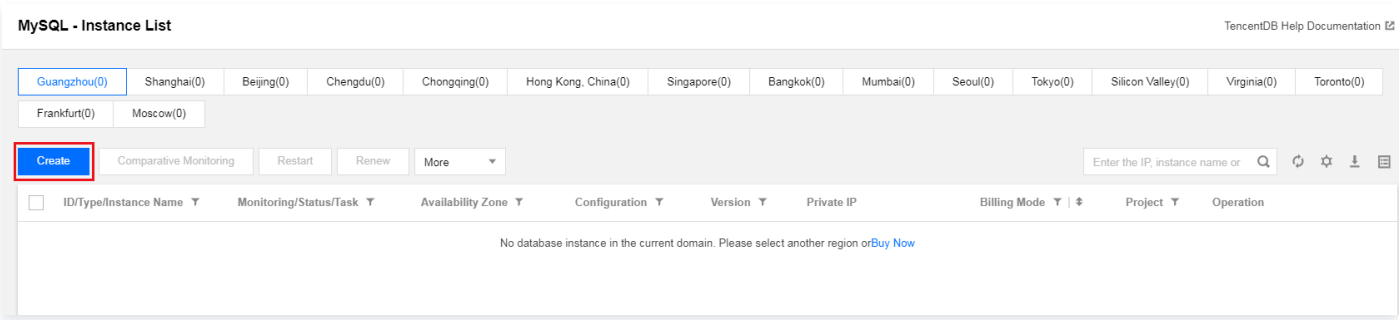
> **Note**
> The database used in the document is TencentDB for MySQL instance.

Instructions

Creating a WordPress service

Creating a TencentDB instance

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



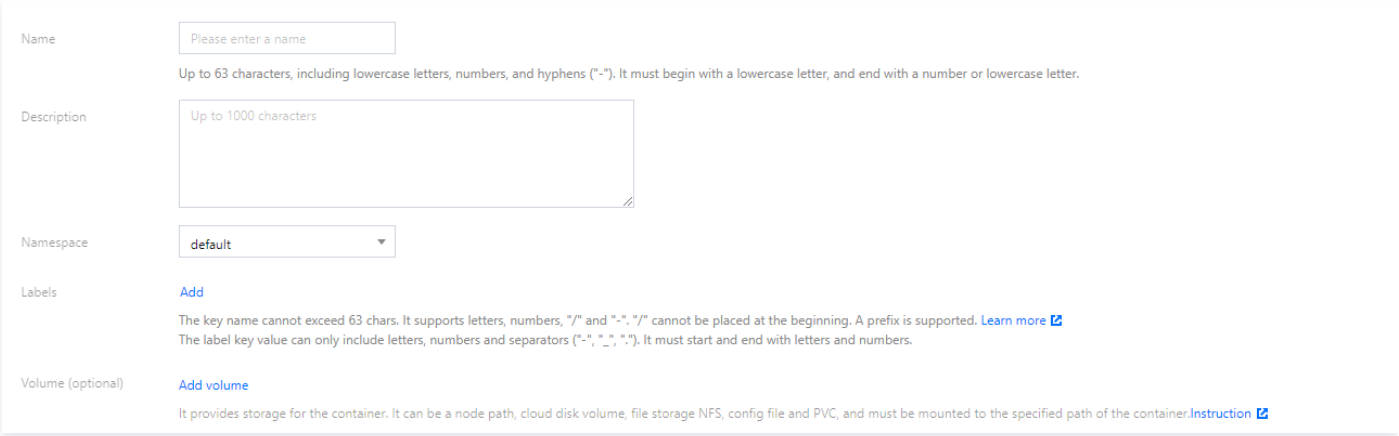
2. Select the configuration to purchase. For more information, see Overview.

> **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 more information, see Initializing MySQL Instance.

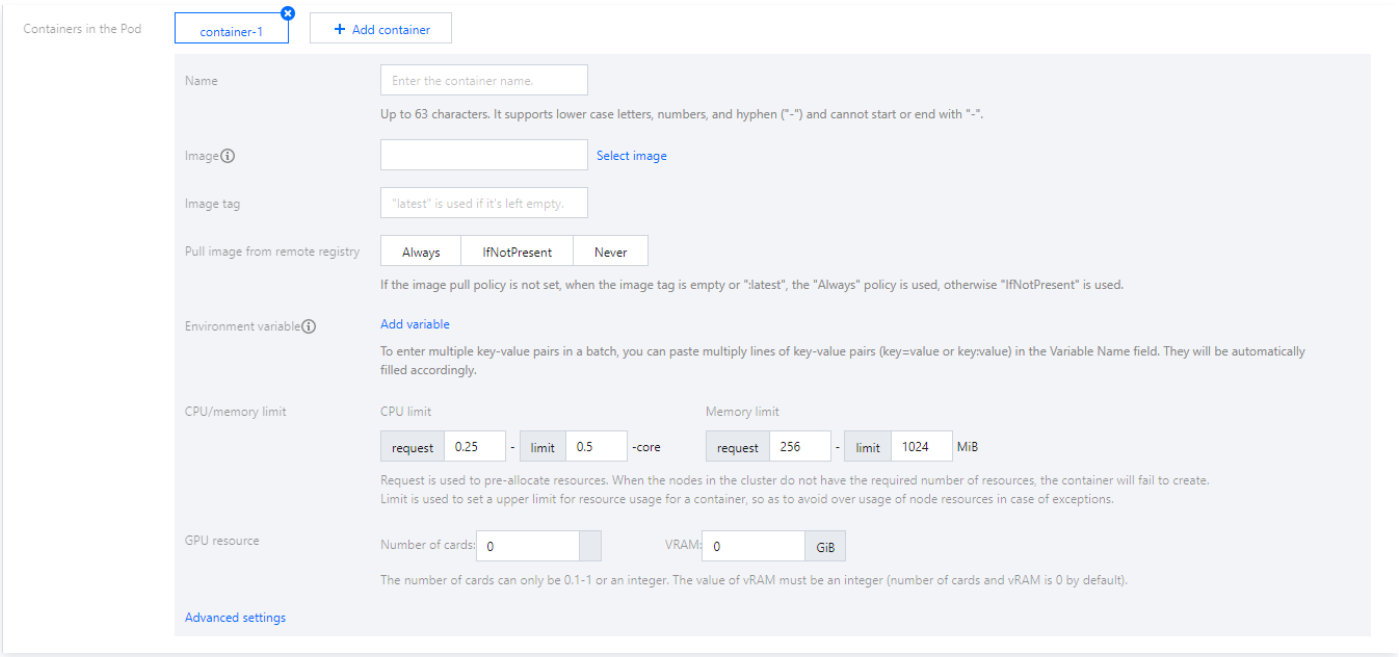
Creating a WordPress service that uses Tencent DB

1. Log in to the TKE console and click 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, set the basic workload information according to the following details, as shown in the image below:



- **Name**: Enter the name of the workload to be created, using "wordpress" as an example in this document.
- **Description**: specify related workload information.
- **Labels**: the default value is `k8s-app = wordpress` in this example.
- **Namespace**: select a namespace based on your actual requirements.
- **Volume**: Set up the workload volumes mounted based on your requirements. For more details, see Volume Management.

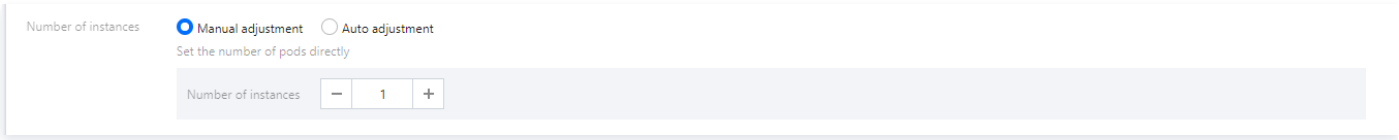
5. Refer to the following information to configure the "Instance Container" as shown in the image below:



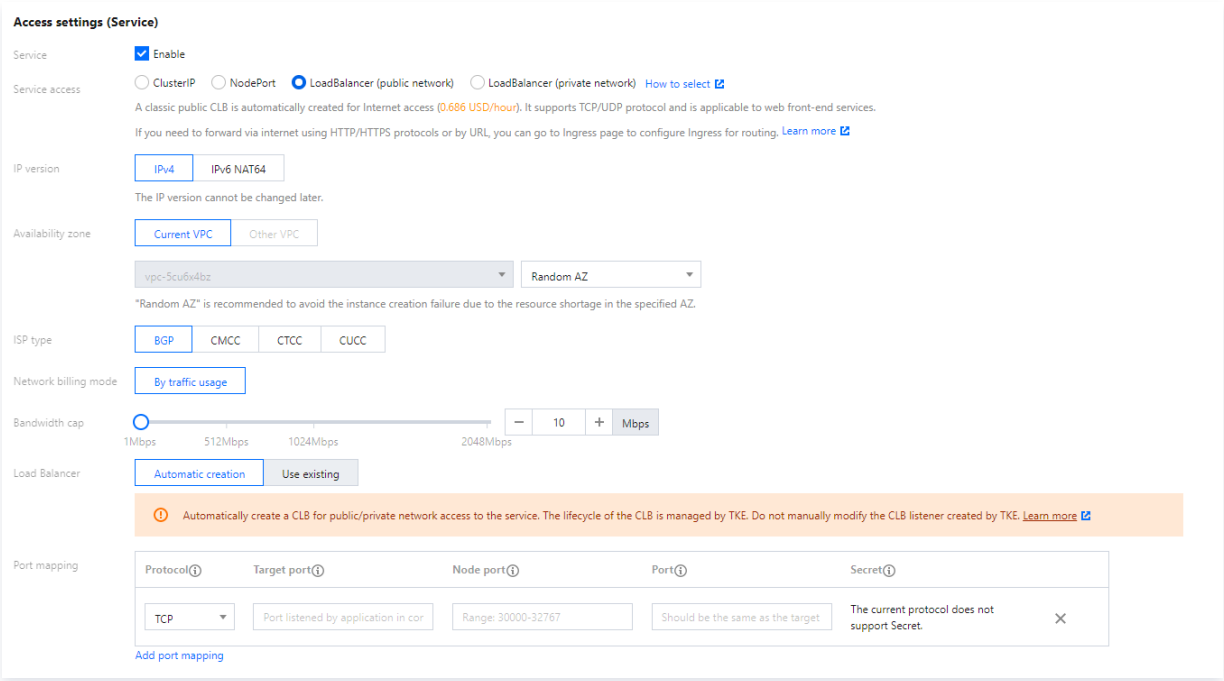
The main parameter information is as follows:

- **Name**: Enter the name of the container in the pod. Here, “test” is used as an example.
- **Image**: enter `wordpress`.
- **Image tag**: Use the default value `latest`.
- **Pull image from remote registry**: Three policies (`Always`, `IfNotPresent`, and `Never`) are available. Select a policy as required. In this example, **the default policy is used**.
- **Environment variable**: Enter the following configuration information in sequence:
  WORDPRESS_DB_HOST = Private IP of TencentDB for MySQL
  WORDPRESS_DB_PASSWORD = Password set during initialization

6. In "Number of Instances", set the number of service instances based on the following information. This article uses **Manual adjustment** as an example, with the instance quantity set to 1. As shown in the image below:



7. Follow the instructions below to configure workload access settings, as shown in the following image:



- **Service**: Select "Enable".
- **Service access**: Select "LoadBlancer(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.

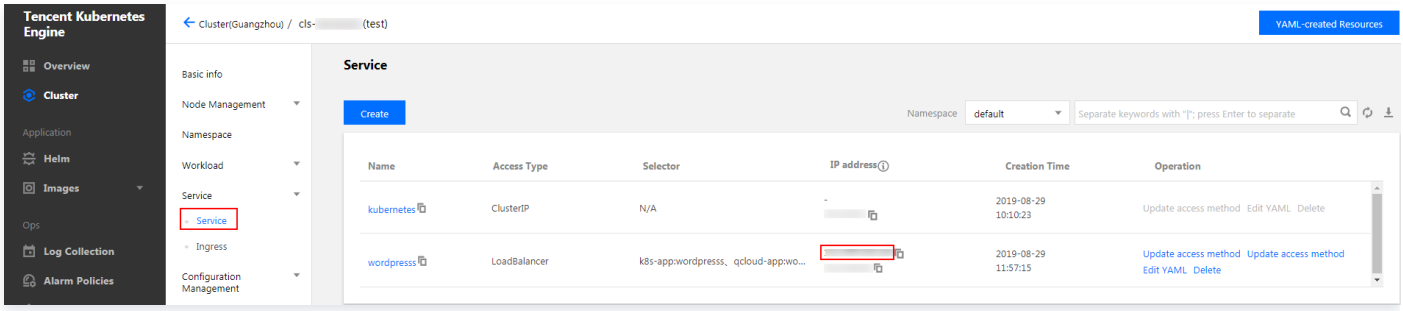
8. Click **Create Deployment**.

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. Click on Clusters in the left navigation bar to enter 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 of the WordPress service as shown below:



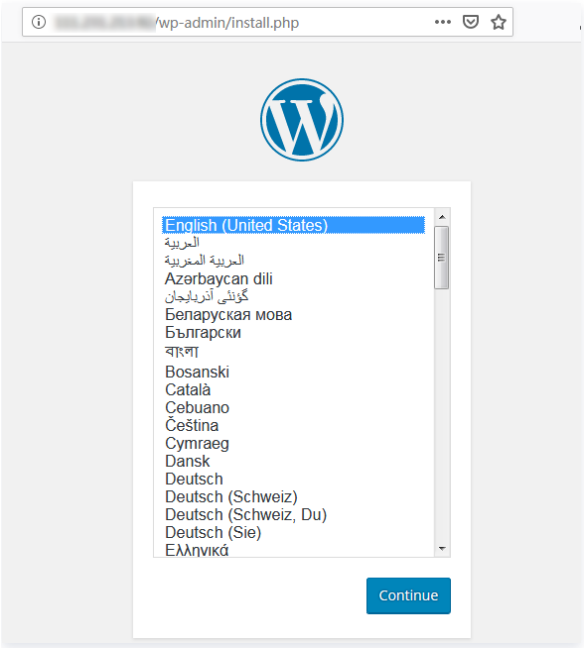
4. Paste the CLB IP address in the browser and press **Enter** to access the service.

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

Upon successful creation of the service, accessing it will take you directly to the WordPress server configuration page, 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: 2023-09-15 15:10:18

Scenario

This document describes how to use Tencent Cloud TKE to construct a simple Web application.
Web applications are divided into the following two parts:

- Frontend service, used to handle query and write requests from clients.
- Data storage service, which uses redis to store data written into the storage to redis-master, while read operations access redis-slave. Redis-master and redis-slave ensure data synchronization through master-slave replication.

This application is an example provided by the Kubernetes project. For more information, please refer to the Guestbook App.

Preparations

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

Instructions

Creating redis-master Service

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click on the cluster ID where you want to create the application, navigate to the cluster's basic information page, and click **Create** in **Workload > Deployment**.
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**: This document uses redis-master 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 document, "master" is used as an example.
   - **Image**: Enter `ccr.ccs.tencentyun.com/library/redis`.
   - **Image Version (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 Method**: Select "Access within the cluster only".
   - **Port Mapping**: Select TCP protocol, and set both the service port and container port to 6379. Other services can access the master container through the service name redis-master and port 6379.
6. Click **Create Deployment**.

Creating redis-slave Service

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click on the cluster ID where you want to create the application, navigate to the cluster's basic information page, and click **Create** in **Workload > Deployment**.
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, using redis-slave as an example in this document.
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 document, "slave" is used as an example.
   - **Image**: Enter `ccr.ccs.tencentyun.com/library/gb-redisslave`.

- **Image Version (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`.

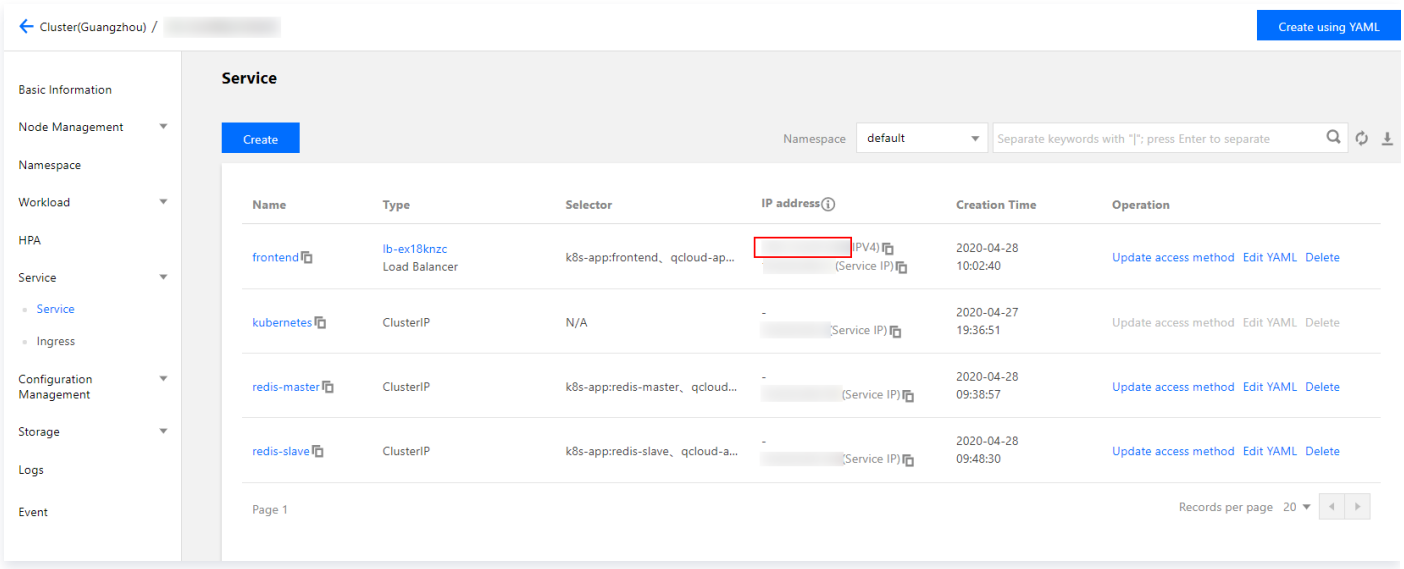
5. Configure **Access Settings (Service)** for the workload as instructed below.
   - **Service**: Select "Enable".
   - **Service Access Method**: Select "Access within the cluster only".
   - **Port Mapping**: Select TCP protocol, and set both the service port and container port to 6379. Other services can access the master container through the service name redis-master and port 6379.
6. Click **Create Deployment**.

Create Frontend Service

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click on the cluster ID where you want to create the application, navigate to the cluster's basic information page, and click **Create** in **Workload > Deployment**.
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, using frontend as an example in this document.
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 document, "frontend" is used as an example.
   - **Image**: Enter `ccr.ccs.tencentyun.com/library/gb-frontend`.
   - **Image Version (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 protocol, and set both the service port and container port to 80. Users can access the frontend container by visiting the load balancer IP through a browser.
6. Click **Create Deployment**.

Verify Web Application

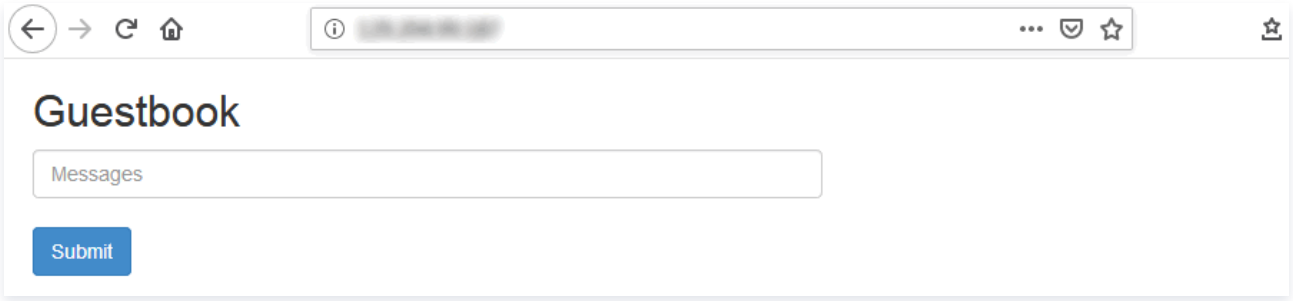
1. Go to the cluster details page, and choose **Services and Routes** > **Service** in the left sidebar.
2. On the Service page, copy the CLB IP of the frontend service as shown below:



Note

- When creating redis-master and redis-slave services, the access mode is set to **cluster-internal access only**, which means the services only have a private IP and can only be accessed by other services within the cluster.
- When creating the frontend service, because the access mode **Via Internet** was set, this service has a cloud load balancer (public IP) and a private IP, so it can be accessed by other services in the cluster and can also be accessed via public network.

3. Access the frontend service's load balancing IP using a browser. If the page returns as shown below, it indicates that the frontend service can be accessed normally.



4. Enter any string in the input box and click **Submit**. You will find that the entered record has been saved and displayed at the bottom of the page. Refresh the browser page and revisit the service IP address. The original input data still exists, indicating that the entered string has been saved to redis.

Development Practices

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

- If it is a set command, it takes the key and value in the parameters, and links it to the redis-master service. It also sets the key and value in redis-master.
- If it is not a set command, it links it to the redis-slave service, obtains the parameter key and corresponding value, and returns it 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 redis-master and redis-slave services, it connects to the **service name and port**. The built-in cluster DNS service resolves the service name into the corresponding service IP and performs load balancing based on the service IP. For example, if the redis-slave service has three instances, when accessing the redis-slave service, connect directly to redis-slave and port 6379. The DNS will automatically resolve redis-slave into the service IP of redis-slave (i.e., a floating IP, similar to the load balancer's IP) and perform load balancing based on the service IP of redis-slave, directing the request to a specific redis-slave service instance.
- Container environment variable settings:
  - **Use default setting (recommended setting)**: When the frontend container runs, if it reads the pre-set GET_HOSTS_FROM environment variable as dns, then it directly connects 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.