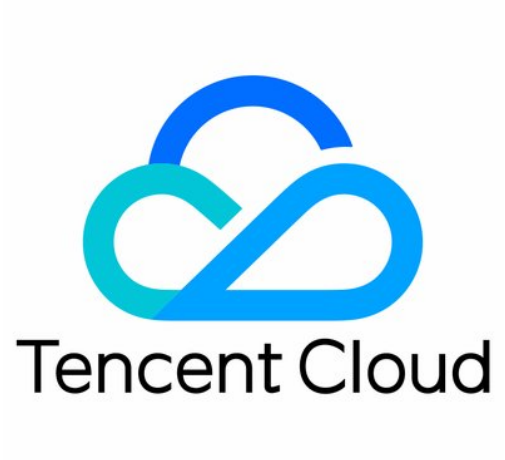
API Gateway

Best Practices

Product Documentation



Contents

Best Practices
Making Serverless Service Available quickly through API Gateway

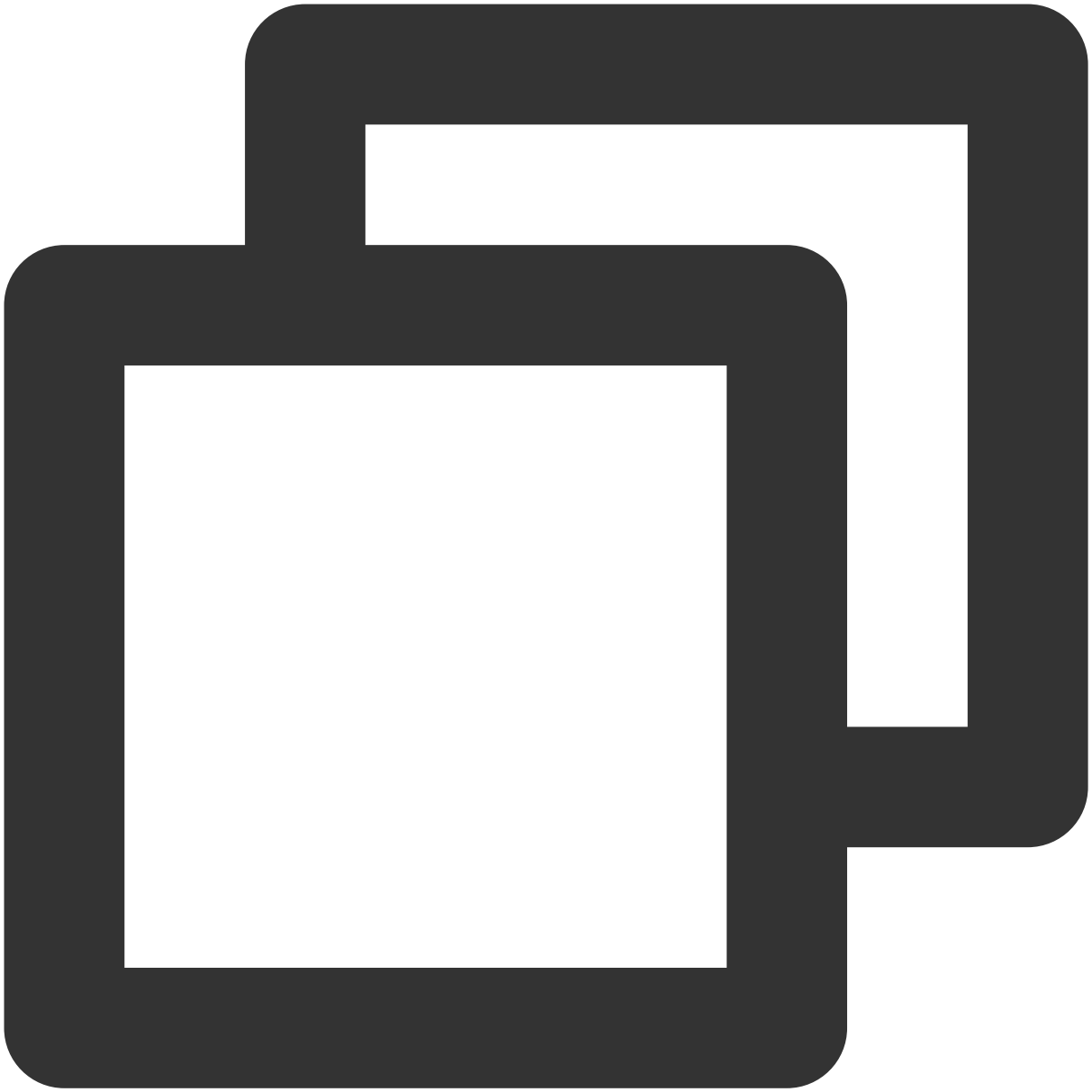
Last updated：2023-12-22 10:02:23

Overview

Serverless is a popular architecture in recent years. Through the Serverless function computing platform, you can focus on the core logic and run your code without purchasing and managing servers. In the Serverless mode, using the API Gateway can make the service available to external users, and realize advanced features such as security protection, traffic throttling, log monitoring, releasing in the cloud market, and automatic generation of SDK and documents.
Tencent Cloud API Gateway is highly integrated with Tencent Cloud SCF. This document describes how to build a website quickly by using the API Gateway as the entry to configure dynamic APIs through SCF and store static resources through COS.
With this method, you can use the API Gateway to make the Serverless service available quickly in the cloud.

Prerequisites

Before building a website, you need to go to the API Gateway official repository to download the website source code, which contains a skeleton HTML file and static resources such as pictures, CSS files and JS files.
The directory structure of the downloaded file is as follows:



```
├── client                                  // Root directory of the project
│   ├── static                              // Static resource
│   │   ├── background.jpg                  // Underground picture of the websi
│   │   ├── favicon.ico                     // Website icon
│   │   ├── index.js                        // Script file
│   │   ├── style.css                       // Style file
│   ├── index.html                          // Website homepage
```

Directions

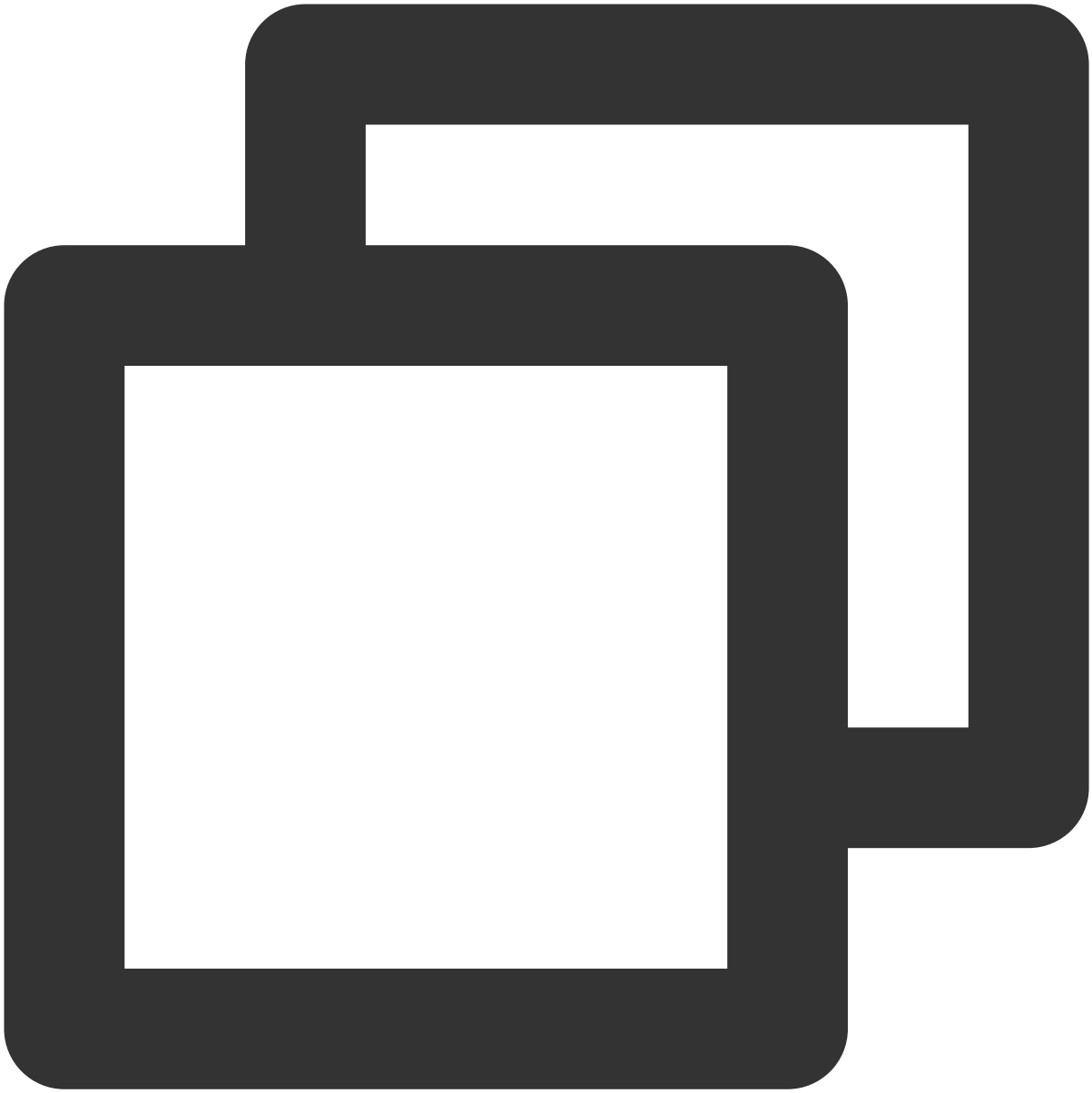
Step 1. Create a COS bucket to store static resources

1. Log in to the COS console. Create a bucket based on the following figure. For more information, see Creating Buckets.
2. Upload the website source code in the bucket. The directory structure must be consistent with the original file. For more information, see Uploading Objects.

Step 2. Create a SCF

1. Log in to the SCF console. Create a SCF using the "hello world" template. For more information, see Create and Update a Function.
2. Modify the SCF code and return simple JSON data.

The SCF code used is as follows:



```
'use strict';
exports.main_handler = async(event, context, callback) => {
    return {
        data: 'hello world' // hello world can be replaced with any content
    }
};
```

Step 3. Create an API Gateway service

1. Log in to the API Gateway console. Create an API Gateway service based on the following figure. For more information, see Creating Services.
2. Click the name of the service in the service list.
3. Click **Manage API** to go to the API management page. In the next step, you need to create three APIs on this page, with each API points to the corresponding backend resource.

Step 4. Configure three APIs

1. Click **Create** to create the first API. It is used to obtain the HTML page of the website. The frontend path is configured as "/". For more information, see API Creation Overview.
The "index.html" that points to the COS bucket is configured for the backend.
2. Click **Create** again to create the second API. It is used to obtain static resources. The frontend path is configured as "^~/static".
The "/static" path that points to the COS bucket is configured for the backend.
3. Click **Create** again to create the third API. It is used to obtain dynamic data. The frontend path is configured as "/fetchData".
Enter the corresponding SCF name in the backend configuration.

Step 5. Release the service and access the website

1. On the **Service information** tab of the service details page, click **Release** in the upper-right corner to release the service to the "Release" environment.
2. On the **Service information** tab of the service details page, view the "public network access address". Click the "Copy" icon to copy the address.
3. Paste the "public network access address" in the address bar of a browser. Press "Enter" to access the deployed website.
4. Click **Obtain data** to initiate a XHR call. The website will return the pre-defined JSON data string through the SCF.

Notes

In addition to access the website you build through the default domain name provided by the API Gateway, you can also bind an custom domain name to the service. After binding, you can access the website through the custom domain name. For more information, see Configuring a Custom Domain Name.

Integrating WAF to API Gateway for Security Protection

Last updated：2023-12-22 10:02:33

Overview

Tencent Cloud Web Application Firewall (WAF) is an AI-based one-stop web service protection solution.
This document describes how to integrate WAF to the API Gateway to protect your APIs.

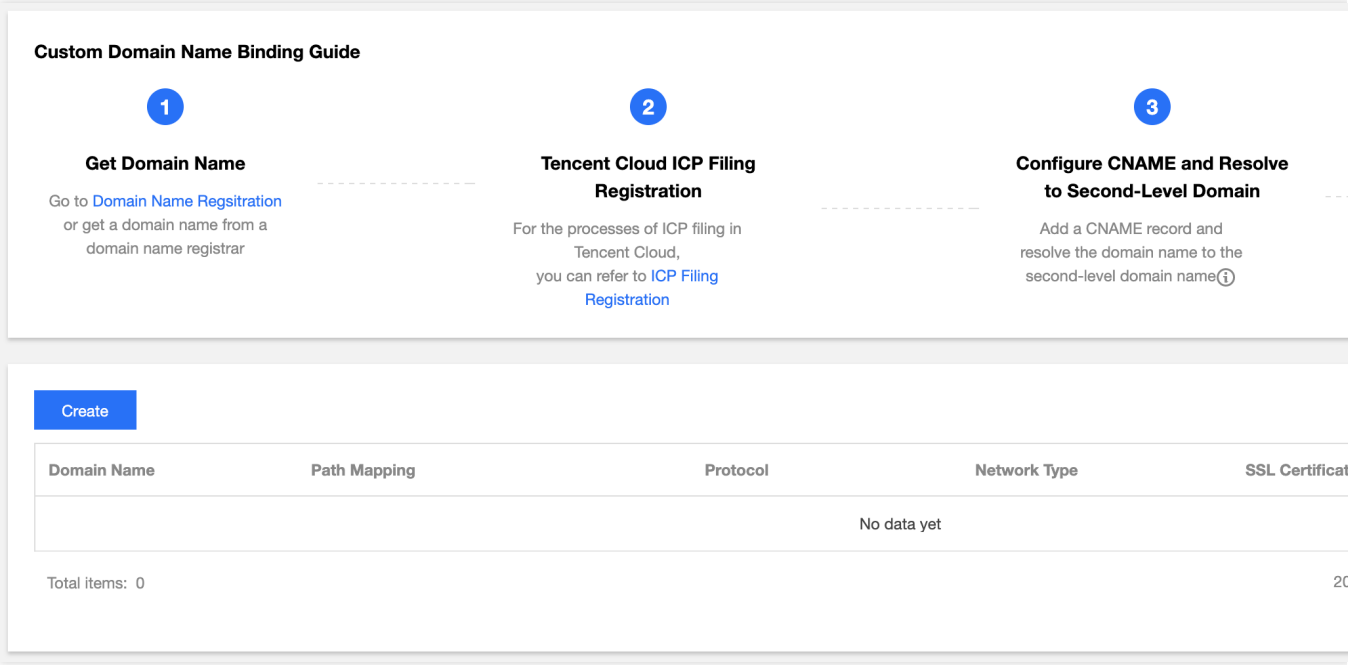
Prerequisites

You have activated Web Application Firewall.
You have released APIs using the API Gateway.

Directions

Step 1. Bind a custom domain name in the API Gateway console

For more information about how to bind a custom domain name in the API Gateway console, see Custom Domain Name and Certificate.

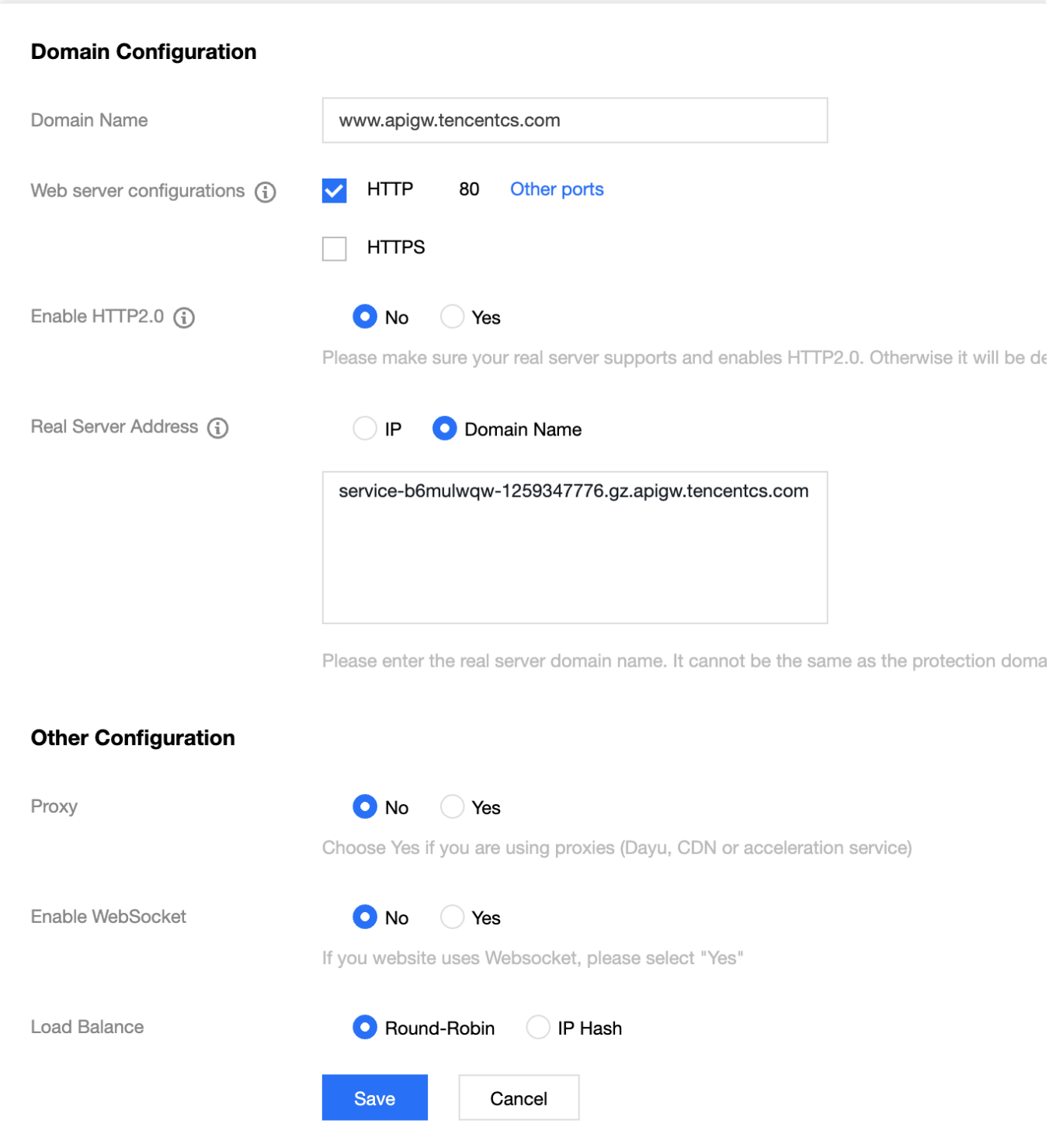


Note:

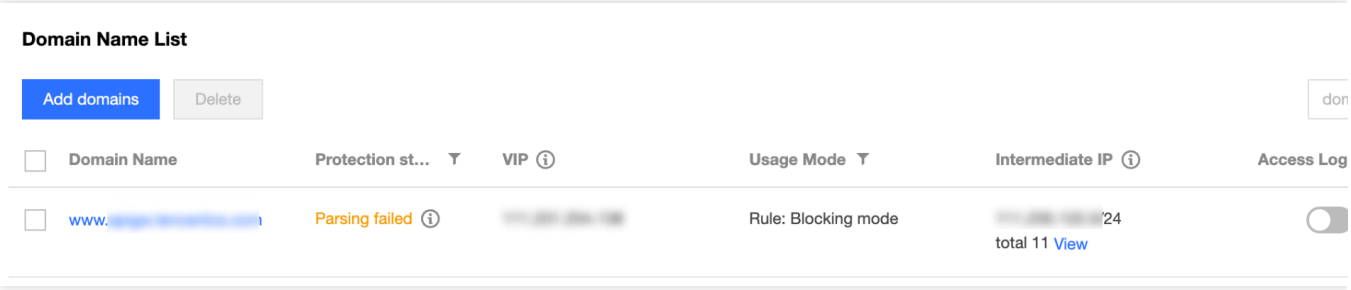
When a custom domain name is bound to the API Gateway console, the system will check whether you have configured CNAME and resolved it to the service subdomain name. Be sure to first configure CNAME and resolve it to the subdomain name of the API Gateway, modify the CNAME record, and point the custom domain name to the WAF domain name.

Step 2. Configure WAF

1. Log in to the WAF console.
2. Click **Web Application Firewall** > **Defense Settings** in the left sidebar.
3. Click **Add domains** at the top of the **Domain name list** module.
4. Select **Domain name** as the real server address and enter the subdomain name of the API Gateway. Complete the other configurations.



5. Click **Save**. The domain name should now be in the “CNAME record not configured” status.



Step 3. Modify the CNAME record

1. Modify the CNAME record and resolve the custom domain name to the WAF domain name.
2. Log in to WAF console. Click **Web Application Firewall** > **Defense Settings** in the left sidebar.
3. Click the refresh icon in the **Protection status** column. The status should change to **Normal protection**.

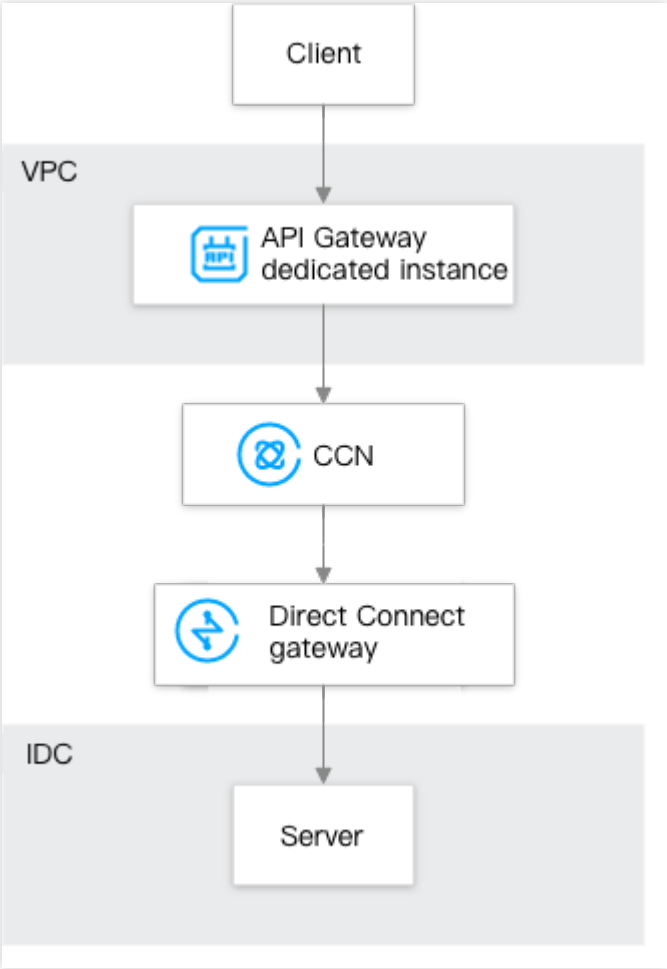
Accessing Resources in IDC via API Gateway Dedicated Instance

Last updated：2023-12-22 10:02:44

Overview

When you use a hybrid cloud architecture, your business may be deployed in both public and private clouds, but a unified access layer is required, which serves as the ingress and egress of traffic and supports the processing of all non-business features such as authentication and traffic throttling.
An API Gateway dedicated instance runs in a VPC and supports forwarding client requests to various services deployed in the VPC or local IDC or on the public network. It is deeply integrated with common backend services to provide a productized connection method. Therefore, it is very suitable as a unified access layer in complex network environments. This document describes how to connect to backend resources in an IDC by using an API Gateway dedicated instance.

Solution Strengths



The API Gateway dedicated instance can forward requests to the backend resources deployed in the VPC and local IDC and on the public network at the same time, seamlessly connecting the cloud and local systems and enabling smooth cloudification.

The rich features provided by API Gateway can also be used, such as IP access control, traffic throttling, and log monitoring.

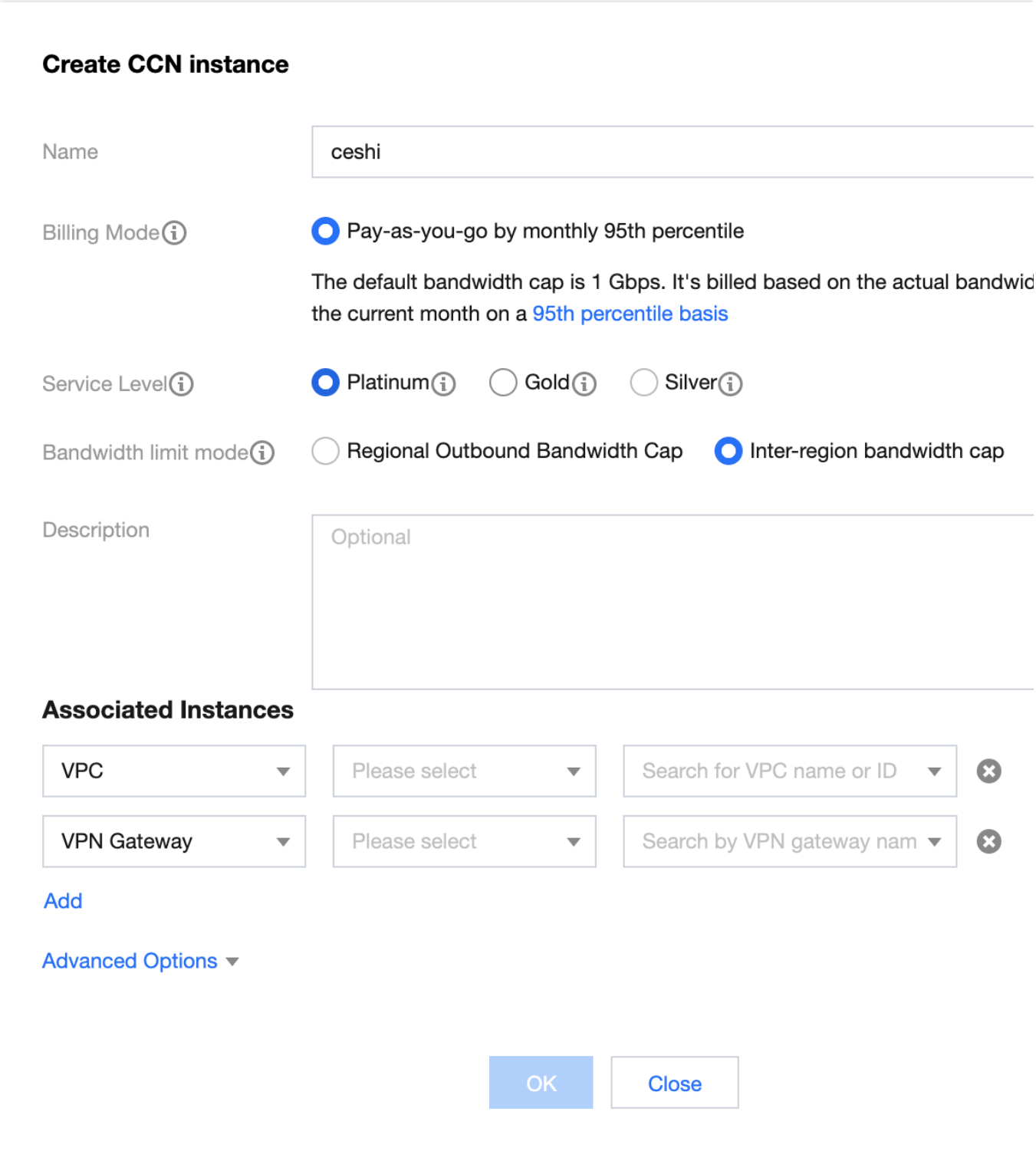
Resources on the private network can be interconnected with each other through CCN, fine-grained routing is supported to guarantee the quality, and diversified tiered pricing is supported to reduce the costs.

Directions

Step 1. Create a CCN instance and associate it with a network instance

1. Log in to the VPC console.
2. Click **CCN** on the left sidebar to go to the CCN management page.
3. Click **+New**.
4. In the box that pops up, enter the name and description for the CCN instance. Select its billing mode, service quality, and bandwidth limit mode.

5. Associate the IDC's Direct Connect gateway with the VPC.



Step 2. Purchase an API Gateway dedicated instance

1. Log in to the API Gateway console. Select **Instance** on the left sidebar.
2. Click **Create** to go to the API Gateway dedicated instance purchase page.

3. Select and enter the instance configuration information.

Note:

The VPC configuration of the dedicated instance should be the same as that of the VPC instance associated with the CCN instance created in step 1.

4. Click **Buy now** and make the payment.

Step 3. Create a service and API under the instance

1. Log in to the API Gateway console. Select **Service** on the left sidebar.
2. Click **Create** and select **Dedicated** as the instance type.
3. In the pop-up window for instance selection, select the dedicated instance purchased in step 1 and click **Submit**.
4. Click the name of the created service in the service list to enter its API management page.
5. Click **Create** to enter the API creation page and set the configuration items. Select public URL/IP as the API backend type, enter the IDC's private IP, port, and path to be bound as the backend address, and click **Submit**.
6. Request the created API, and you will see that the backend resources in the IDC can be accessed through API Gateway.

Quick Access to TEM Application via API Gateway

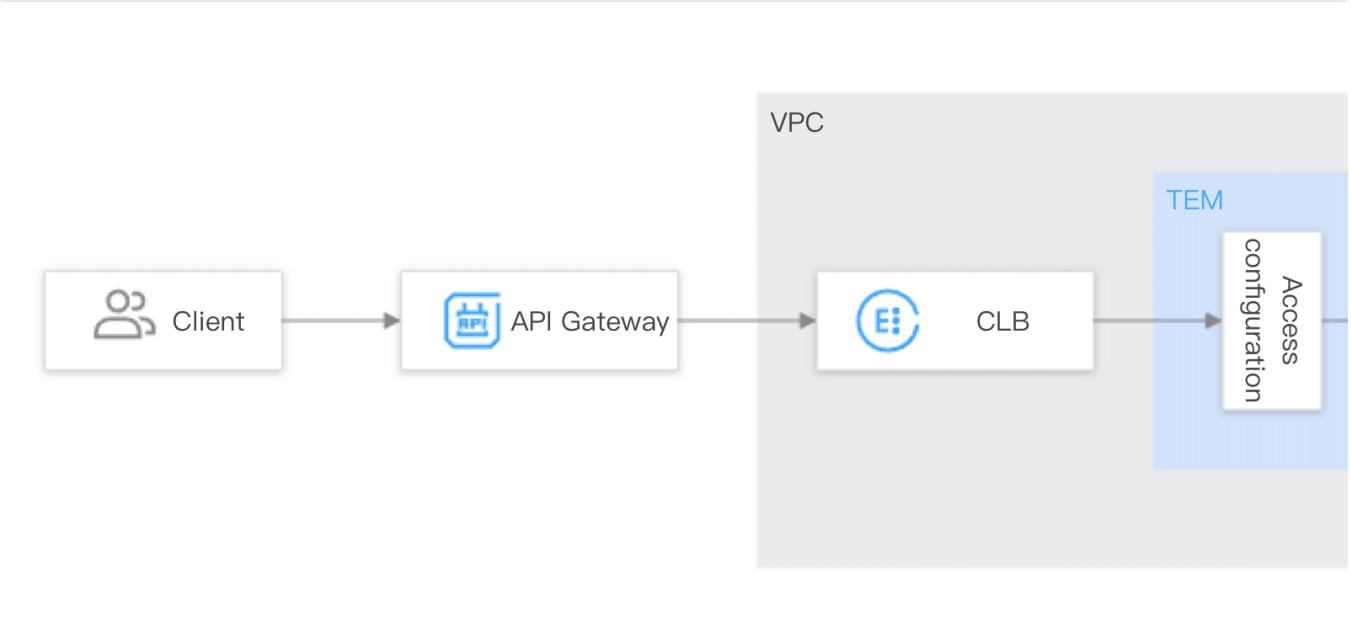
Last updated：2023-12-22 10:02:53

Introduction on TEM

Tencent Cloud Elastic Microservice (TEM) is a Serverless platform designed for microservice applications. It perfectly combines serverless and microservice to provide out-of-the-box microservice solutions. TEM embraces the concept of open source, makes it possible to cloudify applications with zero modifications required, offers a wide range of capabilities such as application hosting, service registration and discovery, microservice governance, and multidimensional monitoring, and well supports Eureka and ZooKeeper registries. Moreover, TEM helps you create and manage pay-as-you-go cloud resources without Ops costs. For more information, see Tencent Cloud Elastic Microservice.

Overview

This document describes how to quickly use API Gateway to access a TEM application and manage its APIs. With the combination of API Gateway and TEM, you can enjoy the advanced capabilities of API Gateway such as traffic throttling, authentication, and caching for better business outcomes.



Prerequisites

Log in to the TEM console, create an environment, and create and deploy an application.

Directions

Step 1. Configure VPC access for the TEM application

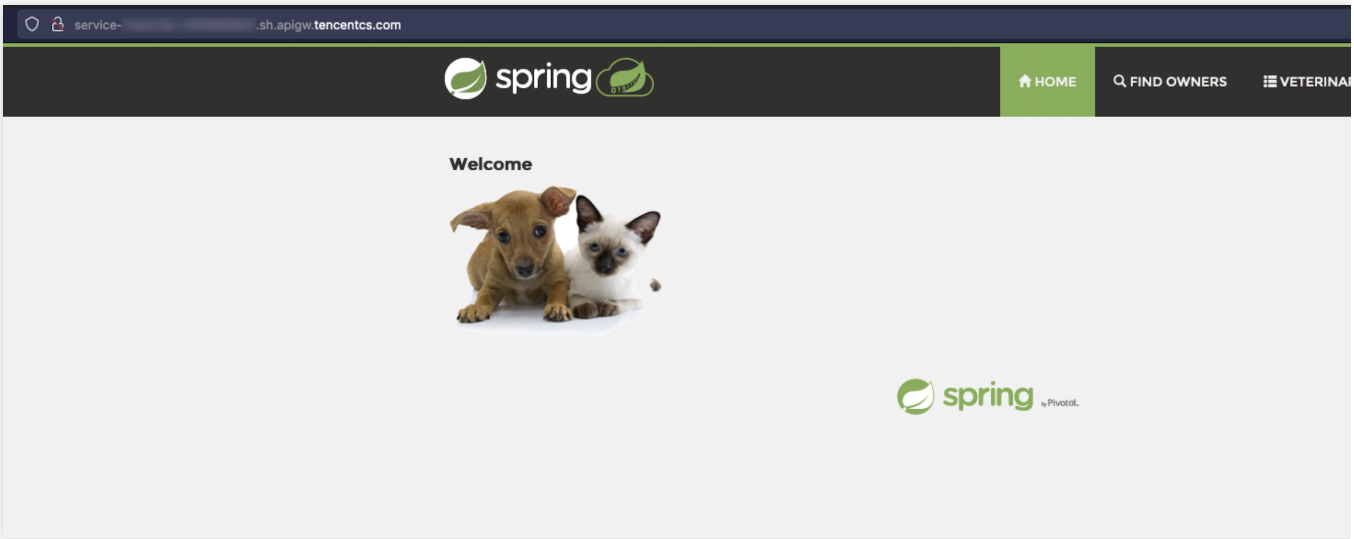
1. Log in to the TEM console, click **Application Management** on the left sidebar, and click the target application to enter the application details page.
2. Click **Edit and update** in the **Access configuration** section to enter the application access configuration page.
3. Select VPC access (layer-4 forwarding), select the subnet, protocol, container port, and application listening port, and click **Submit**. At this point, TEM will automatically create a layer-4 forwarding VPC CLB instance for you.

Step 2. Create an API Gateway service and bind it to the TEM application

1. Log in to the API Gateway console and click **Service** on the left sidebar to enter the service list page.
2. Select the same region as the TEM application and click **Create** in the top-left corner to create a service.
When creating the service, you can select the frontend type (HTTP, HTTPS, or HTTP/HTTPS), access mode (VPC or public network), and instance type (shared or dedicated).
Note:
For more information on the selection of instance type, see Instance Selection.
3. Click the API Gateway service ID to enter the API management page and click **Create API**.
4. In the **Frontend configuration** step, enter the API name, select **HTTP&HTTPS** as the frontend type, / as the path, **ANY** as the request method (to include all requests), and **Authentication-free** as the authentication type, and click **Next**.
5. In the **Backend configuration** step, select **VPC resources** as the backend type, select the VPC where the TEM application deployment environment is located, set the backend domain name, select the CLB instance automatically created by the TEM application (named "cls-xxxdefault{TEM application name}"), select the corresponding listener (i.e., the port mapping set in the previous step), and enter `/` as the backend address.
6. At this point, you can see the API you configured and access your TEM application at the default domain name provided by API Gateway.

Step 3. Access the TEM application through API Gateway

Call the API Gateway API created in step 2 to access the TEM application through the API Gateway.



Notes

In order to ensure that applications can access API Gateway in a non-intrusive manner, we recommend you bind an API Gateway service to only one TEM application and keep the frontend address and backend address the same. If they are both `/`, all APIs can be blocked. You can also make separate configurations for some of your application's APIs.

You can refer to Plugin Usage to bind the plugin to the API Gateway API that connects the backend with the TEM.

Integrating ECDN to API Gateway for Acceleration

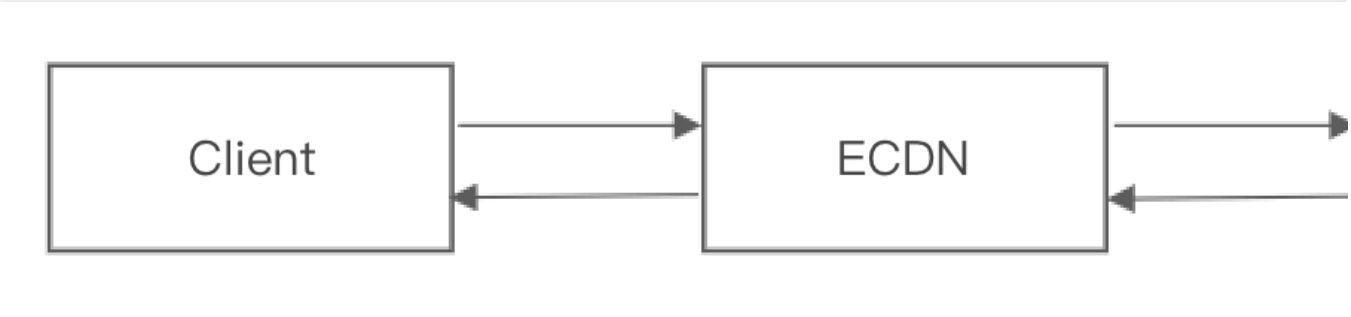
Last updated：2023-12-22 10:03:04

Overview

Tencent Cloud Enterprise Content Delivery Network (ECDN) integrates static edge caching and dynamic origin-pull route optimization. Through Tencent's global nodes and based on more than ten years of technical practice on the QQ platform, ECDN provides one-stop acceleration service with high reliability and low latency.
Connecting the API Gateway with ECDN can help you solve problems such as slow response, high packet loss and unstable service caused by cross-ISP, cross-boarder and cross-network data transfer.

Relevant Products

API Gateway
ECDN



Directions

1. Log in to the API Gateway console. Create an API Gateway service. A default access address is generated.
2. Configure a custom domain name in the ECDN console.
Enter the custom domain name in the "Add acceleration domain name" field.
Select "Origin server domain name" as the origin server type.
Select "Optimal origin-pull" as the origin-pull policy.

Enter the default access address generated in step 1 (without the protocol and port number) as the origin-pull address.

Select the origin-pull protocol and acceleration region as needed.

Click **Save** to complete the configuration. A CNAME domain name will be generated in the CDN console after a while.

3. Modify the origin server Host.Select the just configured domain name in the domain name list in ECDN console.

Click it to go to the details page.

Change the origin server Host to the default public network access address generated in step 1.

4. Configuring CNAME.

Configure the CNAME domain name for the connected domain name.

```
(base) [redacted]@MacBook-Pro ~ dig yizhanwu.flyfly.wang

; <<>> DiG 9.10.6 <<>> [redacted]
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39627
;; flags: qr rd ra; QUERY: 1, ANSWER: 15, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;[redacted]          IN      A

;; ANSWER SECTION:
[redacted].   409     IN      CNAME   [redacted].dsa.dnsv
yizhanwu.flyfly.wang.dsa.dnsv1.com. 409 IN CNAME 3ts0ixbj.sched.d0.tdnsv5.com
3ts0ixbj.sched.d0.tdnsv5.com. 60 IN     A       [redacted]
3ts0ixbj.sched.d0.tdnsv5.com. 60 IN     A       [redacted]
3ts0ixbj.sched.d0.tdnsv5.com. 60 IN     A       [redacted]
3ts0ixbj.sched.d0.tdnsv5.com. 60 IN     A       [redacted]
3ts0ixbj.sched.d0.tdnsv5.com. 60 IN     A       [redacted]
```

5. Initiate a call, and you can see that they are already connected.

API Gateway Providing the Access Capability for TKE

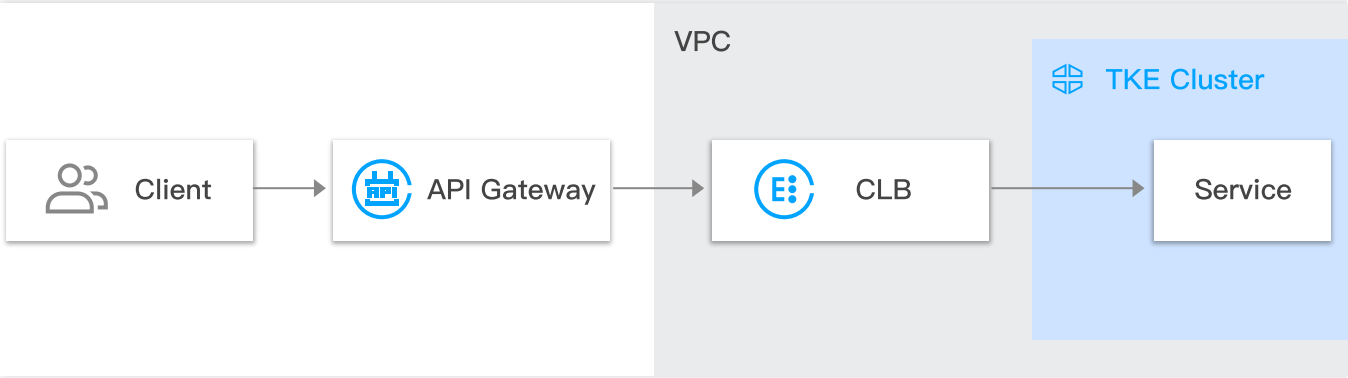
Last updated：2023-12-22 10:03:13

Introduction to TKE

Based on native Kubernetes, Tencent Kubernetes Engine (TKE) is a container-oriented, highly scalable, and high-performance container management service. Compared with a client's container service, TKE has core advantages such as its ease of use, flexible expansion, security, reliability, high efficiency, and low costs. For more information, see Tencent Kubernetes Engine.

Overview

As an open source platform for automated container operations, Kubernetes is a mainstream choice for developers. However, the access capability of Kubernetes clusters are not sufficient and cannot meet the requirements of large applications. Using API Gateway as the access layer of Kubernetes can significantly improve the access capability of Kubernetes clusters and empower Kubernetes clusters with advanced capabilities of API Gateway, adapting to more scenarios of more customers.



Prerequisites

You have activated Tencent Cloud services such as API Gateway, TKE, Cloud Load Balancer (CLB), Virtual Private Cloud (VPC), and Cloud Virtual Machine (CVM) and have permission to configure these services, as they will be used during the configuration process.

Directions

Step 1: Create a VPC

1. Log in to the VPC console.
2. In the left sidebar, click **Virtual Private Cloud** to access the VPC list page.
3. Click **+ New**. In the pop-up dialog box, set the parameters to create a VPC.

For more information, see Managing VPC Instances.

Step 2: Create a CVM

1. Log in to the CVM console.
2. In the left sidebar, click **Instances** to access the CVM instance list page.
3. Click **Create** to access the CVM purchase page.
4. Create a CVM by following the instructions in Creating Instances via the CVM Purchase Page.

Note:

When creating a CVM, select the VPC and the subnet created in Step 1 and retain the default values for the other parameters. In this example, a standard S5 CVM instance is created.

Step 3: Create a TKE cluster in the same VPC

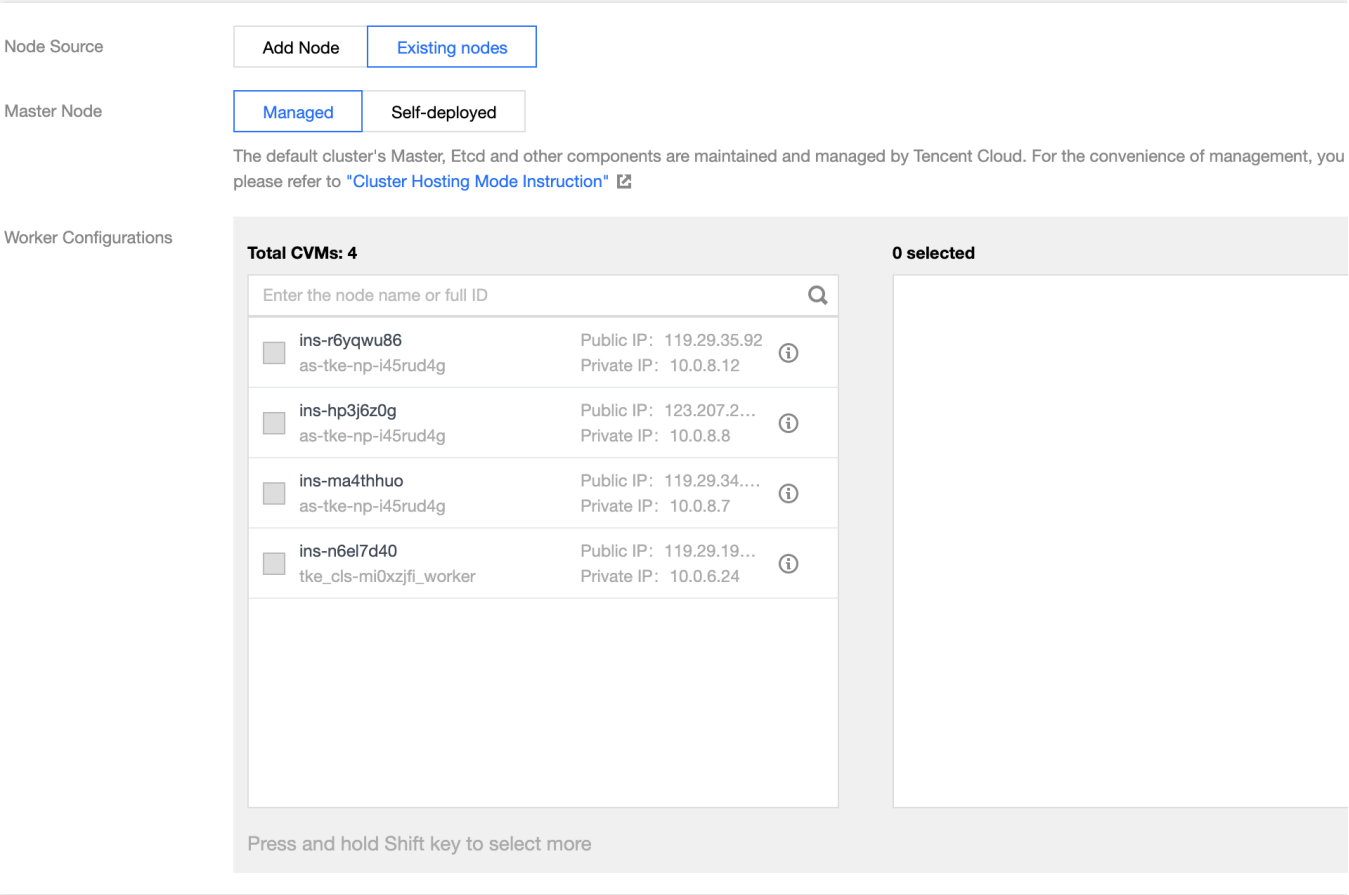
1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to access the TKE cluster list page.
3. Click **Create** at the top of the TKE cluster list. On the **Create Cluster** page, create a TKE cluster by following the instructions in Creating a Cluster.

Note:

When configuring the cluster information, set **Cluster network** to the VPC created in Step 1.

When selecting a model, set **Node Source** to **Existing nodes** and **Master Node** to **Managed**, and select the CVM created in Step 2 in the **Worker Configurations** area.

Retain the default values for the other parameters.



Step 4: Create a nginx service in the TKE cluster

1. Log in to the TKE console. In the left sidebar, click **Cluster** to access the TKE cluster list page.
2. Click the ID of the TKE cluster created in Step 3 to access the cluster details page.
3. In the left sidebar, click **Workload** -> **Deployment** to access the deployment list page.
4. Click **Create** at the top of the deployment list page to access the workload creation page.
5. Set the parameters on the workload creation page by following the instructions in Creating a Simple Nginx Service.

Note:

Select a DockerHub nginx image.

Set **Service Access** to **Via VPC**.

Set **Load Balancer** to **Automatic creation**.

In the **Port Mapping** area, set **Protocol** to **TCP**, and set both **Target Port** and **Port** to 80.

6. Click **Create Workload** to finish creating the Workload. TKE will automatically create the corresponding deployment and service.

Step 5: Create an API Gateway service

1. Log in to the API Gateway console.
2. In the left sidebar, click **Service** to access the service list page.

3. Click **Create** at the top of the service list. In the pop-up dialog box, create an API Gateway service by following the instructions in Creating Services.

Step 6: Create an API in the API Gateway service

1. In the API Gateway console, click **Service** in the left sidebar to access the service list page.
2. Click the name of the created API Gateway service to access the service details page.
3. Click the **Manage API** tab, and then click **Create** to access the API creation page.
4. Enter the frontend configuration, the backend configuration, and the response result, and click **Complete** to finish creating the API.

Note:

When entering the backend configuration, set **Backend Type** to **HTTP**, **VPC Info** to the VPC created in Step 1, **VPC resources** to **CLB**, and **Backend Path** to /.

Step 7: Open the private IP ranges of API Gateway to the internet

1. Log in to the CVM console. In the left sidebar, click **Security Groups** to access the security group list page.
2. Select a region and click **+ New**. In the pop-up dialog box, set the parameters and click **OK** to create a security group.
3. In the security group list, click the name of the created security group to access the security group details page. Click the **Security Group Rule** tab and then the **Inbound rule** tab to access the inbound rule list.
4. Click **Add a Rule**. In the pop-up dialog box, enter the following 5 private IP ranges of API Gateway: **9.0.0.0/8**, **10.0.0.0/8**, **100.64.0.0/10**, **11.0.0.0/8**, and **30.0.0.0/8**. Set **Protocol port** to **ALL** and **Policy** to **Allow** for the 5 private IP ranges, and click **Completed** to add the 5 inbound rules.
5. Return to the security group details page. Click the **Associate with Instance** tab and then the **Cloud Virtual Machine** tab. Click **Add Instances**. In the pop-up dialog box, associate the created security group with the CVM created in Step 2 to open the private IP ranges of API Gateway to the internet.

Step 8: Publish and test the API Gateway service

1. In the API Gateway console, click **Service** in the left sidebar to access the service list page.
2. Click the name of the created API Gateway service to access the service details page.
3. Click the **Basic Configuration** tab, and click **Publish** in the upper right corner of the page to publish the service to the **Publish** environment.
4. Request the API created in Step 6. If the nginx page is displayed, the access is successful.

Welcome to nginx!

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

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

Thank you for using nginx.