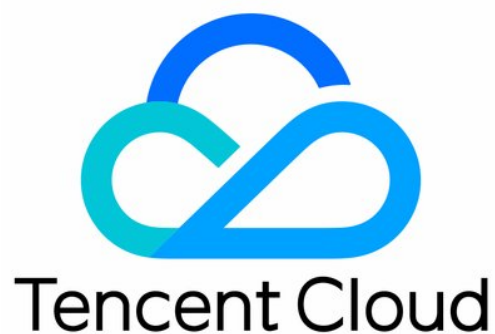


# **Tencent Cloud Elastic Microservice**

## **Practical Tutorial**

### **Product Documentation**



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Practical Tutorial

Use of GitHub Actions in TEM

Hosting a Static Website

Public Network Access of TEM Applications

TEM Application Access to Public Network (Through API Gateway)

TEM Application Failure Troubleshooting Guide

Quick Access to TEM Application Through API Gateway

Java Application Fine-Tuning

Migration from Java 8 to Java 11

# Practical Tutorial

## Use of GitHub Actions in TEM

Last updated : 2024-01-09 12:42:59

## Use of GitHub Actions in TEM

### GitHub Actions

TEM integrates world-class CI/CD tools to facilitate your use of GitHub workflows. You can learn more by referring to the [official documentation of GitHub Actions](#).

#### Note :

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD.

### Application release types supported by TEM

The TEM platform uses cloud native as its infrastructure, where all applications exist in the form of containers at runtime. TEM especially supports the release of JAR and WAR packages for Java applications and takes care of the build and management of images. For other languages, you need to build images on your own and push them to Tencent Cloud Image Registry.

### How to use

The following takes .NET as an example to describe how to use GitHub Actions.



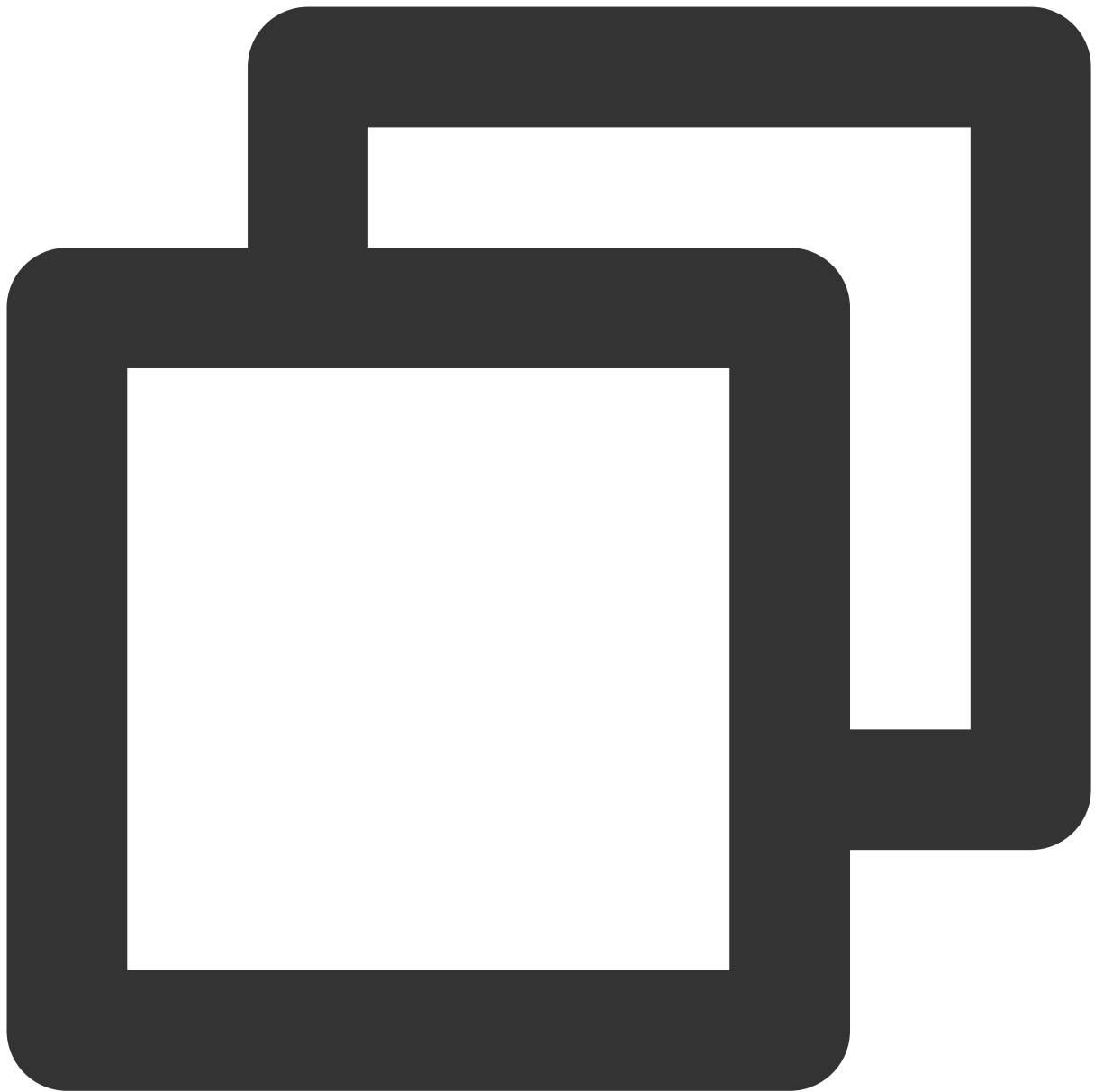
```
name: .NET

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:
    runs-on: ubuntu-latest
```

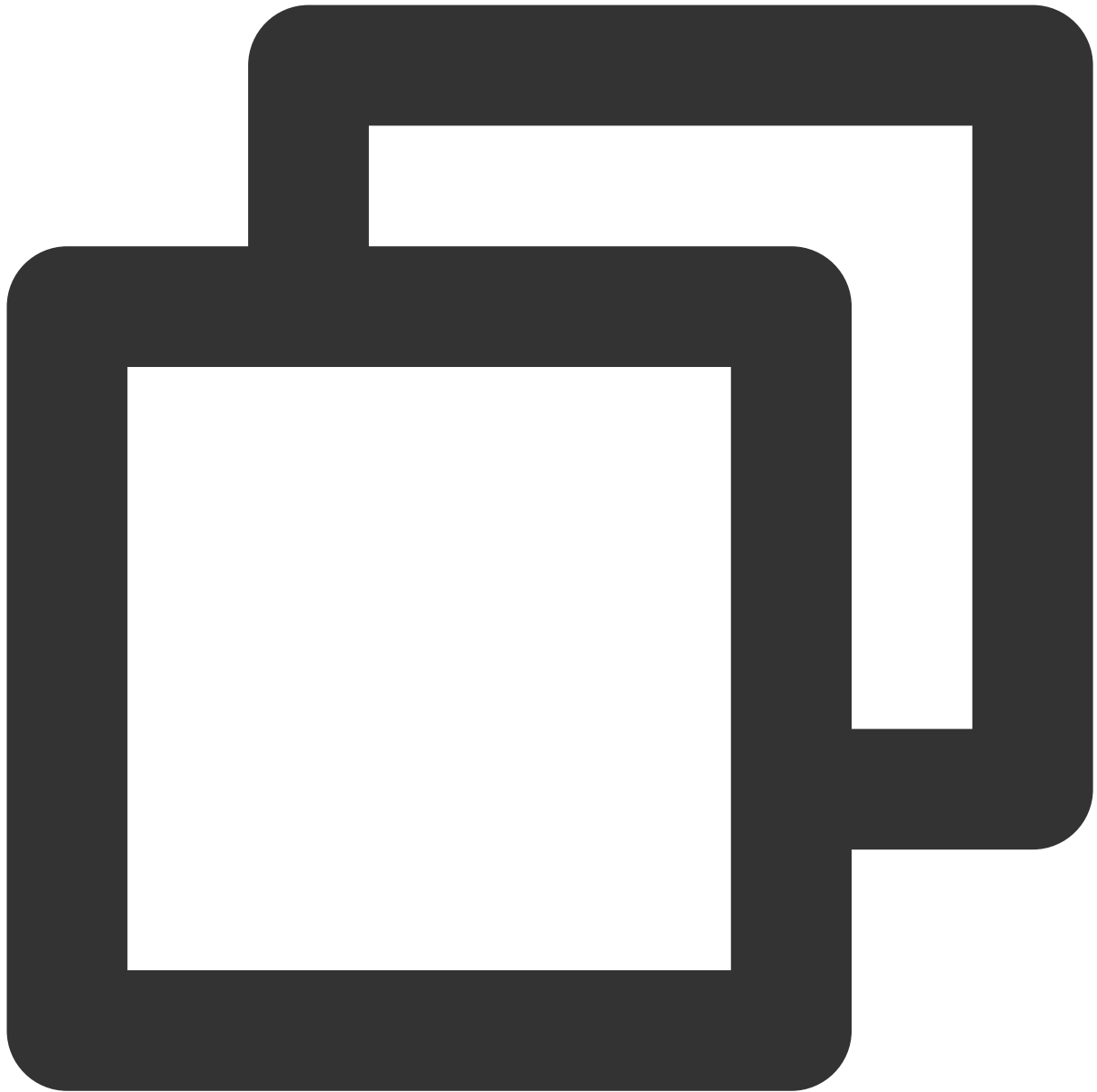
```
steps:
- uses: actions/checkout@v2
- name: Setup .NET
  uses: actions/setup-dotnet@v1
  with:
    dotnet-version: 5.0.x
- name: Declare some variables
  id: vars
  shell: bash
  run: |
    echo "::set-output name=sha_short::$(git rev-parse --short HEAD)"
- name: Build Code
  run: dotnet publish -o ./target
- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v1
- name: Login to Registry
  uses: docker/login-action@v1
  with:
    registry: ${ secrets.REGISTRY_URL }
    username: ${ secrets.REGISTRY_USERNAME }
    password: ${ secrets.REGISTRY_TOKEN }
- name: Build and push
  uses: docker/build-push-action@v2
  with:
    context: .
    push: true
    platforms: linux/amd64,linux/arm64
    tags: ccr.ccs.tencentyun.com/han_test/my-web-app:${ steps.vars.outputs.s
```

1. Your code repository should include the `dockerfile` file for use by the built action.



```
FROM mcr.microsoft.com/dotnet/aspnet:5.0
COPY ./target /app
WORKDIR /app
ENTRYPOINT ["dotnet", "myWebApp.dll"]
```

2. Here, `commitId` is used as the image tag, which makes it easier to confirm the runtime's application code version. If you don't need this, you can directly use the latest image tag.



```
git rev-parse --short HEAD
```

3. Tencent Cloud [Image Registry Personal Edition](#) requires your login information. The account information will automatically pop up when you open the Personal Edition page for the first time. The Enterprise Edition works in a similar way. You can find relevant documents by yourself.





```
- name: Login to Registry
  uses: docker/login-action@v1
  with:
    registry: ${ secrets.REGISTRY_URL }
    username: ${ secrets.REGISTRY_USERNAME }
    password: ${ secrets.REGISTRY_TOKEN }
```

4. Relevant keys can be managed by using the **Secrets** module on the **Repository Settings** page.

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Environments

Secrets

Actions

Dependabot

Pages

Moderation settings

Actions secrets







Secrets are environment variables that are encrypted. Anyone with collaborator access to this repository can view them. Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more.](#)

Environment secrets

There are no secrets for this repository's environments

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository. [Manage your environments and add environment secrets](#)

Repository secrets

 QCLOUD_REGION	Updated 13 days ago
 QCLOUD_SECRET_ID	Updated 13 days ago
 QCLOUD_SECRET_KEY	Updated 13 days ago
 REGISTRY_TOKEN	Updated 13 days ago
 REGISTRY_URL	Updated 13 days ago
 REGISTRY_USERNAME	Updated 13 days ago

# Hosting a Static Website

Last updated : 2024-07-04 16:25:38

## Overview

TEM provides static website resource hosting capabilities through **application instance + CFS**. This document takes the popular static website service [Hugo](#) as an example to describe the practical tutorial for static resource hosting. The following will generate a personal static blog through Hugo, deploy a reverse proxy application through TEM, work together with CFS to manage static resources, and finally offer access to the personal static blog over the public network through the access configuration in TEM.

The overall process is as follows:

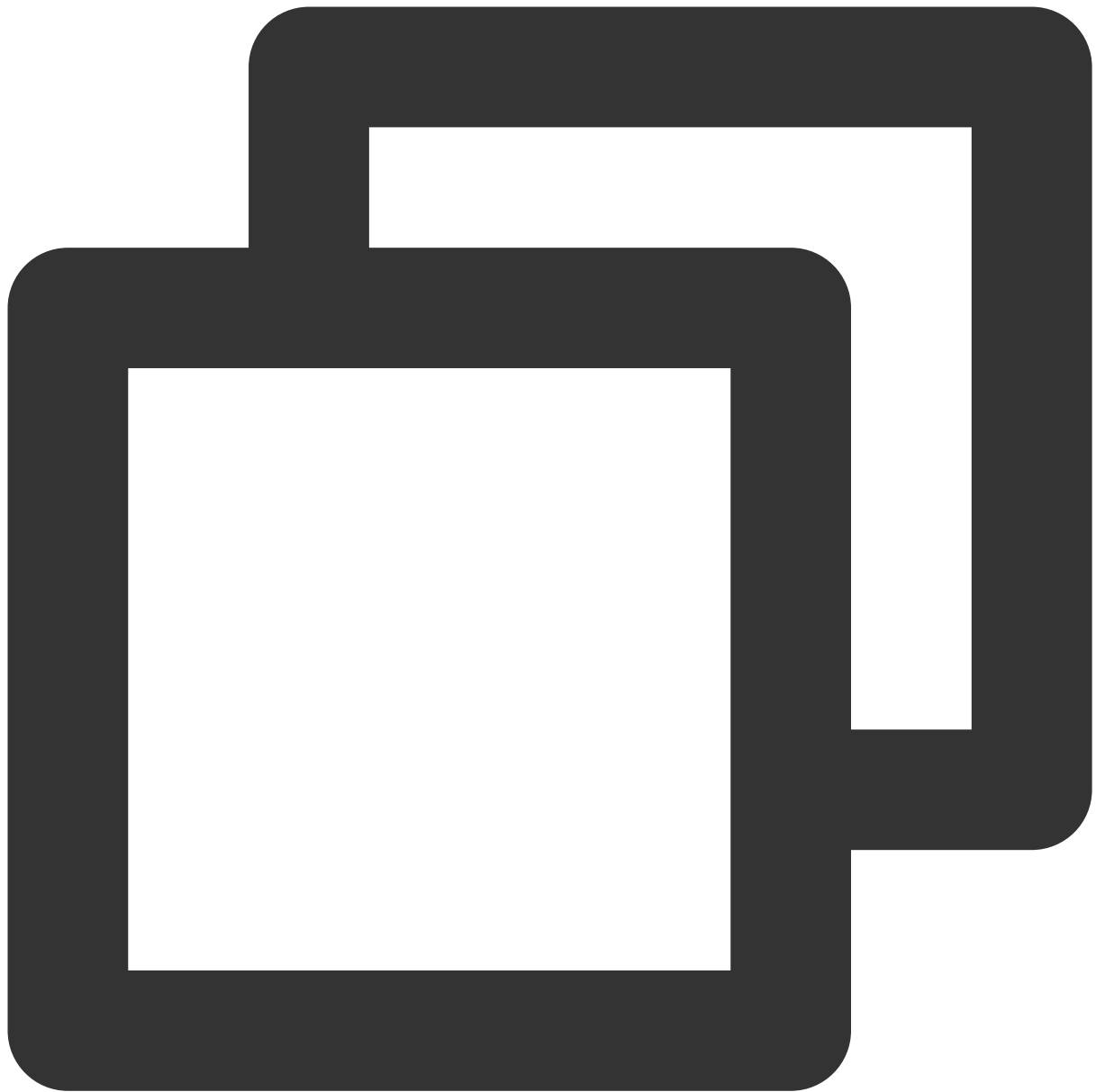
1. [Generate a static blog locally through Hugo](#)
2. [Upload the static blog content to CFS](#)
3. [Deploy the nginx application in TEM and associate CFS](#)
4. [Configure the nginx network access in TEM](#)
5. (Optional) [Configure a domain name](#)
6. (Optional) [Configure a firewall](#)
7. (Optional) [Configure CDN](#)

## Directions

### Step 1. Generate a static blog locally through Hugo

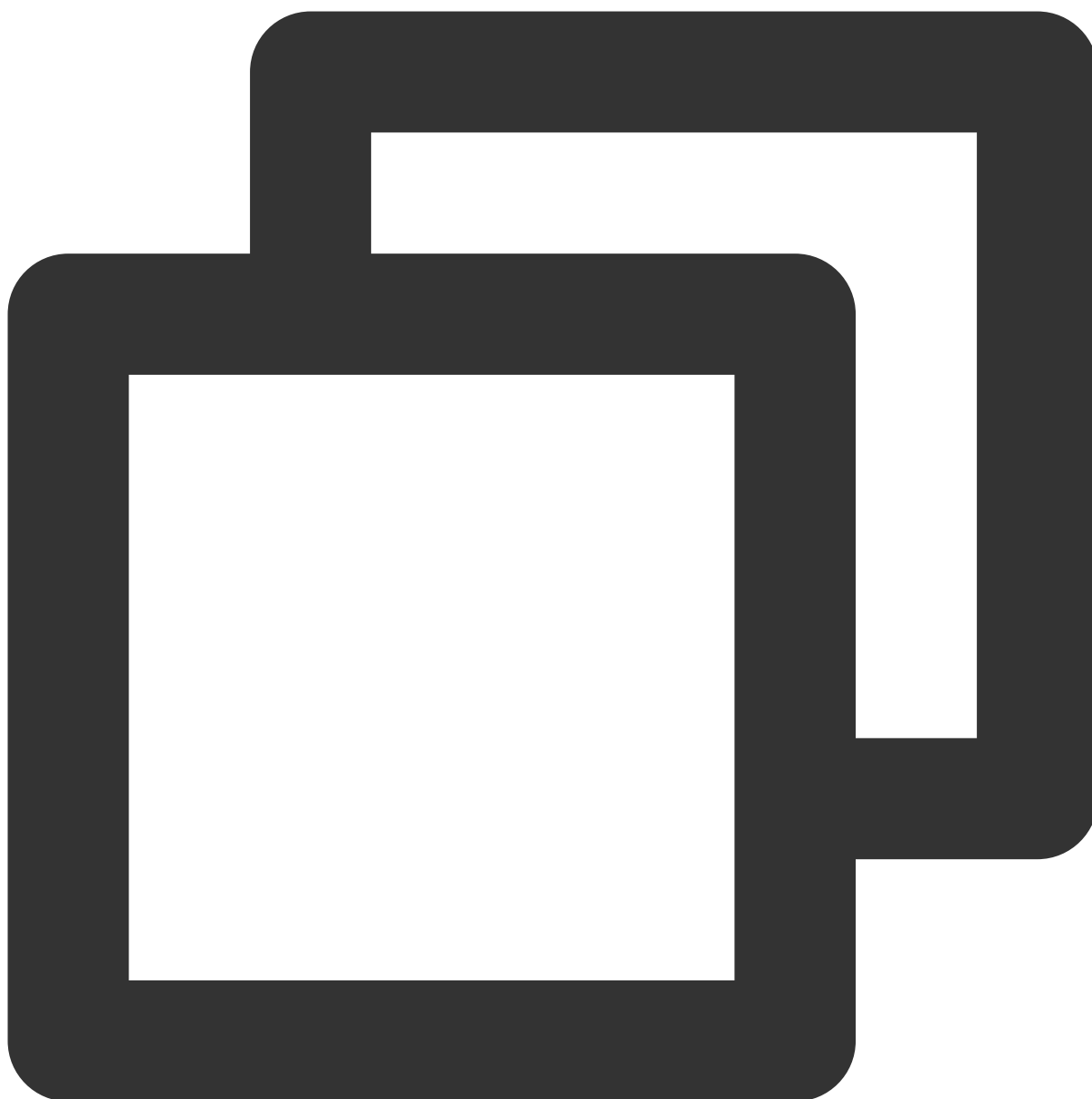
1. Install Hugo (for more information, please see [Install Hugo](#)).

Taking macOS as an example, install it with the following command:



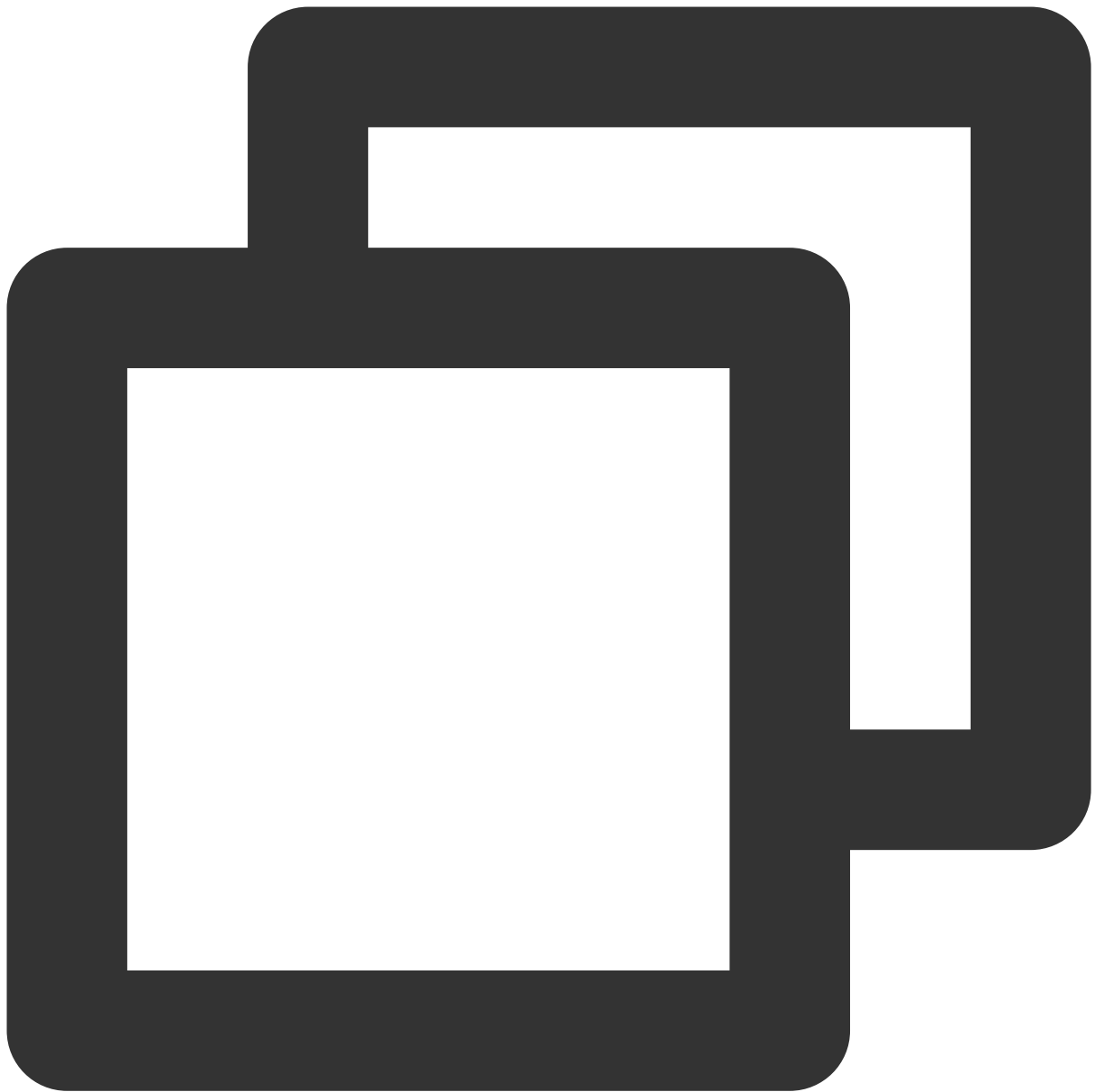
```
brew install hugo
```

2. Run the following command to create a static site.



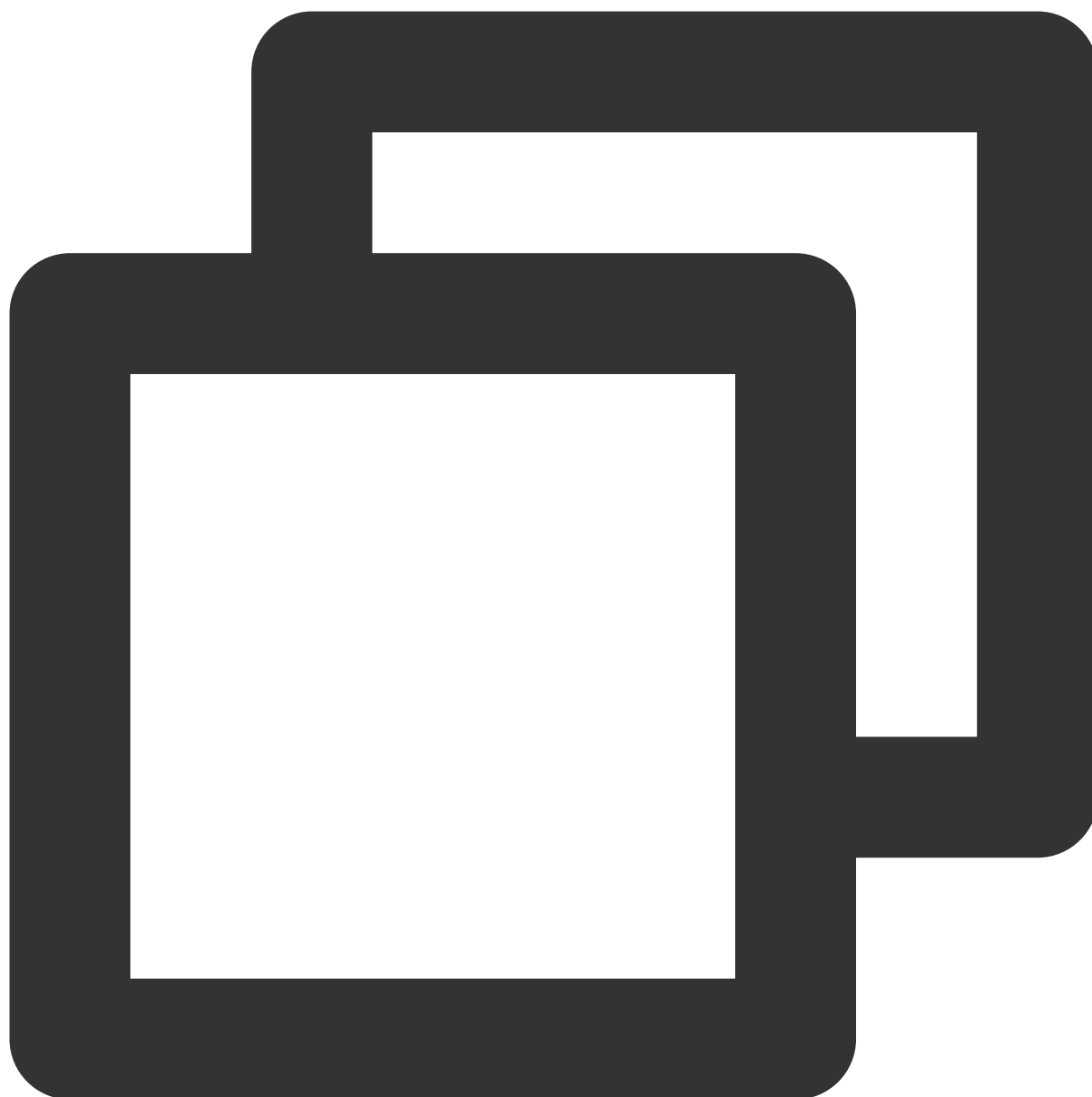
```
hugo new site quickstart
```

3. Run the following command to add a theme.



```
cd quickstart
git init
git submodule add https://github.com/theNewDynamic/gohugo-theme-ananke.git themes/a
echo theme = \"\"ananke\"\" >> config.toml
```

4. Run the following command to add a blog.



```
hugo new posts/my-first-post.md
```

5. Run the following command to build a static page.



```
hugo -D
```

6. The generated static content is stored in the `public/` directory of the `quickstart` project.



```
→ quickstart git:(master) ✕ ll
total 8
drwxr-xr-x  3 zhangyifei  staff   96B Jul 11 15:45 archetypes
-rw-r--r--  1 zhangyifei  staff   99B Jul 11 15:47 config.toml
drwxr-xr-x  3 zhangyifei  staff   96B Jul 11 15:47 content
drwxr-xr-x  2 zhangyifei  staff   64B Jul 11 15:45 data
drwxr-xr-x  2 zhangyifei  staff   64B Jul 11 15:45 layouts
drwxr-xr-x 11 zhangyifei  staff  352B Jul 11 15:50 public ←
drwxr-xr-x  3 zhangyifei  staff   96B Jul 11 15:47 resources
drwxr-xr-x  2 zhangyifei  staff   64B Jul 11 15:45 static
drwxr-xr-x  3 zhangyifei  staff   96B Jul 11 15:46 themes

→ quickstart git:(master) ✕ tree public
public
├── 404.html
├── ananke
│   └── css
│       ├── main.css.map
│       └── main.min.css
├── categories
│   ├── index.html
│   └── index.xml
├── images
│   └── gohugo-default-sample-hero-image.jpg
├── index.html
├── index.xml
├── posts
│   ├── index.html
│   ├── index.xml
│   ├── my-first-post
│   │   └── index.html
│   ├── page
│   │   └── 1
│   │       └── index.html
├── sitemap.xml
├── tags
│   ├── index.html
│   └── index.xml
└── 9 directories, 15 files
```

## Step 2. Upload the static blog content to CFS

1. Purchase a CFS file system as instructed in [Creating File Systems and Mount Targets](#).

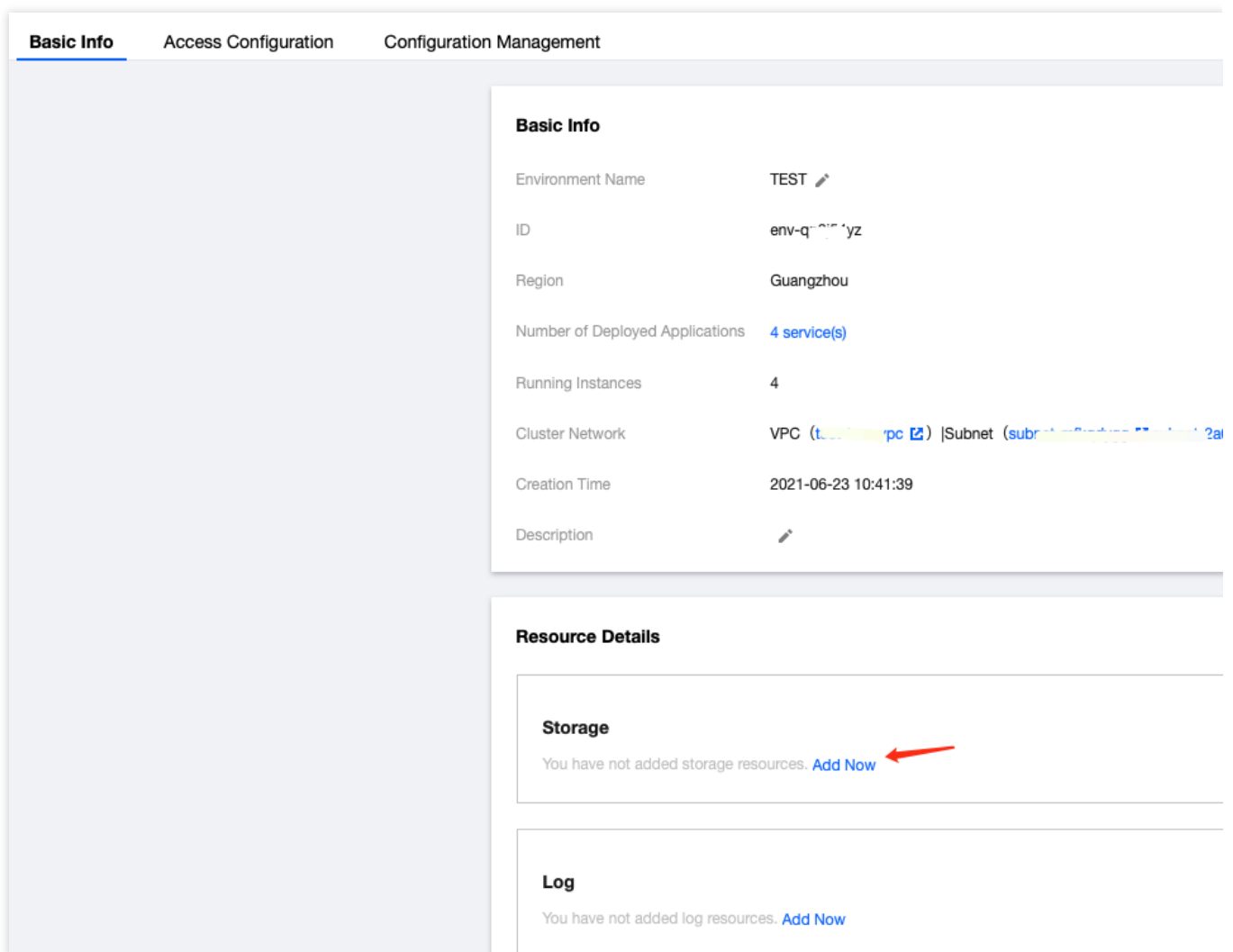
### Note:

The **region** and **VPC** of the purchased CFS file system should be the same as those of the application deployed in TEM.

2. Upload the files in the `public/` directory generated by Hugo to the root directory or subdirectory of the CFS file system as instructed in [Using CFS File Systems on Linux Clients](#) or [Using CFS File Systems on Windows Clients](#).

### Step 3. Deploy the nginx application in TEM and associate CFS

1. Log in to the [TEM console](#) and associate the CFS instance purchased above with the environment where the application is deployed.



2. Create an application named `hugo` on the [Application Management](#) page.

finisz

Page 19 of 63

▼ Access Configuration

▼ Application Lifecycle Management Configures the task to execute before the application launch and termination, suc

▼ Configuration Setting

▼ Environment Variables

▼ Health Check

▲ Persistent Storage Provides storage for the container. Currently, CFS is supported, which needs to be mounted to th

Data Volume

Enter the data volume name

Select the CFS address ▼

Enter the CFS pa

Add

Purchase CFS [🔗](#)

Mount Target

Select a data volume ▼

Target path, such as "/root" ✕

Add

▼ Security Group

## Step 4. Configure the nginx network access in TEM

Scheme 1. Configure a forwarding rule (recommended)

Scheme 2. Configure public network CLB

1. On the [Application Management](#) page, click the ID of the application you just created to enter its basic information page.
2. On the application basic information page, click **Configure Now** in the **Access Configuration** module to enter the environment access configuration page.

©2013-2022 Tencent Cloud. All rights reserved.

Page 20 of 63

Instance List Log Monitoring **Basic Info**

---

**Basic Info**

Application Name	hugo
Running Instances	1
Application description	-
Programming Language	JAVA
Package Name	-
Image	tem-2021-06-28 21:53:11-test-nqdkennr
Image Version	hello-world
Deployment Method	IMAGE
VPC	vpc-2psnvird
Subnet	subnet-1-2021-06-28 21:53:26-2a0g4a
Creation/Update Time	2021-06-28 21:53:11 2021-06-28 21:53:26

---

**Auto-Scaling**

Number of Initialized Instances	1
Specification	1-core 1 GB

---

**Access Configuration**

Access Method	Not set
Access Address	Not set
Forwarding Configuration	<a href="#">Go to Settings</a>

3. On the environment access configuration page, click **Create** to create an access configuration rule.

Rule Name

Enter up to 63 characters containing lowercase letters, digits, and "-" (must start with a lowercase letter and end with a digit or lowercase letter).

Network Type

Load Balancer

CLB instance (supports HTTP/HTTPS) 0.2 USD/day

Protocol & Port ☒ Http:80 ☐ Https:443

Forwarding Configuration

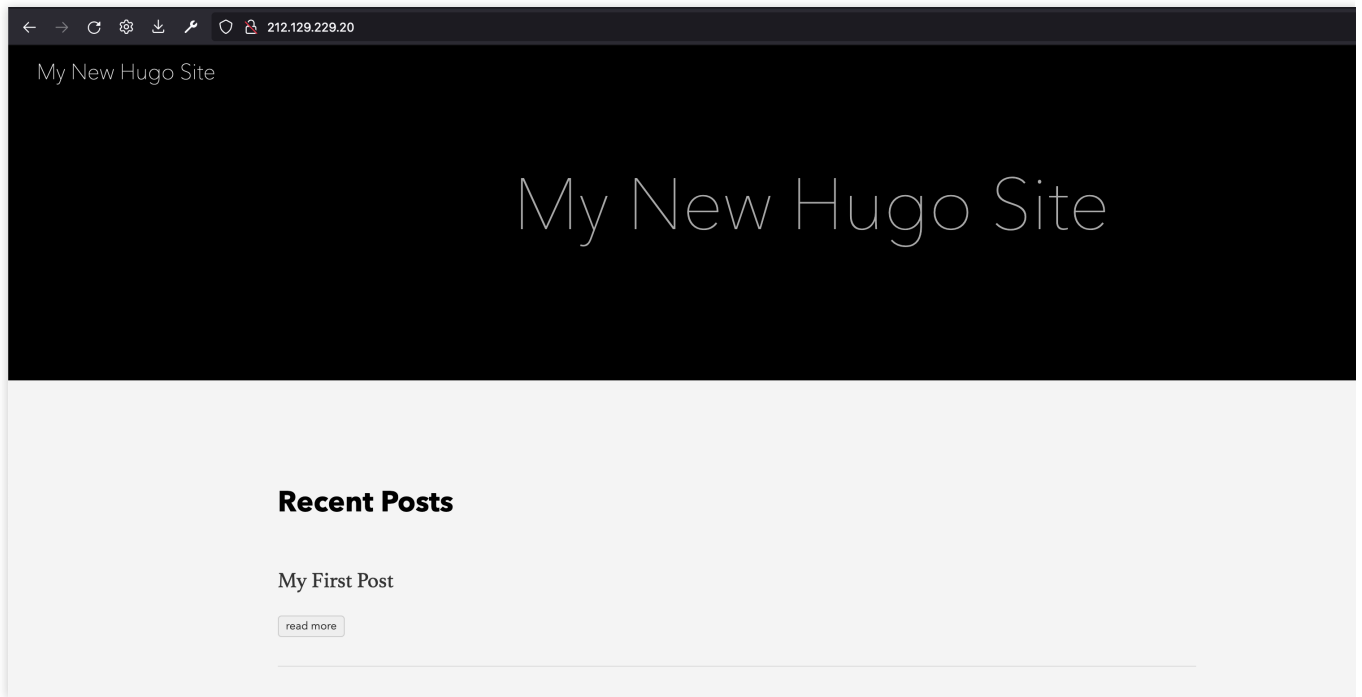
Protocol	Listenin...	Domain name ⓘ	Path	Backend Application	Ap
HTTP ▾	80	IPv4 IP assigned by defa	/	hugo ▾	ε

[Add forwarding rule](#)

4. Click **Complete** to get the following IP address.

Create				
Name	CLB	IP ⓘ	Creation Time	Oper
hugo	lb-7...20	1.1...122	2021-08-04 10:53:40	Edit

Access the Hugo service at the generated address:



1. On the **Application Management** page, click the ID of the application you just created to enter its basic information page.
2. On the application basic information page, click **Edit and Update** in the top-right corner of the **Access Configuration** module to add a public network CLB instance.

**Basic Info**

Application Name	hugo
Running Instances	1
Application description	-
Programming Language	OTHER
Package Name	-
Image	tem-200019-...-q-wave-hugo-wdquzles
Image Version	hello-world
Deployment Method	IMAGE
VPC	vpc-2psnvld
Subnet	subnet-r-...-2a6gqw4a
Creation/Update Time	2021-08-04 10:52:27 2021-08-04 10:52:38

**Auto-Scaling**

Number of Initialized Instances	1
Specification	1-core 2 GB

**Access Configuration**

Access Method	Not set
Access Address	Not set

3. Select the public network access method and enter the port mapping relationship.

### Edit Access Configuration

Access Method ☐ Intra-Environment Access ☐ VPC Access (Layer-4 Forwarding) ☒ Public Network Access (Layer-

4) Automatically creates a public network CLB (0.2 USD/day) to provide an Internet access entry. The TCP/UDP is supported. You can select this option for frontend web applications.

If you want to configure HTTP/HTTPS forwarding rules over the public network, we suggest you select "Intra-Environment Access" and configure forwarding rules on the [routing policy page](#).

Port Mapping

Protocol ⓘ	Container Port ⓘ	Application Listening Port ⓘ
TCP ▼	80 ▲▼	80 ▲▼

[Add port mapping](#)

Submit

Disable

4. Click **Submit** to get the following public IP.



**Basic Info**

**Basic Info**

Application Name	hugo
Running Instances	1
Application description	-
Programming Language	OTHER
Package Name	-
Image	tem-20001-100000-qpsvld-hugo-0qzles
Image Version	hello-world
Deployment Method	IMAGE
VPC	vpc-2psnvld
Subnet	subnet-m-100000-qpsvld-2gqw4a
Creation/Update Time	2021-08-04 10:52:27 2021-08-04 10:56:16

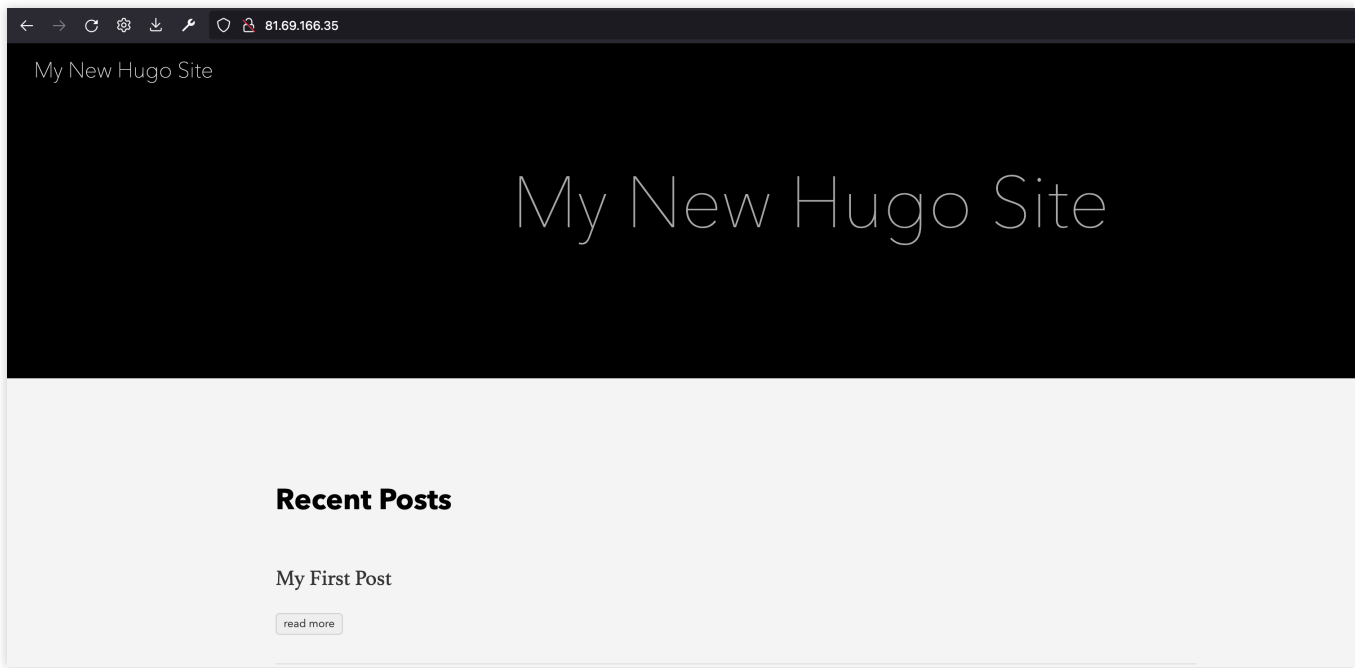
**Auto-Scaling**

Number of Initialized Instances	1
Specification	1-core 2 GB

**Access Configuration**

Access Method	Public Network Access (Layer-4 Forwarding)
Access Address	Public IP 119.28.11.139:80 ←

Access the Hugo service at the generated address:



## Step 5. (Optional) Configure a domain name

1. Register a domain name.
2. Associate the domain name with the CLB instance generated above, and you can access the static website at the domain name. For more information, please see [Getting Started with CLB](#).

## Step 6. (Optional) Configure a firewall

If the static website is accessed through the **forwarding configuration** settings, it can be connected to Tencent Cloud [WAF](#) for security protection. For more information, please see [Configuring WAF protection for CLB listening domain names](#).

## Step 7. (Optional) Configure CDN

In order to deliver a better user experience, the hosted static website can be connected to Tencent Cloud CDN for access acceleration. For more information, please see [Getting Started](#).

# Public Network Access of TEM Applications

Last updated : 2024-01-09 12:42:59

## Overview

Applications running on TEM usually need public network access, and also require allowlist access in scenarios such as mini programs. In these cases, the application should have a fixed public IP.

This document describes how to enable public network access of the applications deployed on TEM.

### Solution

The applications are deployed in a TEM environment, which associates with your VPC. In other words, they are essentially deployed in your VPC. You can configure a NAT Gateway instance and associate it with an EIP for your VPC, allowing the applications in your VPC to access the public network.

### Steps

1. [Deploy the applications in TEM.](#)
2. [Create a NAT Gateway.](#)
3. [Configure the NAT Gateway in the VPC console.](#)
4. [Verify whether the TEM applications can access the public network.](#)
5. [\(Optional\) Query public network access IP addresses.](#)

## Directions

### Step 1: deploy the applications in TEM

Configure the applications in the TEM console as instructed in [Creating Environment](#) and [Creating and Deploying Application](#).

### Step 2: create a NAT Gateway

Log in to the [NAT Gateway](#) console, select the region where the TEM applications are deployed, and click **+New** to create a NAT Gateway instance.

#### Create NAT Gateway



Gateway Name

tem-demo

52 more characters allowed

Network

vpc-2psnvld(██████████,███'███ 0.0.0/16) ▼

Region

South China (Guangzhou)

Gateway Type

Small-scale (Max concurrent connections: 10 ▼)

Outbound Bandwidth Cap

10Mbps ▼ ?

The public network traffic is determined by the lower of the NAT Gateway's or EIP's bandwidth cap.

Elastic IP

Create EIP ▼

↻ [Create Now](#) [🔗](#)

[+Bind IP](#) Only the EIP billed by traffic or by bandwidth package can be bound. Up to 10 IPs can be bound ?

Tag

Tag key	Tag value	Operation
Please select	Please select	×

Add

Gateway Fee

USD /hr

Network Fee

USD/GB

?

After creation, you need to configure routing rules and direct the subnet traffic to NAT Gateway.  
To get notified about abnormal NAT gateway behaviors instantly, please [configure alarms](#).

Create

Close

**Network:** select the VPC with which the environment of the TEM applications associates.

**Elastic IP:** if there is no available elastic IP (EIP), click **Create Now** to purchase an EIP, and then return to the **Create NAT Gateway** page to select it.

### Step 3: configure the NAT Gateway in the VPC console

1. Log in to the TEM console and access the [Environment](#) page. Select the environment in which the TEM applications are deployed to enter its details page.
2. Click the VPC next to **Cluster Network** to enter the VPC details page.

Basic Info	Resource Management	Access Management	Configuration Management
<b>Basic Info</b>			
Environment Name	TEST		
ID	env-qn2j54yz		
Region	Guangzhou		
Number of Deployed Applications	5 service(s)		
Running Instances	5		
Cluster Network	VPC (te...dpc )   Subnet (subnet-...2a6gqw4a )		
Creation Time	2021-06-23 10:41:39		
Description			

3. Select the **Route Table** module.
4. Click **Create** on the **Route Table** page to configure a route table.

### Create Route Table

Name
60 more characters allowed

Network
vpc-2psnvlrd(test-tem-vpc | 1

Advanced Options

#### Routing Rules

*i* Routing policies control the traffic flow in the subnet. For details, please see [Configuring Routing Policies](#).

Destination	Next hop type	Next hop	Notes
Local	LOCAL	Local	Delivered by default, indica
such as 10.0.0.0/16	NAT Gateway	No available NAT Gateway	

Create a NAT Gateway

**Destination:** select the public IP address to be accessed. You can configure a CIDR block for this parameter. For example, if you enter `0.0.0.0/0`, all traffic will be forwarded to the NAT Gateway.

**Next hop type:** select **NAT Gateway**.

**Next hop:** select the NAT Gateway created in the **step 2**.

For detailed directions, see [Creating Custom Route Tables](#).

5. On the **Route Table** page, locate the route table just created, and click **More > Associated Subnets** under the **Operation** column. In the pop-up window, select the subnet associated with the environment in which the TEM applications are deployed.

+ New

ID/Name	Type	Network	Associated sub...	Creation Tim
rtb-p0s5sbja	Default route table	vpc-2psnvlrd test-tem-vpc	2	2021-06-23 11

## Step 4: verify whether the TEM applications can access the public network

1. Log in to the TEM console and access the [Application Management](#) page. Click the **ID/Name** of the TEM applications to enter the instance list page.
2. Click **Webshell** under the **Operation** column of the target application.

Basic Info Log Monitoring Instance List

**Default Deployment Information**

Terminate Running instances: 1 / Desired instances: 1

ID	AZ	IP	Status	Creation Time
hugo-99c94bdff-n9zd5	Guangzhou Zone 6	10.10.10.10	Running	2021-08-04 10:10:10

3. Verify whether the application can access the public network.

```
# bash
root@hugo-99c94bdff-n9zd5:/# ping qq.com
PING qq.com (183.3.226.35) 56(84) bytes of data.
64 bytes from 183.3.226.35 (183.3.226.35): icmp seq=1 ttl=48 time=31.2 ms
64 bytes from 183.3.226.35 (183.3.226.35): icmp seq=2 ttl=48 time=31.2 ms
64 bytes from 183.3.226.35 (183.3.226.35): icmp seq=3 ttl=48 time=31.3 ms
64 bytes from 183.3.226.35 (183.3.226.35): icmp seq=4 ttl=48 time=31.3 ms
```



可以访问通公网

## Step 5: (optional) query public network access IP addresses

1. Log in to the TEM console and access the [Environment](#) page. Select the environment in which the TEM applications are deployed to enter its details page.
2. Click the VPC next to **Cluster Network** to enter the VPC details page.

Basic InfoResource ManagementAccess ManagementConfiguration Management

Basic Info

Environment Name	TEST 
ID	env-qn2j54yz
Region	Guangzhou
Number of Deployed Applications	<a href="#">5 service(s)</a>
Running Instances	5
Cluster Network	VPC ( <a href="#">vpc-...</a> )   Subnet ( <a href="#">subnet-...</a> )
Creation Time	2021-06-23 10:41:39
Description	

3. Select the **NAT Gateway** model to go to the **NAT Gateway** page.

4. Click the **ID/Name** of the target NAT Gateway to access its details page. Select the **Bind Elastic IP** tab to view the IP addresses that can access the public network.

## Additional Fees

The [NAT Gateway](#) and EIP will be charged separately. For pricing details, see:

[NAT Gateway Billing Overview](#)

[Elastic IP Billing](#)



# TEM Application Access to Public Network (Through API Gateway)

Last updated : 2024-01-26 16:40:24

## Overview

Applications running on TEM usually need to access the public network for business and other reasons. In many cases, these requests are all HTTP/HTTPS requests. You can use API Gateway to easily access HTTP/HTTPS requests from the public network through simple configuration.

### Note:

If your access to the public network does not only include HTTP/HTTPS, refer to [Public Network Access of TEM Applications](#) to configure a NAT gateway for implementation.

## Prerequisites

Create an [environment](#) and create and deploy an [application](#).

## Directions

### Step 1. Associate public network HTTP/HTTPS requests in API Gateway

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), and instance type (shared).

Up to 50 chars, supporting a-z, A-Z, 0-9, and underscores.

Type **HTTP&HTTPS** HTTP HTTPS

Mode Pay as you go

Network ☒ Private VPC ☐ Public Network

When a private VPC is selected, the generated private VPC domain name can be accessed in the VPC network in the same region of the service

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 **No authentication** as the authentication type, and click **Next**.

1 Frontend Configuration

2 Backend Configuration

3 Response Result

Service test

API Name test-api  
Up to 60 chars

Frontend Type **HTTP&HTTPS** WS&WSS

Path /

1. Supports starting with "/" and "=/". Starting with "/" means fuzzy match, while starting with "=/" means exact match.
2. Supports uppercase and lowercase letters, numbers, and [-.\*/\*~%]
3. The Path parameter must be wrapped with curly braces {} as a separate part of the path (such as /{param}/)
4. When the path starts with "=/", adding request parameter of type Path is not supported.

Request Method GET POST PUT DELETE HEAD **ANY**

Authentication Type **Authentication-Free** App Authentication OAuth 2.0 EIAM Verification Key pair

An authentication-free mode under which APIs are accessible to all users, featuring a low security level. For more information, see [user guide for authentication](#)

CORS is supported ☐

1. When it's enabled, "access-control-allow-origin : \*" will be added to the response header by default.
2. To customize CORS configuration, please create a CORS plugin and bind it with the API. See [CORS Plugin Usage Guide](#)

Remarks Please enter remarks

Parameter Name	Parameter Location ⓘ	Type	Default Value ⓘ	Required	F
----------------	----------------------	------	-----------------	----------	---

5. In the **Backend Configuration** step, select **Public URL/IP** as the backend type, configure the public domain name and path you need to access (Tencent Cloud official website is used as an example here), and click **Next**.

Frontend Configuration

Backend Configuration

Response Result

Backend Type

Public URL/IP

Provide backend services externally through the public network

VPC resources

Connects to hosts and containers in VPC via upstreams and private CLB

Serverless Cloud Function (SCF)

Serverless computing service provided by Tencent Cloud

Mock

Simulate response data for testing

Backend Domain Name

http://

cloud.tencent.com

It starts with "http" or "https" and contains domain content, and "/" is not required at the end

For a shared instance, you can only enter the public network domain name. For a dedicated instance, both the public network domain name and private IP :

Backend Path

/

1. It starts with "/", and supports uppercase and lowercase letters, digits, and \$-\_.+!\*()/%.

2. "=" and "^~" in the frontend parameters are used for exact match to the frontend path, which are not available to the backend path.

3. The Path parameter must be wrapped with curly braces {} as a separate part of the path (such as {/param}/)

Request Method

GET

POST

PUT

DELETE

HEAD

ANY

Backend timeout

15

seconds

Time range: 1-1800s

Constant parameter

Parameter Name	Parameter Location	Parameter Value	Remarks
Newly added constant parameter (0/30)			

Previous

Next

6. Set the return type of the application (which is HTML here), select **JSON** as the RESTful service, and click **Complete** to publish the service.

## Step 2. Verify public network request connectivity

1. Go to the API Gateway service's basic configuration page and copy its VPC access address.

Basic Info

Service Name

Service ID

Region

Published Environments

Number of Created APIs

QPS Limit ⓘ

Remarks

Guangzhou

Publish

1/200

500 (If you need to increase the QPS limit, please [buy a dedicated instance](#) ⓘ)

- ⓘ

Publish

Network

Frontend Type

Private VPC Access Address

IP Version

HTTP&HTTPS ⓘ

IPv4 ⓘ

Billing Details

Instance Type

Billable Item

Creation Time

Shared Type

Number of calls + public network outbou

2022-01-20 16:38:48

2. Open the deployed TEM application page, enter the webshell of the application instance, and visit the API Gateway VPC access address to verify the network connectivity.

```
Select to copy the texts you want, and press Shift + Insert to paste.
sh-4.2# curl http://service- -in.gz.apigw.tencentcs.com:
<!DOCTYPE html>
<html lang="zh">
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, maximum-scale=1">
<script>var timeStamps = [new Date()];</script>
<meta charset="utf-8">
<title>
<meta name="description" content="
<meta property="og:title" content="
<meta property="og:description" content="
<meta property="og:image" content="https://cloud.tencent.com/open_proj/proj_qc201903182234/ver_geticon?size=1000x1000.png" />
<meta property="og:url" content="https://cloud.tencent.com" />
<meta name="format-detection" content="telephone=no">
<meta name="baidu-site-verification" content="Yqh6LFrxHs" />
<meta name="google-site-verification" content="FLzxWYFG80HK_XLvVz6GpBdsY-f7fdNfnucHDWMCr7k" />
<meta name="shenma-site-verification" content="d3Jdb0ae4613f827196c6e785aaeeed_1593673238" />
<meta name="360-site-verification" content="943b53ddelf410dc44724f15d9el1954" />
<meta name="sogou_site_verification" content="j0R8AKsqAJ" />
<meta name="bytedance-verification-code" content="" />
<link rel="dns-prefetch" href="//cloudcache.tencent-cloud.com">
<link rel="dns-prefetch" href="//qz.gs.qq.com">
<link rel="icon" href="/favicon.ico?201903182234" type="image/x-icon"/>
<link rel="alternate" href="https://intl.cloud.tencent.com/sh/" hreflang="zh-Hant"/>
<link rel="alternate" href="https://cloud.tencent.com/" hreflang="zh-Hans"/>
<link rel="alternate" href="https://intl.cloud.tencent.com/jp/" hreflang="ja"/>
<link rel="alternate" href="https://intl.cloud.tencent.com/ko/" hreflang="ko"/>
<link rel="alternate" href="https://intl.cloud.tencent.com/" hreflang="x-default"/>
<link rel="canonical" href="https://cloud.tencent.com/">
<script>document.domain = 'cloud.tencent.com';</script>
<script>timeStamps[1] = new Date();</script>
</script>
```

Last updated : 2024-01-09 12:42:59

## Instance Error Status

### Status description

## Solution

View the instance logs and troubleshoot the problem based on the **log content**.



### Status description

## Solution

Default log
CLS logs

❗ Default logging is free of charge. It only supports log display and search for recent standard output. For more advanced features, please use CLS logging.

Instance [dropdown]
↻

Search by the keyword 🔍

```

1.
2.
3.      _ _ _ _ _
4.    / \   \   \   \   \   \   \
5.   /   \_/_/_/_/_/_/_/_/_/_/_\
6.  /     \_/_/_/_/_/_/_/_/_/_/_\
7.  |_____|_____/_/_/_/_/_/_/_\
8. :: Spring Boot ::                (v2.4.5)
9.
10. 2022-08-31 09:56:04.767 INFO 1 --- [main] c.e.helloworld.HellWorldApplication : Starting HellWorldApplication v0.0.1-SNAPSHOT using Java 1.8.0_212 on test-l2bk5 with PID
11. 2022-08-31 09:56:04.775 INFO 1 --- [main] c.e.helloworld.HellWorldApplication : No active profile set, falling back to default profiles: default
12. 2022-08-31 09:56:07.637 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8201 (http)
13. 2022-08-31 09:56:07.679 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
14. 2022-08-31 09:56:07.680 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.45]
15. 2022-08-31 09:56:07.847 INFO 1 --- [main] o.a.c.o.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
16. 2022-08-31 09:56:07.847 INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2910 ms
17. 2022-08-31 09:56:09.050 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
18. 2022-08-31 09:56:09.628 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8201 (http) with context path ''
19. 2022-08-31 09:56:09.660 INFO 1 --- [main] c.e.helloworld.HellWorldApplication : Started HellWorldApplication in 6.146 seconds (JVM running for 7.261)
```

## Running Unhealthy: Readiness probe failed

### Status description

## Solution

Go to **Application Deployment > Health check** and check whether the **Readiness Probe** configuration item of the application is correct.

Readiness check	<input checked="" type="checkbox"/>
Checking method	HTTP request check ▼
Protocol	HTTP ▼
Request path ⓘ	<input health\""="" type="text" value="Enter the URL path of the request, such as \"/>
Port	<input type="text" value="Enter the request port"/> Port range: 1 - 65535. Port names are supported.
Start-up latency	<input type="text" value="10"/> seconds How long to wait after application startup before Readiness Probe. It defaults to 10 second(s).
Response timeout	<input type="text" value="3"/> seconds The timeout period for a Readiness Probe. It defaults to 3 seconds
Check interval	<input type="text" value="30"/> seconds The timeout period for a Readiness Probe. It defaults to 30 seconds

## Running Unhealthy: Liveness probe failed

### Status description

The aliveness health check configured for the application failed.

### Solution

Go to **Application Deployment > Health check** and check whether the **Aliveness Probe** configuration of the application is correct.

Liveness Check	<input checked="" type="checkbox"/>
Checking method	HTTP request check ▼
Protocol	HTTP ▼
Request path ⓘ	<input health\""="" type="text" value="Enter the URL path of the request, such as \"/>
Port	<input type="text" value="Enter the request port"/> <small>Port range: 1 - 65535. Port names are supported.</small>
Start-up latency	<input type="text" value="5"/> seconds <small>How long to wait after application startup before Aliveness Probe. It defaults to 5 second(s).</small>
Response timeout	<input type="text" value="3"/> seconds <small>The timeout period for a Aliveness Probe. It defaults to 3 seconds</small>
Check interval	<input type="text" value="30"/> seconds <small>The timeout period for a Aliveness Probe. It defaults to 30 seconds</small>

**Running Unhealthy: Readiness check failed according to I4 listener: xxx of CLB xxx. Service: xxx**

#### Status description

The access configuration of the application cannot take effect, and the application cannot be accessed.

#### Solution

Go to **Application details > Basic information > Access configuration** and check whether the port and protocol are correct.



### Edit access configuration

Access method

k8s service

Private CLB

Public CLB

Registry

1. Provide an access point for another applications in the same environment. It supports TCP and UDP.

2. To configure an HTTP/HTTPS forwarding rule for internet access, go to [Manage environment - Access](#) .

Service name

test1

1 - 63 characters. It supports only lowercase letters, digits and hyphens (-). It must end with a lowercase and cannot end with "-tem".

Port mapping

Protocol ⓘ	External access port ⓘ	Application Listening Port ⓘ
<div>UDP ▼</div>	<div>1</div>	<div>1</div>

Add port mapping

Submit

Close

## PostStartHookError

### Status description

PostStart configured for the application failed.

### Solution

Go to **Application Deployment > Application start/stop management** and check whether **PostStart** configured for the application can run normally.

▲ Application start/stop

shell

sh

bash method

PostStart ⓘ

One command per line. Empty lines are not allowed. For example  
echo Hello  
nginx -s quit

shell

sh

bash method

PreStop ⓘ

One command per line. Empty lines are not allowed. For example  
echo Hello  
nginx -s quit

## ContainerCreating

### Status description

The instance container failed to be created.

### Solution

Go to **Application Deployment > Persistent storage** and check whether the application is mounted with a nonexistent data volume.

▲ **Persistent storage**

Data volume

Volume name	CFS address	CFS Path
<input type="text" value="Enter the data volume name"/>	<div>Select the CFS address ▼</div>	<input type="text" value="Enter the CFS path"/>

Add [Purchase CFS](#)

Mount target

Data volume	Target path	Operat...
<div>Select a data volume ▼</div>	<input root\""="" type="text" value="Target path, such as \"/>	<div>×</div>

Add

## CreateContainerConfigError

### Status description

The instance container failed to be configured.

### Solution

Go to **Application Deployment > Environment variable** and check whether the application uses nonexistent configuration.

▲ **Environment variable**

Environment variable

Type	Variable Name	Variable value ⓘ
<div>Custom ▼</div>	<input type="text" value="name"/>	<input type="text" value="Please enter"/>

Add

## ImagePullBackOff

### Status description

The instance image failed to be pulled.

## Solution

Go to the [TCR console](#) and check whether the image used by the application exists or has been deleted by mistake.

# Quick Access to TEM Application Through API Gateway

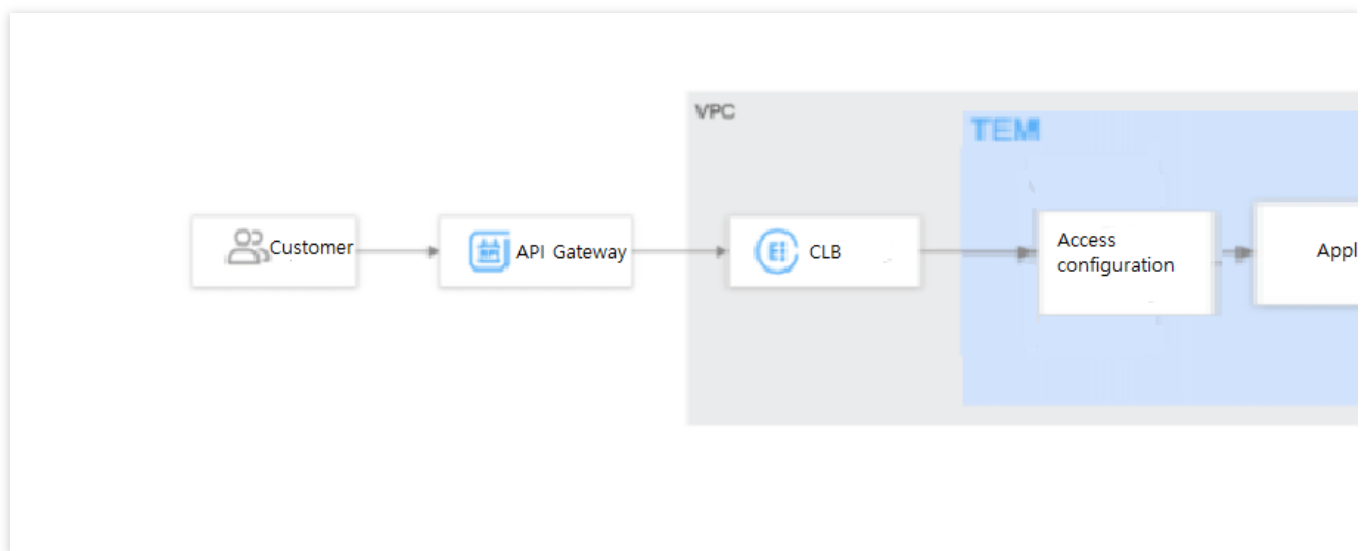
Last updated : 2024-01-09 12:42:59

## API Gateway Overview

API Gateway is an API hosting service launched by Tencent Cloud, which supports the management of APIs throughout their lifecycle from creation, maintenance, launch, and operation to deactivation. For more information, see [API Gateway product documentation](#).

## 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](#).

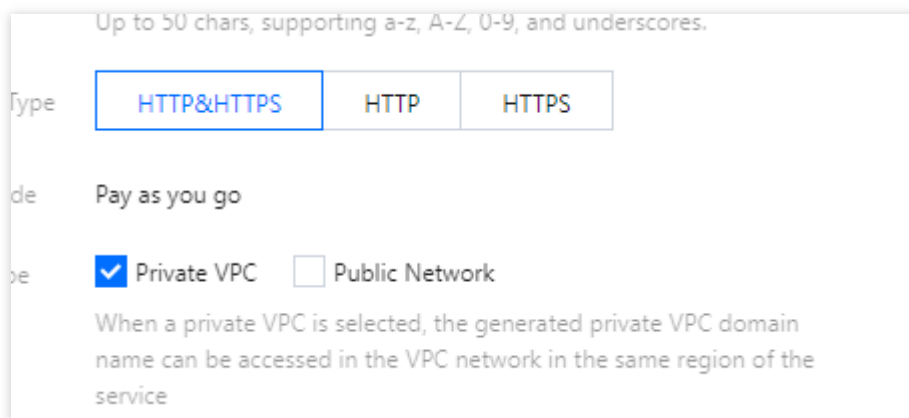
## Directions

### 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).



Up to 50 chars, supporting a-z, A-Z, 0-9, and underscores.

Type: **HTTP&HTTPS** HTTP HTTPS

Mode: Pay as you go

Access Mode: ☒ Private VPC ☐ Public Network

When a private VPC is selected, the generated private VPC domain name can be accessed in the VPC network in the same region of the service

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**.

1 Frontend Configuration

2 Backend Configuration

3 Response Result

Service: test

API Name: test-api  
Up to 60 chars

Frontend Type: HTTP&HTTPS WS&WSS

Path: /  
1. Supports starting with "/" and "=/". Starting with "/" means fuzzy match, while starting with "=/\" means exact match.  
2. Supports uppercase and lowercase letters, numbers, and [-\_\*/~%]  
3. The Path parameter must be wrapped with curly braces {} as a separate part of the path (such as /(param)/)  
4. When the path starts with "=/\", adding request parameter of type Path is not supported.

Request Method: GET POST PUT DELETE HEAD ANY

Authentication Type: Authentication-Free App Authentication OAuth 2.0 IAM Verification Key pair  
An authentication-free mode under which APIs are accessible to all users, featuring a low security level. For more information, see [user guide for authentication](#)

CORS is supported: ☐  
1. When it's enabled, "access-control-allow-origin : \*" will be added to the response header by default.  
2. To customize CORS configuration, please create a CORS plugin and bind it with the API. See [CORS Plugin Usage Guide](#)

Remarks: Please enter remarks

Parameter Configurations

Parameter Name	Parameter Location ⓘ	Type	Default Value ⓘ	Required
----------------	----------------------	------	-----------------	----------

5. In the **Backend Configuration** step, select **VPC resource** 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.

Frontend Configuration > **Backend Configuration** > Response Result

Backend Type

Public URL/IP Provide backend services externally through the public network	<b>VPC resources</b> Connects to hosts and containers in VPC via upstreams and private CLB	Serverless Cloud Function (SCF) Serverless computing service provided by Tencent Cloud	Mock Simulate response data for testing
---	---	---	--

Backend Domain Name①

Backend Path①

Request Method

Backend timeout①  seconds  
Time range: 1-1800s

Constant parameter

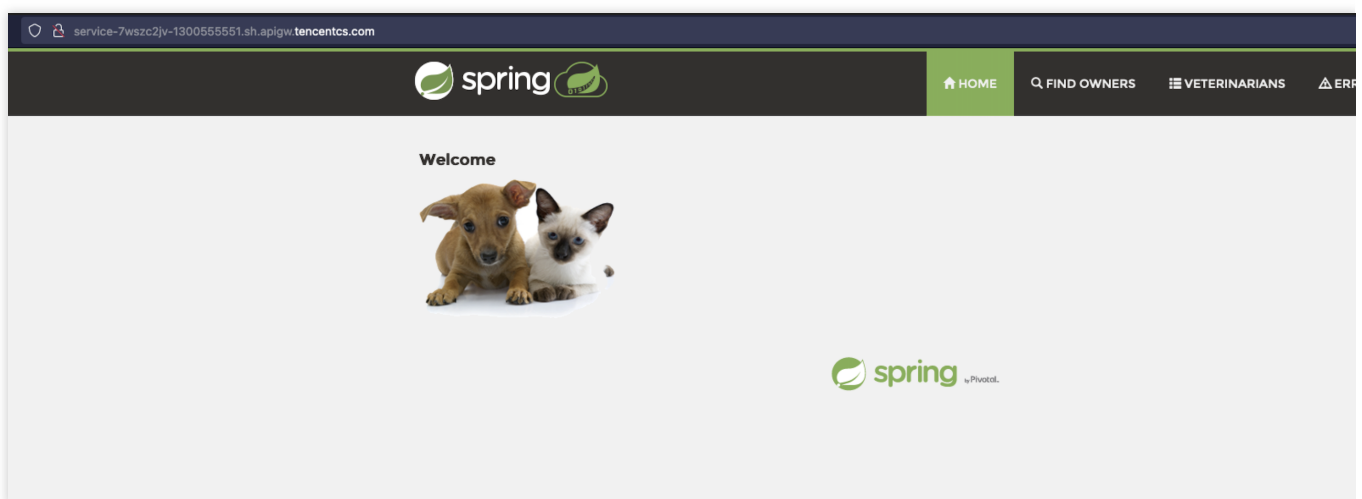
Parameter Name	Parameter Location ①	Parameter Value	Remarks
Newly added constant parameter (0/30)			

[Previous](#) [Next](#)

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 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 [Overview](#) to bind the plugin to the API Gateway API with a TEM backend and then enjoy advanced features provided by API Gateway.

# Java Application Fine-Tuning

Last updated : 2024-07-04 16:25:38

## Optimizing Container Image

By optimizing the container image in the following ways, you can reduce the loading and startup time:

Minimize the container image size.

Avoid using nested JAR packages.

Use TEM JAR/WAR for deployment.

Deployment based on TEM JAR/WAR is easy to use and efficient. It provides practical tutorials for JAR package image builds by default. TEM offers a build process that can fully utilize the build cache by default and uses the new-gen build tool BuildKit to increase the build speed by over 50%. Moreover, build logs can be queried to make the entire build process traceable.

## Setting Application Acceleration

If you use TEM JAR/WAR for deployment and select the KONA JDK 11/Open JDK 11 runtime environment, TEM will enable the application acceleration feature and support zero-code modification acceleration for Spring Boot applications by default. TEM enhances the AppCDS feature in OpenJDK, so you don't need to modify the nested JAR package structure in the original Spring Boot application, and TEM directly provides practical tutorials of Java application acceleration to shorten the startup time by 10%–40% during instance scale-out.

## JVM Parameter Optimization

### Using JDK that can perceive container memory resources

On a VM or physical machine, when allocating CPU and memory resources, JVM will search for available resources from common locations such as `/proc/cpuinfo` and `/proc/meminfo` on Linux. However, at the container runtime, the CPU and memory restrictions are stored in `/proc/cgroups/...`. JDK on an early version will still search for resources in `/proc` but not `/proc/cgroups`, which may cause the CPU and memory usage to exceed the allocated upper limit and further lead to more severe problems:

There are too many threads, as the size of the thread pool is configured by

```
Runtime.availableProcessors()
```

The memory used by JVM exceeds the upper limit of the container memory and causes the `OOMKilled` error in the container.

JDK 8u131 first implements the `UseCGroupMemoryLimitForHeap` parameter, but this parameter has a defect. After the `UnlockExperimentalVMOptions` and `UseCGroupMemoryLimitForHeap` parameters are added to your application, JVM can perceive the container memory and control the actual heap size of the application, but JVM still cannot fully utilize the memory allocated to the container.

Therefore, JVM provides the `-XX:MaxRAMFraction` flag to help better calculate the heap size. The default value of `MaxRAMFraction` is 4 (that is, 4 is the divisor), but it is a fraction, not a percentage; therefore, it is difficult to set a value that can use available memory effectively.

JDK 10 provides better support for the container environment. If you run a Java application in a Linux container, JVM will use the `UseContainerSupport` option to automatically check the memory limits and use `InitialRAMPercentage`, `MaxRAMPercentage`, and `MinRAMPercentage` to control the memory. Here, percentages rather than fractions are used to make the control more accurate.

By default, the `UseContainerSupport` parameter is activated, `MaxRAMPercentage` is 25%, and `MinRAMPercentage` is 50%.

Note that `MinRAMPercentage` here is not used to set the minimum value of the heap size but to restrict the heap size by JVM only when the total available memory of the physical server (or container) is less than 250 MB.

Similarly, `MaxRAMPercentage` here is used to restrict the heap size by JVM only when the total available memory of the physical server (or container) exceeds 250 MB.

These parameters have been backported to JDK 8u191. By default, `UseContainerSupport` is activated. You can set `-XX:InitialRAMPercentage=50.0 -XX:MaxRAMPercentage=80.0` to enable JVM to perceive and fully utilize the available memory of the container. Note that if `-Xms -Xmx` is specified, `InitialRAMPercentage` and `MaxRAMPercentage` will become invalid.

## Disabling the optimization compiler

By default, JVM has multi-stage JIT compilation. Though such stages can gradually improve the application efficiency, they also increase the memory overheads and the application startup time.

For short-lived cloud native applications, you can consider using the following parameters to disable the optimization stage, so as to reduce the startup time at the cost of the long-term running efficiency.

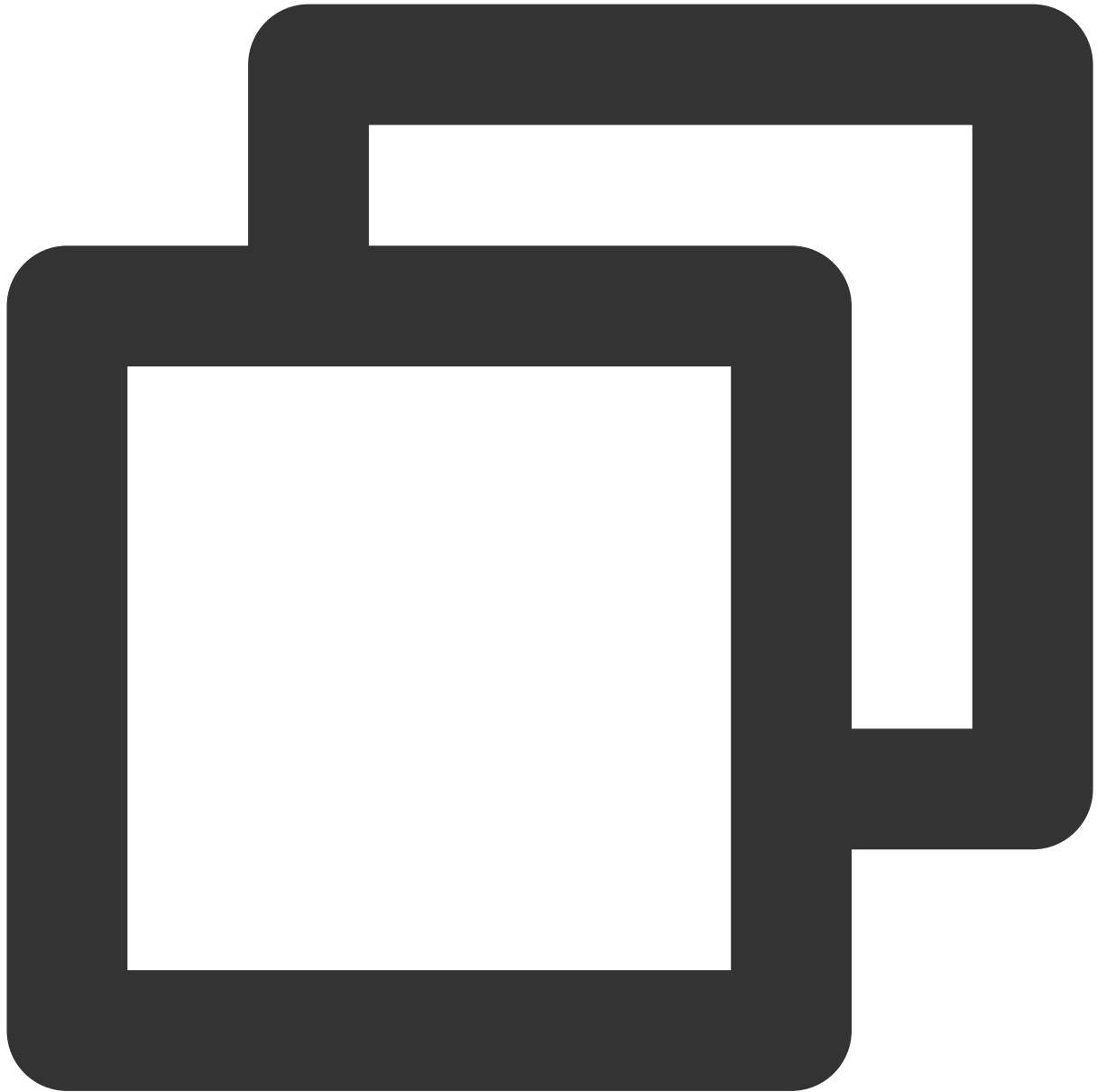


```
JAVA_TOOL_OPTIONS="-XX:+TieredCompilation -XX:TieredStopAtLevel=1"
```

## Disabling class verification

When JVM loads a class to the memory for execution, it will verify whether the class is not tampered with, modified maliciously, or corrupted. However, in a cloud native environment, the CI/CD pipeline is provided by the cloud native platform, which means that the application compilation and deployment are trusted. Therefore, you need to consider

using the following parameters to disable verification. If many classes need to be loaded during startup, disabling verification may accelerate the startup.

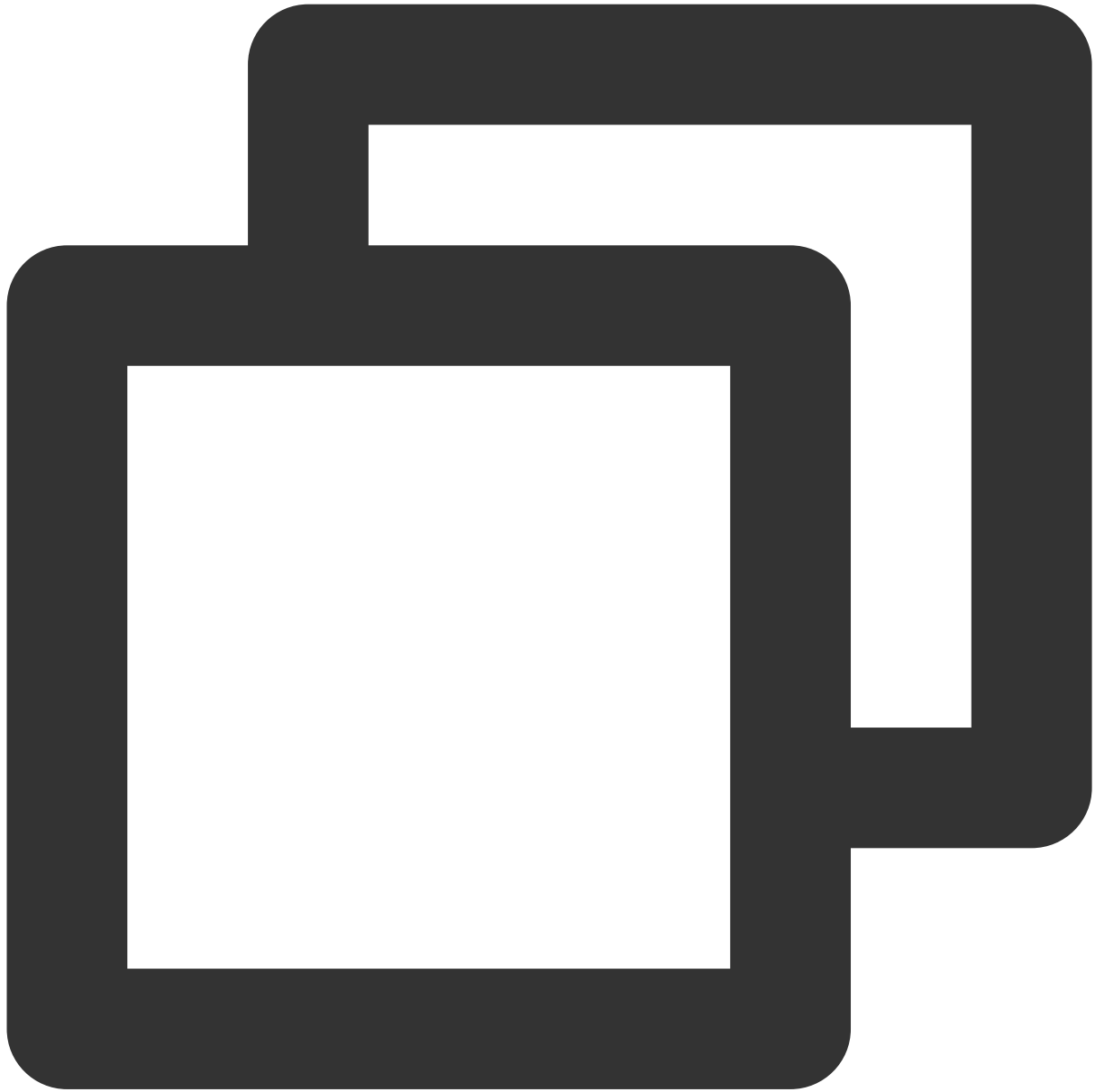


```
JAVA_TOOL_OPTIONS="-noverify"
```

### Reducing the thread size

Most Java web applications use the one-thread-per-connection model, where each Java thread will consume the memory of the local server rather than the heap memory (this is called thread stack), and each thread is 1 MB in size

by default. If your application processes 100 concurrent requests, it may have at least 100 threads, which means that it uses 100 MB thread stack space. The memory is not counted as the heap size, and you can use the following parameter to reduce the thread stack size:



```
JAVA_TOOL_OPTIONS="-Xss256k"
```

Note that if you reduce the size too much, `java.lang.StackOverflowError` will occur. You can analyze the application to find the optimal thread stack size to be configured.

# Optimizing a Spring Boot Application

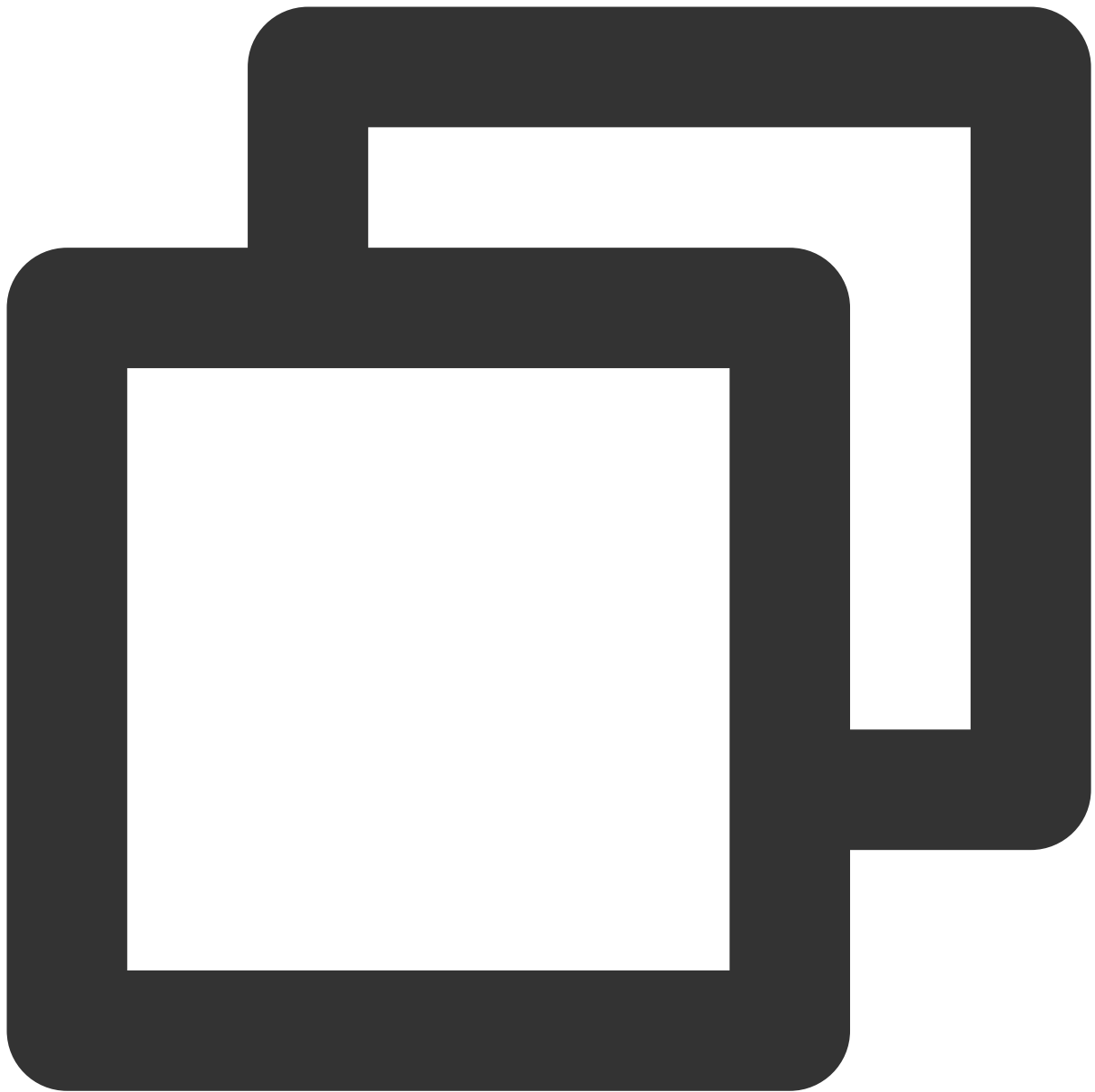
## Using Spring Boot 2.2 or later

Starting from v2.2, Spring Boot has significantly increased the startup speed. However, if you use an earlier version, consider upgrading the version or [making optimizations manually](#).

## Using delayed initialization

On Spring Boot 2.2 or later, you can enable global delayed initialization to increase the startup speed at the cost of lengthening the delay of the first request, as you need to wait for the first initialization of the component.

You can enable delayed initialization in `application.properties` :



```
spring.main.lazy-initialization=true
```

Alternatively, you can use the following environment variable:





```
SPRING_MAIN_LAZY_INITIALIZATION=true
```

# Migration from Java 8 to Java 11

Last updated : 2024-07-04 16:25:38

Since the release of Java 9, Java has been improved and enhanced in many features with some modifications in APIs to improve startup, performance, and memory usage of your applications.

## Significant Improvements From Java 8 to Java 11

### Module system

The module system [JSR 376](#) is integrated to Java since Java 9 to solve problems such as disordered class paths, complex configuration, and ineffective encapsulation in large applications.

A module is a collection of Java classes, APIs, and relevant resources. It can customize the application runtime configuration. It uses a smaller space (which is very useful in the microservice architecture) and allows you to use [jlink](#) to link an application to a custom runtime for deployment. JVM is faster to load a class from a module than to directly load it from a class path.

Modules help implement strong encapsulation by requiring explicit declaration of which packages a module exports and which components it requires, and by restricting reflective access. This level of encapsulation makes an application more secure and easier to maintain.

An application compiled with Java 8 can continue to use the class path and does not have to migrate to modules as a requisite for running on Java 11.

For more information on how the module system works, see [The State of the Module System](#).

### JVM analysis and diagnosis tools

#### Java Flight Recorder and Java Mission Control

Java Flight Recorder (JFR) [JEP 328](#) collects diagnosis and analysis data from running Java applications. It almost has no impact on running applications. You can use Mission Control (JMC) and other tools to analyze collected data. JFR and JMC are commercial features in Java 8 but open-source in Java 11.

### JVM log system

Java 11 has a common logging system [JEP 158](#) for all components of the JVM. This unified logging system allows you to define what components to log, and to what level. This fine-grained logging is useful for performing root-cause analysis on JVM crashes and for diagnosing performance issues in a production environment.

### Low-overhead heap profiling

A new API has been added to the Java Virtual Machine Tool Interface (JVMTI) for sampling Java heap allocations ([JEP 331](#)). The sampling features low overheads.

## Garbage collection

The following garbage collectors are available in Java 11: Serial, Parallel, Garbage-First (G1), and Epsilon. The default garbage collector in Java 11 is G1.

The aim of G1 is to strike a balance between latency and throughput. It is designed to avoid full collections, but when the concurrent collections can't reclaim memory fast enough, a full GC will occur.

The Parallel garbage collector is the default collector in Java 8. It is a throughput collector that uses multiple threads to speed up garbage collection.

The Epsilon garbage collector handles allocations but does not reclaim any memory. When the heap is exhausted, the JVM will shut down. It is useful for short-lived services and for applications that are known to be garbage-free.

In addition, Java 11 provides other three garbage collectors:

ZGC is a concurrent and low-latency collector that attempts to keep pause times under 10 milliseconds. It is available as an experimental feature in Java 11.

Shenandoah is a low-pause collector that reduces GC pause times by performing more garbage collection concurrently with the running Java program. It is an experimental feature in Java 12, but there are backports to Java 11.

CMS is available in Java 11 but has been deprecated since Java 9.

## Improvements on the container environment

Prior to Java 10, memory and CPU constraints set on a container were not recognized by the JVM. In Java 8, for example, the JVM will default the maximum heap size to 1/4 of the physical memory of the underlying host. Starting with Java 10, the JVM uses constraints set by container control groups (cgroups) to set memory and CPU limits. For example, the default maximum heap size is 1/4 of the container's memory limit.

New JVM parameters were also added to Java 10 to give Docker container users fine-grained control over the amount of system memory that will be used for the Java heap.

### Note:

Most of the cgroup enablement work was backported to Java 8 as of JDK 8u191.

## Migration from Java 8 to Java 11

There's no one-size-fits-all solution to migrate applications from Java 8 to Java 11. Potential issues include removed APIs, deprecated packages, use of internal APIs, changes to class loaders, and changes to garbage collectors.

## Trying directly compiling and running the application

In general, the simplest method is to try running the application compiled with Java 8 on Java 11 without recompiling, or to compile with JDK 11 first. If the goal is to get an application up and running as quickly as possible, just trying running on Java 11 is often the best method.

## Other tools

Java 11 has two tools, `jdeprscan` and `jdeps`, which are useful for sniffing out potential issues. These tools can be run against existing class or JAR files. You can assess the migration effort without having to recompile.

### `jdeprscan`

`jdeprscan` looks for use of deprecated or removed APIs. Use of deprecated APIs is not a blocking issue, but is something to look into, as such APIs may be removed in later versions.

The easiest way to use `jdeprscan` is to give it a JAR file from an existing build. You can also give it a directory, such as the compiler output directory, or an individual class name. Use the `--release 11` parameter to get the most complete list of deprecated APIs; for example, you can run `jdeprscan --release 11 my-application.jar`.

If an `error: cannot find class XXX` error occurs, you need to check whether the dependent class file exists in the class path of the JAR file. If the dependent class is not a third-party dependency, you may use an API removed in Java 11.

You can run `jdeprscan --release 11 --list` to get an understanding of what APIs have been deprecated since Java 8. To get the list of APIs that have been removed, run `jdeprscan --release 11 --list --for-removal`.

### `jdeps`

`jdeps` is a Java class dependency analyzer. When used with the `--jdk-internals` parameter, `jdeps` tells you which class depends on which internal APIs. We recommend you add the `--multi-release 11` parameter to support multi-version build JAR files; for example, you can run `jdeps --jdk-internals --multi-release 11 --class-path log4j-core-2.13.0.jar my-application.jar`.

You can continue to use internal APIs in Java 11, but replacing the usage should be a priority. The OpenJDK wiki page [Java Dependency Analysis Tool](#) has recommended replacements for some commonly used JDK internal APIs. Try to eliminate the use of any API coming from the module `jdk.unsupported`. Even though your code may use JDK internal APIs, it will continue to run, for a while at least. Do take a look at [JEP 260](#) since it does point to replacements for some internal APIs.

There are `jdeps` and `jdeprscan` plugins for both Gradle and Maven. We recommend you add these tools to your build scripts.

Tool	Gradle Plugin	Maven Plugin
<code>jdeps</code>	<a href="#">jdeps-gradle-plugin</a>	<a href="#">Apache Maven JDepr Plugin</a>
<code>jdeprscan</code>	<a href="#">jdeprscan-gradle-plugin</a>	<a href="#">Apache Maven JDeprScan Plugin</a>

What `jdeprscan` and `jdeps` cannot do is warn about the use of reflection to access encapsulated APIs. You need to check for reflective access in your code at runtime.

## Check at runtime

### Checking JVM parameters

Check JVM parameters before running your application on Java 11. Using a removed JVM parameter will cause JVM to crash ( `Error: Could not create the Java Virtual Machine` ). If you enable GC logs, parameter check is especially important, as GC logs have changed drastically from Java 8. You can use JaCoLine to check JVM parameters.

### Checking third-party dependent class libraries

You need to update all your third-party dependent class libraries to versions supporting Java 11. The OpenJDK Quality Group maintains a [Quality Outreach](#) wiki page that lists the status of testing of many Free Open Source Software (FOSS) projects against versions of OpenJDK.

### Checking garbage collection parameters

The Parallel garbage collector is the default collector in Java 8. Starting from Java 9, the default garbage collector has been changed to G1. You need to check whether your garbage collection parameters are correct.

### Notes on class loaders

The class loader hierarchy has changed in Java 11. `SystemClassLoader` (also known as `AppClassLoader` ) is now an internal class. Casting to a `URLClassLoader` will report a `ClassCastException` at runtime. Java 11 does not have APIs to dynamically augment the `classpath` at runtime but it can be done through reflection. In Java 11, `BootstrapClassLoader` only loads core modules. If you create a class loader with a null parent, it may not find all platform classes. In Java 11, you need to pass in `ClassLoader.getPlatformClassLoader()` instead of `null` as the parent class loader in such cases.

### Locale data changes

The default source for locale data in Java 11 was changed with [JEP 252](#) to the Unicode Consortium's Common Locale Data Repository. This may have an impact on localized formatting. Set the system property `java.locale.providers=COMPAT, SPI` to revert to the Java 8 locale behavior, if necessary.

## Common issues

### Unrecognized options

If a JVM parameter has been removed, the application will print `Unrecognized option:` or `Unrecognized VM option` . An unrecognized parameter will cause the JVM to exit ( `Error: Could not create the Java Virtual Machine` ). Options that have been deprecated, but not removed, will produce a JVM warning ( `VM Warning: Option <option> was deprecated` ).

In general, such unrecognized JVM parameters need to be removed. The exception is parameters for garbage collection logging. GC logging was reimplemented in [JEP 271](#). Refer to [Enable Logging with the JVM Unified Logging](#)

[Framework](#) to configure parameters.

### **WARNING: An illegal reflective access operation has occurred**

When Java code uses reflection to access a JDK internal API, the runtime will issue an illegal reflective access warning.

#### **java.lang.reflect.InaccessibleObjectException**

This exception indicates that you are trying to call `setAccessible(true)` on a field or method of an encapsulated class/module. Use the `--add-opens` parameter to give your code access to the non-public members of a package/module.

#### **java.lang.NoClassDefFoundError**

If the application runs on Java 8 but reports a `java.lang.NoClassDefFoundError` or `java.lang.ClassNotFoundException` error on Java 11, then it is likely that the application is using a package from the Java EE or CORBA modules. These modules were deprecated in Java 9 and removed in Java 11. For more information, see [JEP 320: Remove the Java EE and CORBA Modules](#).

To resolve the issue, add a runtime dependency to your project.

Removed Module	Affected Package	Suggested Dependency
JAX-WS	java.xml.ws	<a href="#">JAX WS RI Runtime</a>
JAXB	java.xml.bind	<a href="#">JAXB Runtime</a>
JAV	java.activation	<a href="#">JavaBeans (TM) Activation Framework</a>
Common Annotations	java.xml.ws.annotation	<a href="#">Javax Annotation API</a>
CORBA	java.corba	<a href="#">GlassFish CORBA ORB</a>
JTA	java.transaction	<a href="#">Java Transaction API</a>

#### **UnsupportedClassVersionError**

This exception means that you are trying to run code that was compiled with a later version of Java on an earlier version of Java. For example, you are running on Java 11 with a JAR that was compiled with JDK 13.

Java Version	Class File Format Version
8	52
9	53
10	54

---

11	55
12	56
13	57