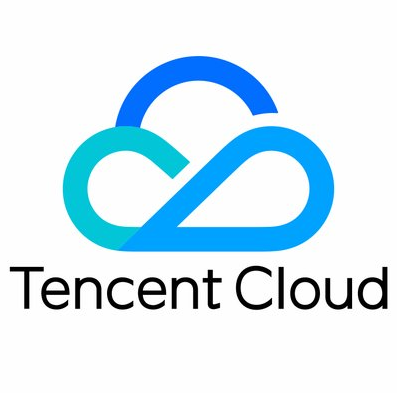
Serverless Cloud Function
Web Framework Development



©2013–2023 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2023 Tencent Cloud. All rights reserved.

The complete copyright of this document, including all text, data, images, and other content, is solely and exclusively owned by Tencent Cloud Computing (Beijing) Co., Ltd. ("Tencent Cloud"); Without prior explicit written permission from Tencent Cloud, no entity shall reproduce, modify, use, plagiarize, or disseminate the entire or partial content of this document in any form. Such actions constitute an infringement of Tencent Cloud's copyright, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Trademark Notice

Tencent Cloud

This trademark and its related service trademarks are owned by Tencent Cloud Computing (Beijing) Co., Ltd. and its affiliated companies("Tencent Cloud"). The trademarks of third parties mentioned in this document are the property of their respective owners under the applicable laws. Without the written permission of Tencent Cloud and the relevant trademark rights owners, no entity shall use, reproduce, modify, disseminate, or copy the trademarks as mentioned above in any way. Any such actions will constitute an infringement of Tencent Cloud's and the relevant owners' trademark rights, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

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

©2013–2023 Tencent Cloud. All rights reserved.

Web Framework Development
Deploying Framework On Command Line

Last updated: 2023-09-27 14:52:26

In addition to the console, you can also rapidly deploy a Web framework via the command line. This document will specifically guide you on how to locally deploy a Web application using the HTTP component of the Serverless Cloud Framework.

Preparations

You have installed the Serverless Cloud Framework, activated the service, and completed the permission configuration for the Serverless Cloud Framework.

Supported Framework

Supported Framework	Documentation
Express	Quickly Deploying Express Framework
Koa	Quickly Deploying Koa Framework
Egg	Quickly Deploying Egg Framework
Next.js	Quickly Deploying Next.js Framework
Nuxt.js	Quickly Deploying Nuxt.js Framework
Nest.js	Quickly Deploying Nest.js Framework
Flask	Quickly Deploying Flask Framework
Django	Quickly Deploying Django Framework
Laravel	Quickly Deploying Laravel Framework

Instructions

1. Developing Applications Locally
   Based on your actual business scenario, complete the development locally. For more details, please refer to the Supported Frameworks development documentation.
2. Configuring the YML File
   In the project root directory, create a new `serverless.yml` file and write the configuration according to the following example. For comprehensive configuration, please refer to the Configuration Document.

```
# serverless.yml
component: http # Component name, which is required
name: webDemo # Component instance name, which is required.

inputs:
  region: ap-guangzhou # The region where the cloud function is located
  src: # Deploy the code files under src, package them into a zip file, and upload to the bucket.
    src: ./ # The local directory of files that need to be packaged
    exclude: # Files or directories to be excluded
      - .env
```

©2013-2023 Tencent Cloud. All rights reserved.

```
    - 'node_modules/**'
  faas: # Function configuration related
    framework: express # Select framework, using express as an example here
    runtime: Nodejs12.16
    name: webDemo # Cloud function name
    timeout: 10 # Timeout period, in seconds
    memorySize: 512 # Memory size, default is 512 MB
    layers:
      - name: layerName # Layer name
        version: 1 # Version

  apigw: # The HTTP component will automatically create an API Gateway service by default
    isDisabled: false # Whether to disable the automatic creation of API Gateway feature
    id: service-xxx # API Gateway service ID, if not filled, a new gateway will be automatically created
    name: serverless # API Gateway service ID
    api: # Configuration related to the created API
      cors: true # Allow cross-origin resource sharing (CORS)
      timeout: 15 # API timeout duration
      name: apiName # API name
      qualifier: $DEFAULT # Version associated with the API
    protocols:
      - http
      - https
    environment: test
```

3. Upon completion, execute `scf deploy` in the root directory for deployment. The component will automatically generate the `scf_bootstrap` startup file for deployment based on the selected framework type.

> **Note**
> As the bootstrap file logic is strongly related to your business logic, the generated default bootstrap file may cause the framework start to fail. We recommend you manually configure the bootstrap file according to your actual business needs. For more information, please see the deployment guide document of each framework.

Example scf_bootstrap

- express:

```
#!/usr/bin/env bash

/var/lang/node12/bin/node app.js
```

- koa

```
#!/usr/bin/env bash

/var/lang/node12/bin/node app.js
```

- egg

```
#!/var/lang/node12/bin/node
```

©2013–2023 Tencent Cloud. All rights reserved.

```
/**
 * Node path in Docker: /var/lang/node12/bin/node
 * Since the serverless function only has read and write permissions for /tmp, it is necessary to modify two environment variables during startup.
 * NODE_LOG_DIR changes the default node write path of egg-scripts (~/logs) to /tmp
 * EGG_APP_CONFIG changes the default directory of the Egg application to /tmp
 */

process.env.EGG_SERVER_ENV = 'prod';
process.env.NODE_ENV = 'production';
process.env.NODE_LOG_DIR = '/tmp';
process.env.EGG_APP_CONFIG = '{"rundir":"/tmp","logger":{"dir":"/tmp"}}';

const { Application } = require('egg');

// If you deploy node_modules through layers, you need to modify eggPath
Object.defineProperty(Application.prototype, Symbol.for('egg#eggPath'), {
  value: '/opt',
});

const app = new Application({
  mode: 'single',
  env: 'prod',
});

app.listen(9000, '0.0.0.0', () => {
  console.log('Server start on http://0.0.0.0:9000');
});
```

- nextjs

```
#!/var/lang/node12/bin/node

/*
# Since the HTTP direct function runs based on a Docker image, it must listen to the address 0.0.0.0 and the port must be set to 9000.
*/
const { nextStart } = require('next/dist/cli/next-start');
nextStart(['--port', '9000', '--hostname', '0.0.0.0']);
```

- nuxtjs

```
#!/var/lang/node12/bin/node

/*
# Since the HTTP direct function runs based on a Docker image, it must listen to the address 0.0.0.0 and the port must be set to 9000.
*/
require('@nuxt/cli')
  .run(['start', '--port', '9000', '--hostname', '0.0.0.0'])
  .catch((error) => {
    require('consola').fatal(error);
    require('exit')(2);
```

©2013–2023 Tencent Cloud. All rights reserved.

```
});
```

- nestjs

```
#!/bin/bash

# SERVERLESS=1 /var/lang/node12/bin/npm run start -- -e /var/lang/node12/bin/node
SERVERLESS=1 /var/lang/node12/bin/node ./dist/main.js
```

- flask

```
#!/bin/bash

# Since the HTTP direct function runs based on a Docker image, it must listen to the address 0.0.0.0 and the
port must be set to 9000.
/var/lang/python3/bin/python3 app.py
```

- django

```
#!/bin/bash

# Since the HTTP direct function runs based on a Docker image, it must listen to the address 0.0.0.0 and the
port must be set to 9000.
/var/lang/python3/bin/python3 manage.py runserver 0.0.0.0:9000
```

- laravel

```
#!/bin/bash

##########################################
# Inject Environment Variables in the Serverless Environment
##########################################
# Injecting the SERVERLESS Identifier
export SERVERLESS=1
# Modify the template compilation cache path, as only the /tmp directory is readable and writable in the
cloud function.
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# Modify the session to be stored in memory (array type)
export SESSION_DRIVER=array
# Log Output to stderr
export LOG_CHANNEL=stderr
# Modifying the Application Storage Path
export APP_STORAGE=/tmp/storage

# Initialize Template Cache Directory
mkdir -p /tmp/storage/framework/views

# Since the HTTP direct function runs based on a Docker image, it must listen to the address 0.0.0.0 and the
port must be set to 9000.
# Cloud executable file path /var/lang/php7/bin/php
/var/lang/php7/bin/php artisan serve --host 0.0.0.0 --port 9000
```

©2013–2023 Tencent Cloud. All rights reserved.

©2013–2023 Tencent Cloud. All rights reserved.

Quickly Deploying Egg Framework

Last updated: 2023-09-28 17:03:16

Overview

This document describes how to quickly deploy a local Egg project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

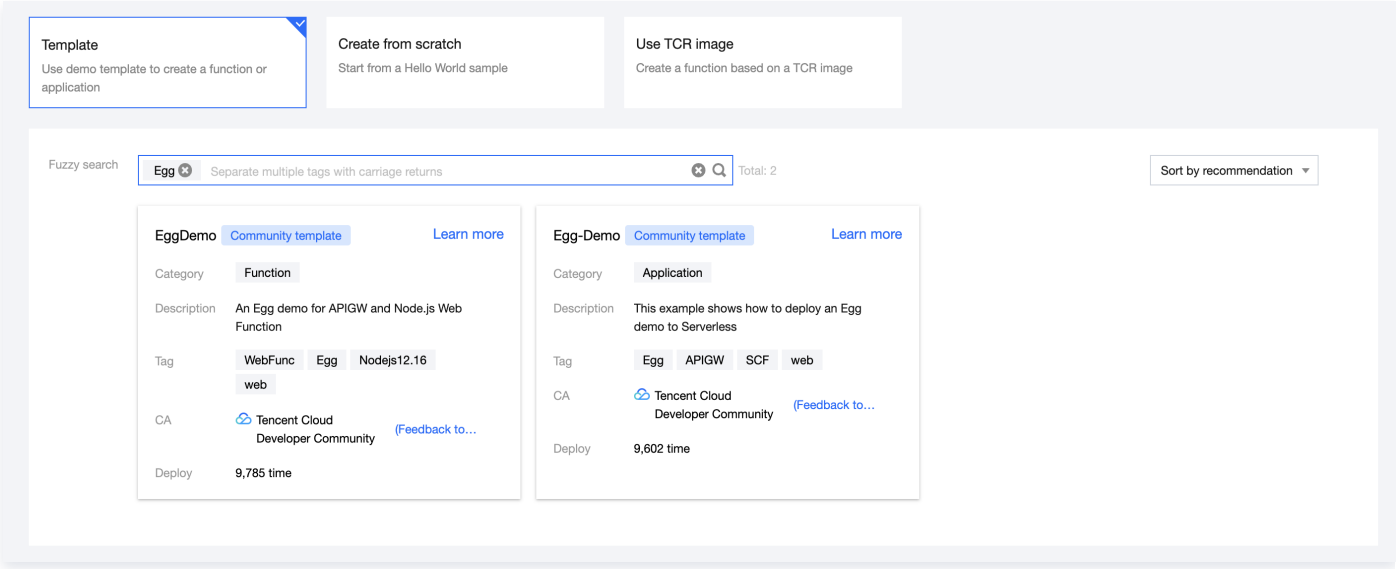
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

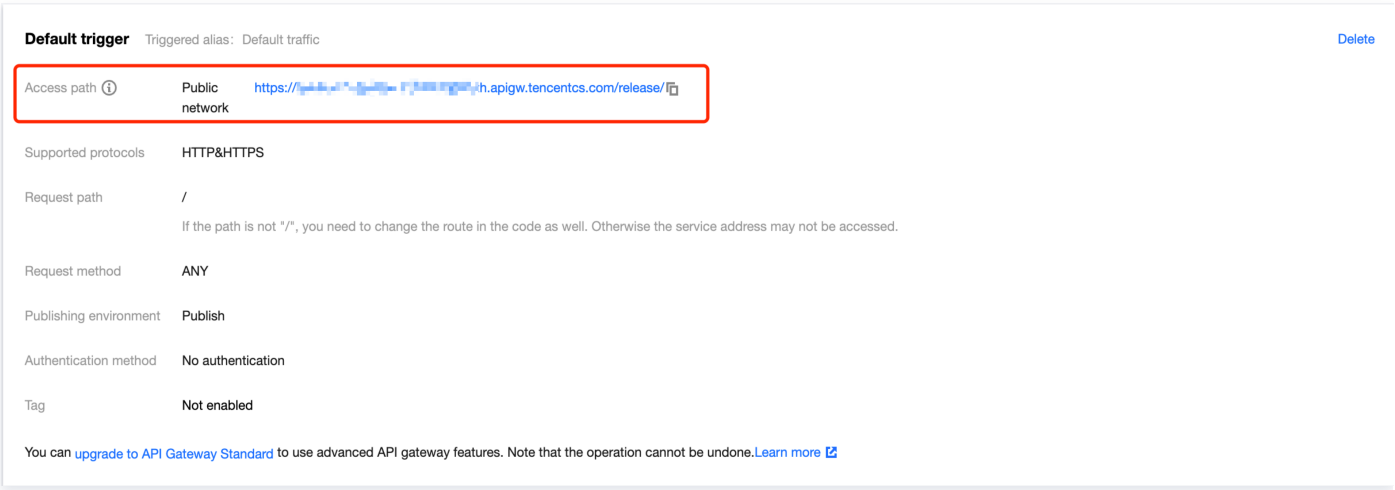
Template Deployment: One-click deployment of Egg projects

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to create a new function using **Template Creation**. Enter `Egg` in the search box to filter function templates, select the **Egg Framework Template**, and click **Next**. As shown in the figure below:



4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click on **Trigger Management** in the left menu to view the access path. You can access your deployed Egg project through the access path URL generated by the API Gateway. As shown in the figure below:

©2013-2023 Tencent Cloud. All rights reserved.



7. Click the access path URL to access the Egg project.

Custom deployment: Quick migration of local project to cloud

Prerequisites

The Node.js runtime environment has been installed locally.

Local development

1. Refer to Quick Start to quickly initialize the sample project as follows:

```
mkdir egg-example && cd egg-example
npm init egg --type=simple
npm i
```

2. In the root directory, run the following command to directly start the service locally.

```
npm run dev
open http://localhost:7001
```

3. Visit `http://localhost:7001` in a browser, and you can access the sample Egg project locally.

Deployment in cloud

Next, perform the following steps to make simple modifications to the initialized project, so that it can be quickly deployed through a HTTP-triggered Function. The project transformation here is usually divided into the following three steps:

- Change the listening address and port to `0.0.0.0:9000` .
- Modify the write path. Only the `/tmp` directory is readable/writable in the SCF environment.
- Add the `scf_bootstrap` file.

The detailed steps are as follows:

1. Create the `scf_bootstrap` bootstrap file in the project root directory and add the following content to it (which is used to configure environment variables and start services. Here is only a sample. Please adjust the specific operations according to your actual business scenario):

```
#!/var/lang/node12/bin/node
```

©2013-2023 Tencent Cloud. All rights reserved.

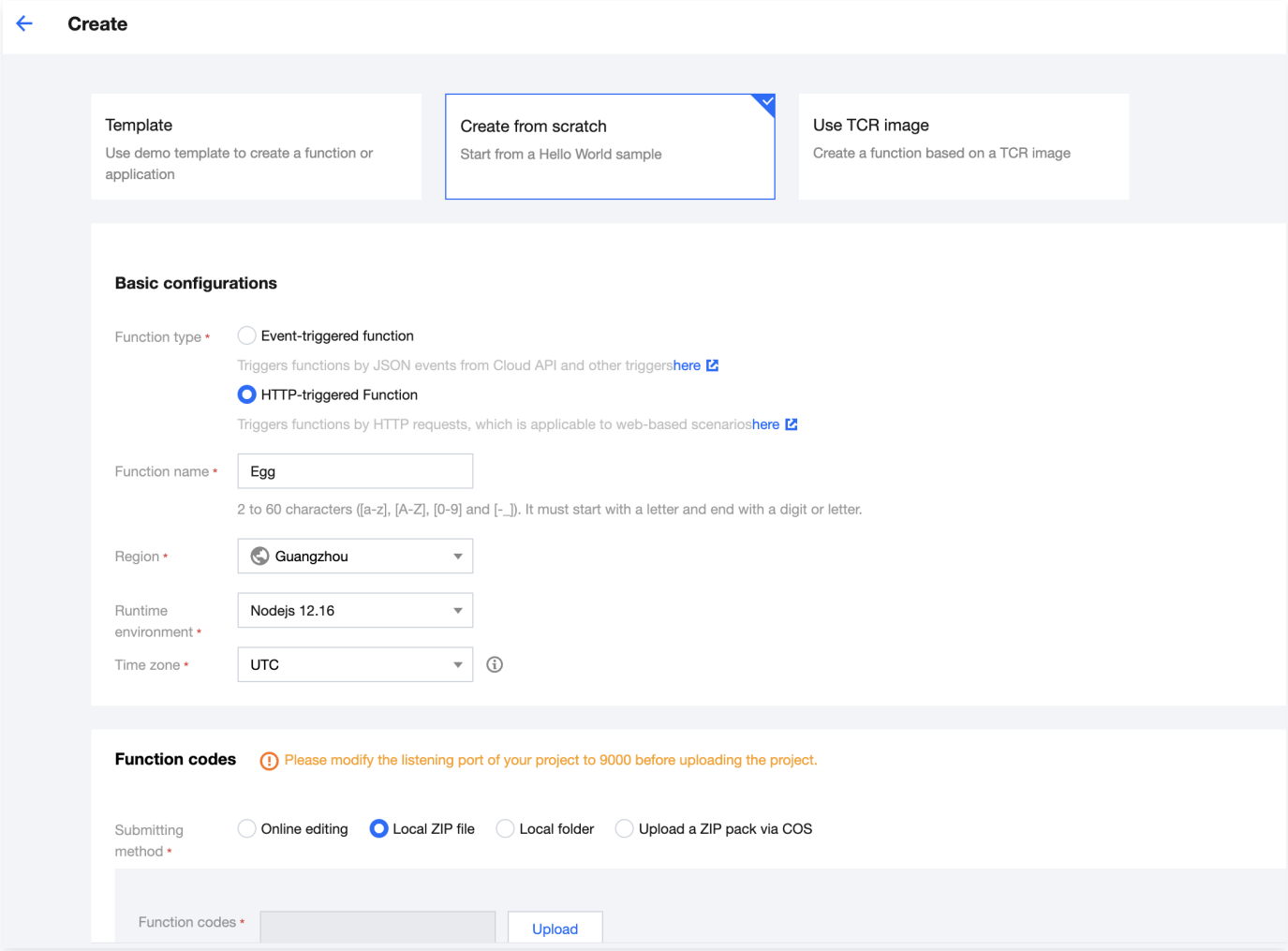
```
'use strict';

/**
* Node path in Docker: /var/lang/node12/bin/node
* Since the serverless function only has read/write permissions for /tmp, two environment variables need to be modified at startup.
* NODE_LOG_DIR changes the default node write path of egg-scripts (~/logs) to /tmp.
* EGG_APP_CONFIG changes the default directory of the Egg application to /tmp.
 */

process.env.EGG_SERVER_ENV = 'prod';
process.env.NODE_ENV = 'production';
process.env.NODE_LOG_DIR = '/tmp';
process.env.EGG_APP_CONFIG = '{"rundir":"/tmp","logger":{"dir":"/tmp"}}';

const { Application } = require('egg');

// If you deploy node_modules through layers, you need to modify eggPath
Object.defineProperty(Application.prototype, Symbol.for('egg#eggPath'), {
  value: '/opt',
});

const app = new Application({
  mode: 'single',
  env: 'prod',
});

app.listen(9000, '0.0.0.0', () => {
  console.log('Server start on http://0.0.0.0:9000');
});
```

2. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

3. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
4. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
5. Choose to **start from scratch** to create a new function, and configure the relevant options as prompted on the page. As shown in the figure below:

©2013–2023 Tencent Cloud. All rights reserved.



- **Function Type**: Select "HTTP-triggered Function".
- **Function name**: Enter the name of your function.
- **Region**: Enter the region where your function is deployed. The default is Guangzhou.
- **Runtime Environment**: Select "Nodejs 12.16".
- **Submission Method**: Select "Upload Local Folder".
- **Function Code**: select the specific local folder where the function code is.

6. Click **Finish** to complete the project deployment.

Development management

After the deployment is completed, you can quickly access and test your web service in the SCF console and try out various features of SCF, such as layer binding and log management. In this way, you can enjoy the advantages of low cost and flexible scaling brought by the serverless architecture.

©2013-2023 Tencent Cloud. All rights reserved.

Quickly Deploying Express Framework

Last updated: 2023-09-28 17:03:24

Overview

This document describes how to quickly deploy a local Express project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

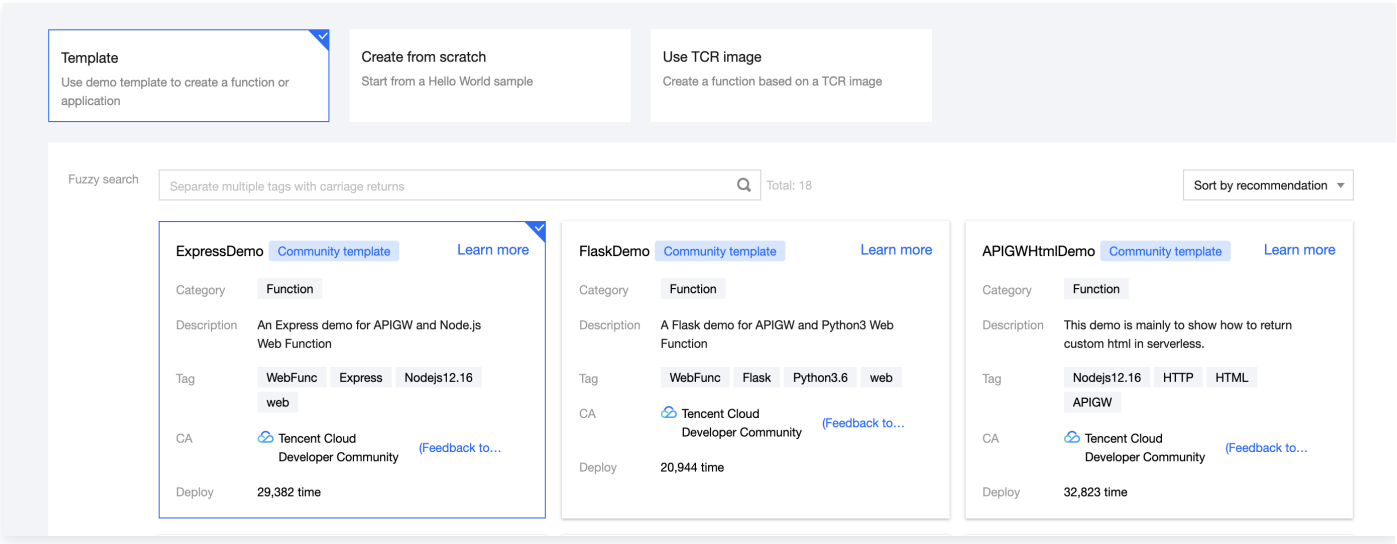
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

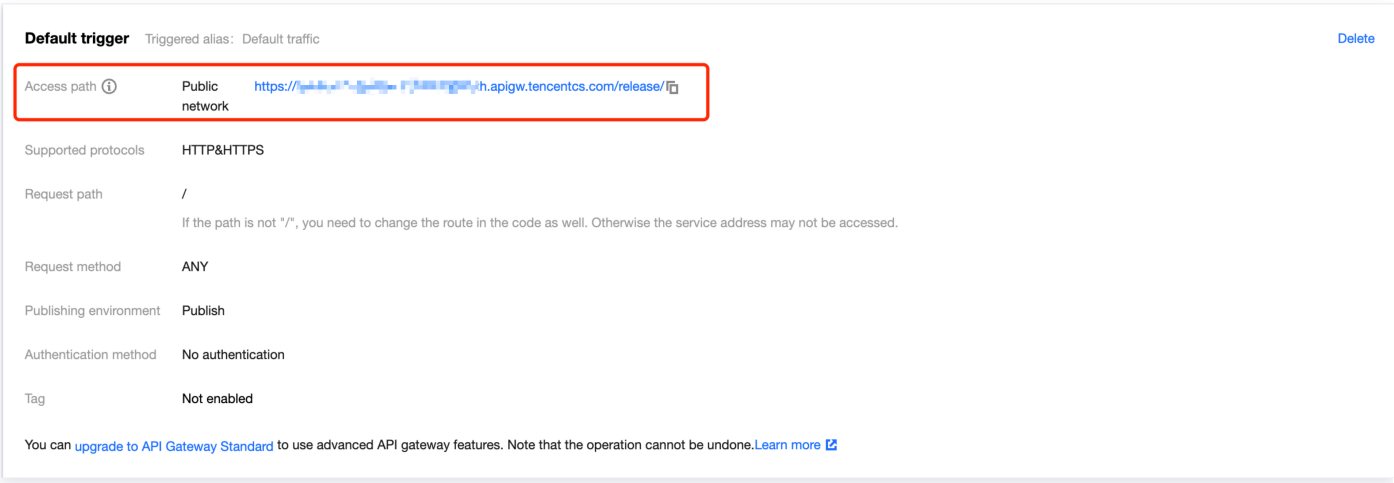
Template deployment: Quick deployment of Express project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to create a new function using **Template Creation**. Enter `WebFunc` in the search box to filter all HTTP-triggered Function templates. Select the **Express Framework Template** and click **Next**. As shown in the figure below:



4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click **Trigger Management** in the left menu bar, view the access path URL, and visit your deployed Express project. As shown in the figure below:

©2013-2023 Tencent Cloud. All rights reserved.



7. Click the access path URL to access the Express project.

Custom deployment: Quick migration of local project to cloud

Prerequisites

The Node.js runtime environment has been installed locally.

Local development

1. Run the following command to install the Express framework and `express-generator` scaffold and initialize the sample Express project.

```
npm install express --save
npm install express-generator --save
express WebApp
```

2. Execute the following command to navigate into the project directory and install the dependency packages.

```
cd WebApp
npm install
```

3. After the installation is completed, run the following command to directly start the service locally.

```
npm start
```

4. Visit `http://localhost:3000` in a browser, and you can access the sample Express project locally.

Deployment in cloud

You need to make simple modifications to the initialized project, so that the project can be quickly deployed through an HTTP-triggered function. The project transformation here is usually divided into the following two steps:

- Change the listening address and port to `0.0.0.0:9000`.
- Add the `scf_bootstrap` file.

The detailed steps are as follows:

1. In the Express sample project, you can set the listening address and port through `./bin/www`. Upon opening this file, you will find that a specific listening port can be set through environment variables, otherwise, it will

©2013-2023 Tencent Cloud. All rights reserved.

automatically listen to port **3000**. As shown below:

```
/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
```

2. Create the `scf_bootstrap` file in the root directory of the project and add the following content to it (the file is used to configure environment variables and start the service):

```
#!/bin/bash
export PORT=9000
npm run start
```

3. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

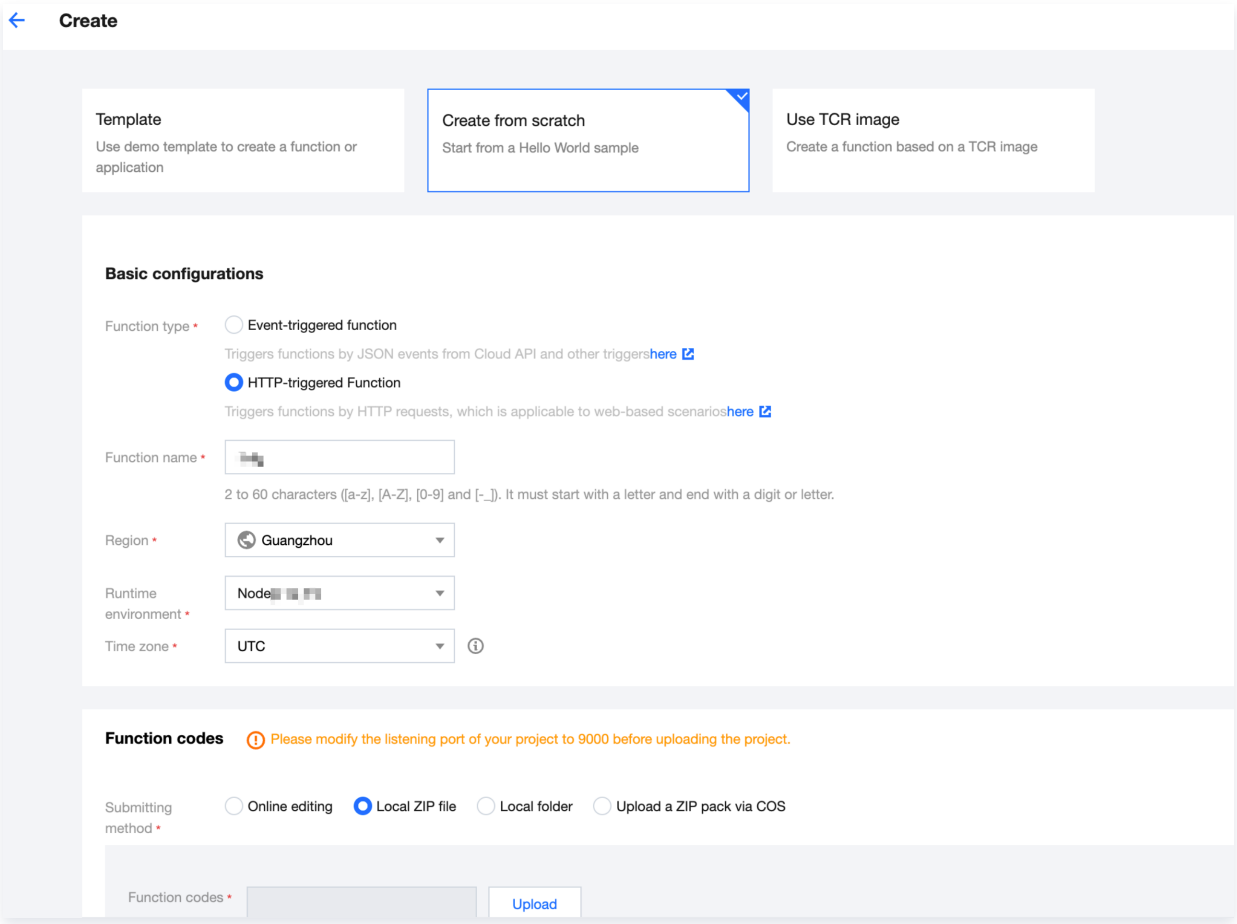
```
chmod 777 scf_bootstrap
```

4. After the local configuration is completed, run the following command to start the file (with execution in the `scf_bootstrap` directory as an example) and make sure that your service can be normally started locally.

```
./scf_bootstrap
```

5. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
6. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
7. Choose to **Start from scratch** to create a new function and configure the relevant options as prompted on the page. As shown below:

©2013–2023 Tencent Cloud. All rights reserved.



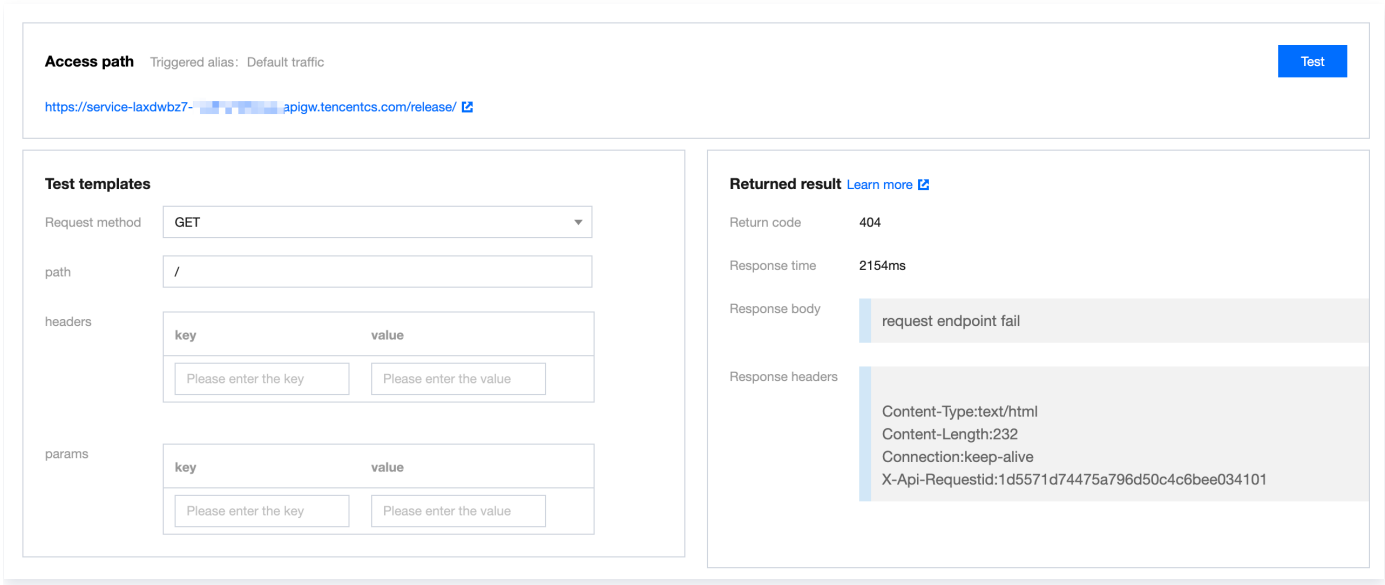
- **Function Type**: Select "HTTP-triggered Function".
- **Function name**: Enter the name of your function.
- **Region**: Enter the region where your function will be deployed. The default is Guangzhou.
- **Runtime Environment**: Select "Nodejs 12.16".
- **Submission Method**: Select "Upload Local Folder" and upload your local project.
- **Function Code**: select the specific local folder where the function code is.

8. Click **Complete**.

Development management

Upon completion of the deployment, you can quickly access and test your web service in the Serverless console. You can also experience various unique features of cloud functions, such as layer binding, log management, and enjoy the advantages of low cost and elastic scaling brought by the Serverless architecture, as shown below:

©2013-2023 Tencent Cloud. All rights reserved.



©2013–2023 Tencent Cloud. All rights reserved.

Quickly Deploying Flask Framework

Last updated: 2023-09-28 17:03:32

Overview

This document describes how to quickly deploy a Flask business to the cloud through an SCF HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

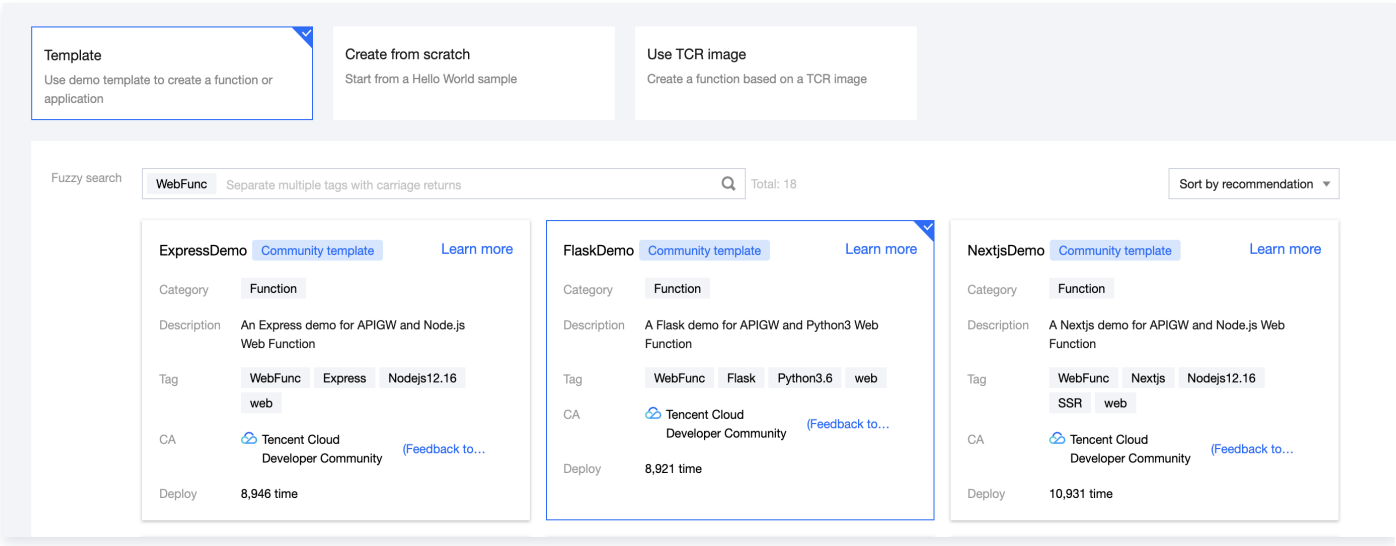
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

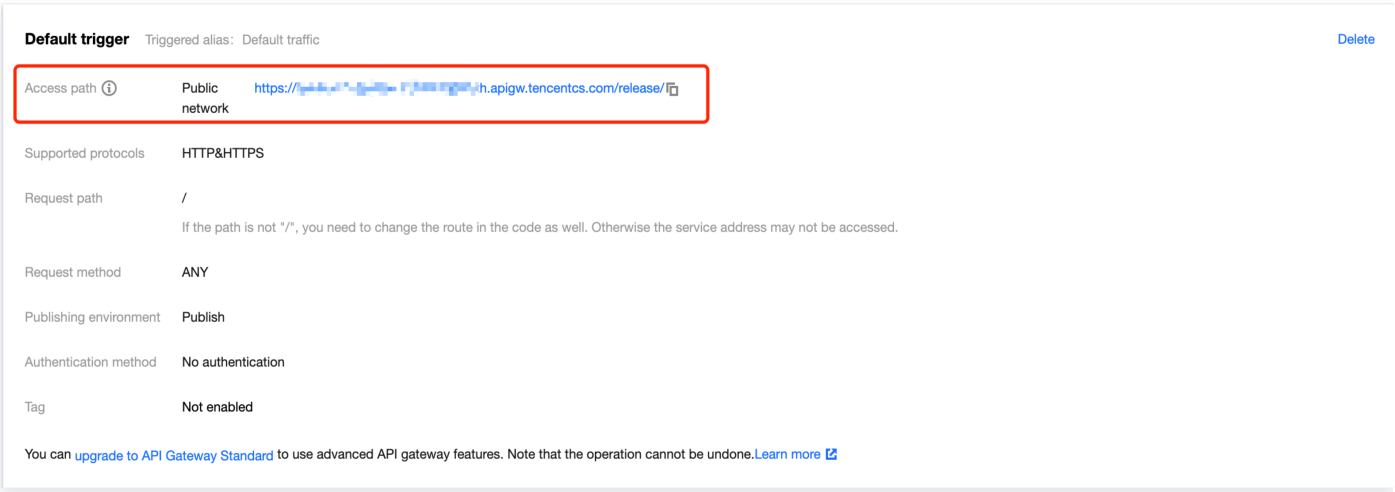
Template Deployment: One-click deployment of Flask projects

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to create a new function using **Template Creation**. Enter `WebFunc` in the search box to filter all HTTP-triggered Function templates, select the **Flask Framework Template** and click **Next**. As shown below:



4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. You can access your deployed Flask project through the automatically generated access path URL. Click on **Trigger Management** in the left menu bar to view the access path. As shown below:

©2013-2023 Tencent Cloud. All rights reserved.



7. Click the access path URL to access the Flask project.

Custom deployment: Quick migration of local project to cloud

Local development

1. Run the following command to confirm that Flask has been installed in your local environment.

```
pip install Flask
```

2. Create the `Hello World` sample project locally.
In the project directory, create a new `app.py` file to implement the Hello World application. The sample code is as follows:

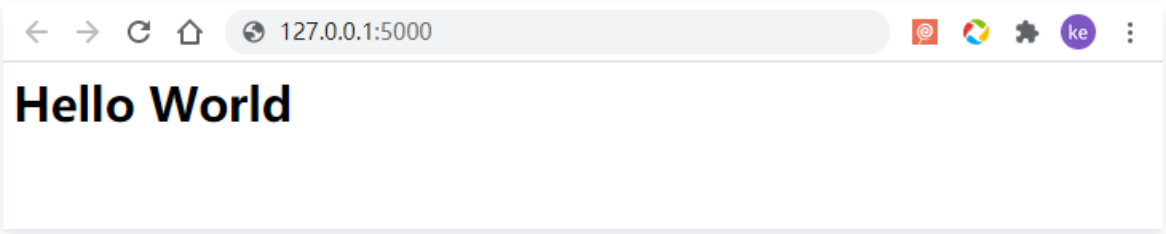
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
  return 'Hello World'

if __name__ == '__main__':
  app.run()
```

3. Execute the `python3 app.py` command locally to run the app.py file. The example is as follows:

```
$ python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Jun/2021 09:41:04] "GET / HTTP/1.1" 200 -
```

©2013–2023 Tencent Cloud. All rights reserved.

4. Visit `http://127.0.0.1:5000` in a browser, and you can access the sample Flask project locally. As shown in the figure below:



Deployment in cloud

Next, perform the following steps to make simple modifications to the locally created project, so that it can be quickly deployed through a HTTP-triggered Function. The steps of project transformation for Flask are as follows:

1. **Install dependencies**

1.1 As the Flask dependency library is not provided in the standard cloud environment of SCF, you must install the dependencies and upload them together with the project code. Please create the `requirements.txt` file first with the following content:

```
#requirements.txt
Flask==1.0.2
werkzeug==0.16.0
```

> **Note**
> Due to the limitation of SCF's built-in runtime environment version (Python 3.6), only lower versions (1.0.x or earlier) of Werkzeug can be used, while higher versions may not work. The runtime environment version upgrade has been planned. Please stay tuned.

1.2 Run the following installation command:

```
pip install -r requirements.txt
```

2. **Modify the listening address and port**

Within the HTTP-triggered Function, the listening port is restricted to **9000**. Therefore, the listening address port needs to be changed to `0.0.0.0:9000`, as shown in the following figure:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World'

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=9000)
```

> **Explanation**
> You can also configure the listening port through the environment variable in `scf_bootstrap`.

©2013-2023 Tencent Cloud. All rights reserved.

3. **Add** a new **scf_bootstrap** **startup file.**
   3.1 Create the `scf_bootstrap` bootstrap file in the project root directory and add the following content to it (which is used to configure environment variables, specify service bootstrap commands, and make sure that your service can be started normally through this file):

```
#!/bin/bash
/var/lang/python3/bin/python3 app.py
```

   3.2 After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

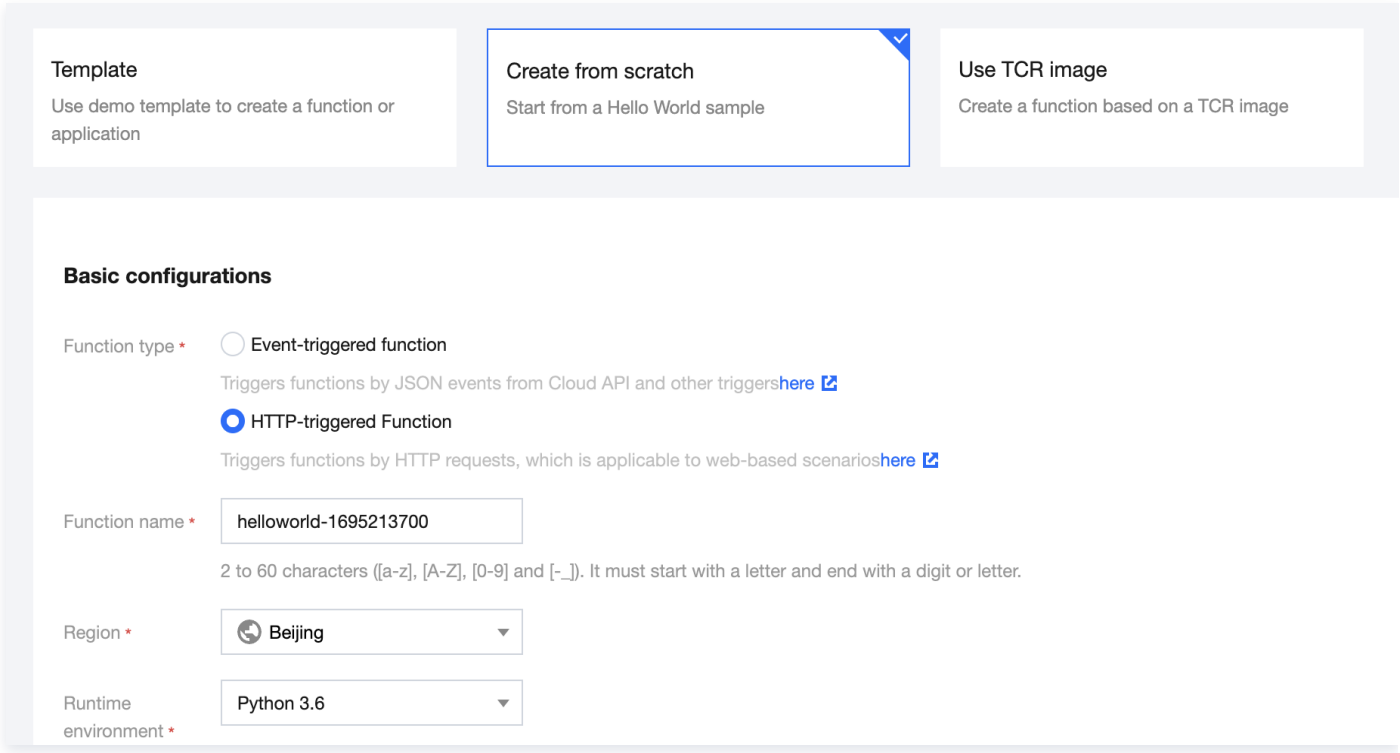
> **Note**
> - In the SCF environment, only files in the `/tmp` directory are readable/writable. We recommend you select `/tmp` when outputting files. If you select other directories, write will fail due to the lack of permissions.
> - If you want to output environment variables in logs, you need to add the `-u` parameter before the bootstrap command, such as `python -u app.py` .

4. After the local configuration is completed, run the following command to start the service (take running the command in the `scf_bootstrap` directory as an example) and make sure that your service can be normally started locally.

```
./scf_bootstrap
```

5. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
6. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
7. Choose to **create a function from scratch** and configure the relevant options according to the page prompts. As shown in the figure below:

©2013–2023 Tencent Cloud. All rights reserved.



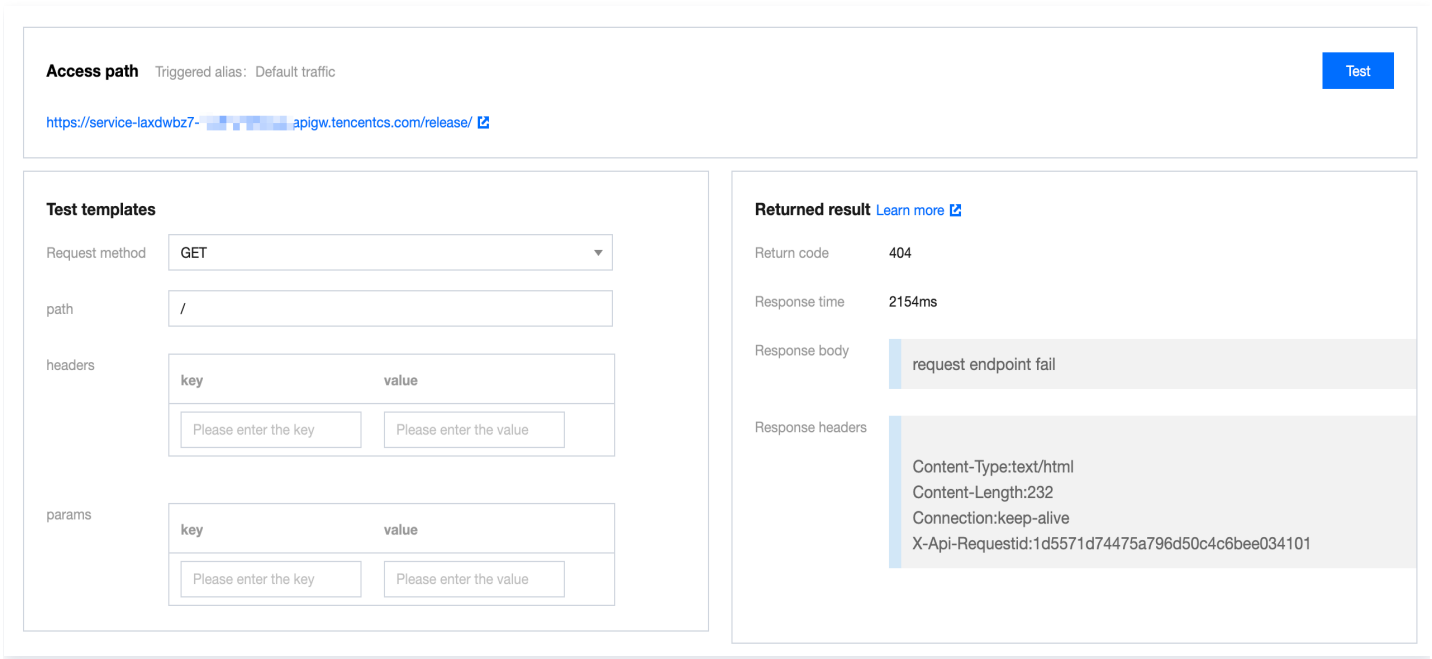
- **Function Type**: Select "HTTP-triggered Function".
- **Function name**: Enter the name of your function.
- **Region**: enter your function deployment region, such as Chengdu.
- **Deployment Method**: Select "Code Deployment" and upload your local project.
- **Runtime Environment**: Select "Python3.6".

8. Click **Complete**.

Development management

Upon completion of the deployment, you can quickly access and test your web service via the SCF console. You can also experience various unique features of cloud functions, such as layer binding, log management, and enjoy the benefits of a Serverless architecture, such as low cost and elastic scaling. As shown in the figure below:

©2013–2023 Tencent Cloud. All rights reserved.



©2013–2023 Tencent Cloud. All rights reserved.

Quickly Deploying Koa Framework

Last updated: 2023-09-28 17:03:40

Operational Overview

This document describes how to quickly deploy a local Koa project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

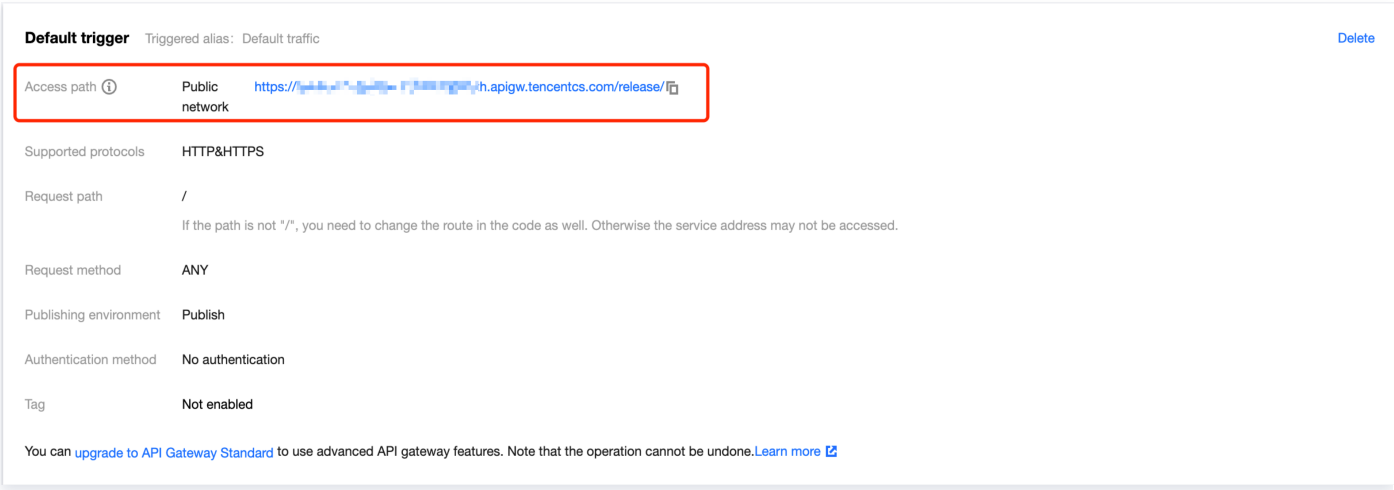
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Koa project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to use **Template Creation** to create a new function. Enter `koa` in the search box to filter function templates, select the **Koa Framework Template**, and click **Next**.
4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click on **Trigger Management** in the left menu bar, check the access path URL, and visit your deployed Koa project, as shown below:



7. Click the access path URL to access the Koa project.

Custom deployment: Quick migration of local project to cloud

Prerequisites

©2013-2023 Tencent Cloud. All rights reserved.

The Node.js runtime environment has been installed locally.

Local development

1. Refer to the Koa.js official documentation to install the Koa environment and initialize your Koa project. The following takes `hello world` as an example. The content of `app.js` is as follows:

```
// app.js
const Koa = require('koa');
const app = new Koa();

const main = ctx => {
  ctx.response.body = 'Hello World';
};

app.use(main);
app.listen(3000);
```

2. In the root directory, run the following command to directly start the service locally.

```
node app.js
```

3. Visit `http://localhost:3000` in a browser, and you can access the sample Koa project locally.

Deployment in cloud

You need to make simple modifications to the initialized project, so that the project can be quickly deployed through an HTTP-triggered function. The project transformation here is usually divided into the following two steps:

- Change the listening address and port to `0.0.0.0:9000` .
- Add the `scf_bootstrap` file.

The detailed steps are as follows:

1. In the Koa sample project, change the listening port to `9000` . As shown in the figure below:

2. Create the `scf_bootstrap` file in the root directory of the project and add the following content to it (the file is used to configure environment variables and start the service):

```
#!/bin/bash
/var/lang/node12/bin/node app.js
```

3. After the file is created, you need to run the following command to modify the executable permission of the file. By

©2013-2023 Tencent Cloud. All rights reserved.

default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

4. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
5. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
6. Choose to **start from scratch** to create a new function, and configure the relevant options as prompted on the page.
   - **Function Type**: Select "HTTP-triggered Function".
   - **Function name**: Enter the name of your function.
   - **Region**: Enter the region where your function is deployed, the default is Guangzhou.
   - **Runtime Environment**: Select "Nodejs 12.16".
   - **Submission Method**: Select "Upload Local Folder" and upload your local project.
   - **Function Code**: select the specific local folder where the function code is.
7. Click **Complete**.

Development management

After the deployment is completed, you can quickly access and test your web service in the SCF console and try out various features of SCF, such as layer binding and log management. In this way, you can enjoy the advantages of low cost and flexible scaling brought by the serverless architecture.

©2013-2023 Tencent Cloud. All rights reserved.

Quickly Deploying Laravel Framework

Last updated: 2023-09-28 17:03:47

Operational Overview

This document describes how to use a HTTP-triggered Function to quickly migrate a local Laravel service to the cloud.

> **Note**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

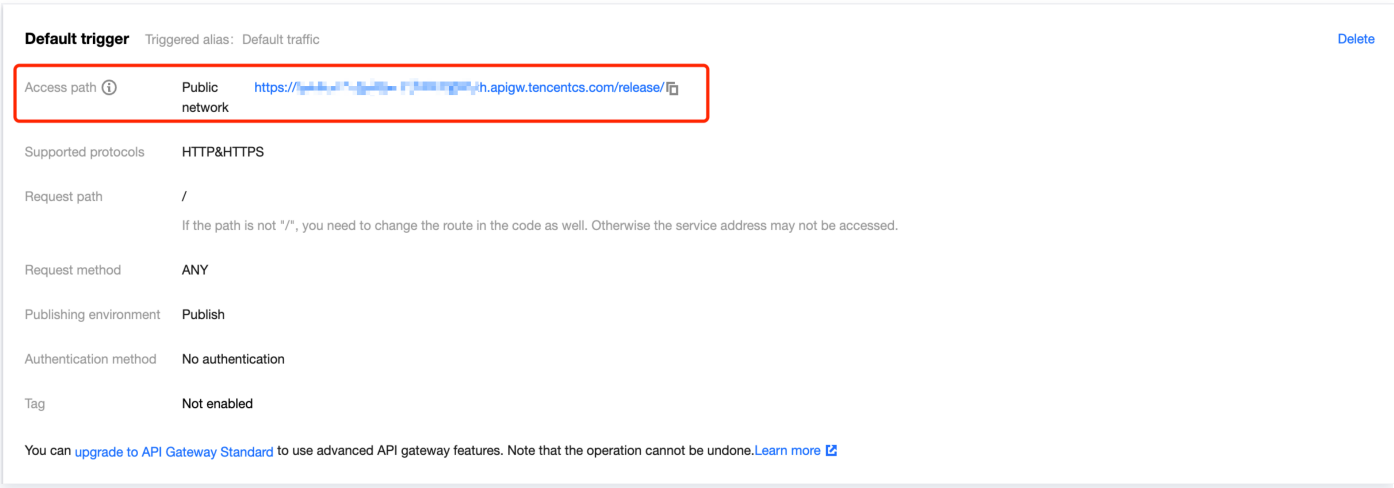
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Laravel project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to create a new function using **Template Creation**. Filter for `WebFunc` in the search box to select all HTTP-triggered Function templates. Choose the **Laravel Framework Template** and click **Next**.
4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click on **Trigger Management** in the left menu bar, view the access path URL, and access your deployed Laravel project, as shown in the following figure:



7. Click the access path URL to access the Laravel project.

Custom deployment: Quick migration of local project to cloud

©2013-2023 Tencent Cloud. All rights reserved.

Local development

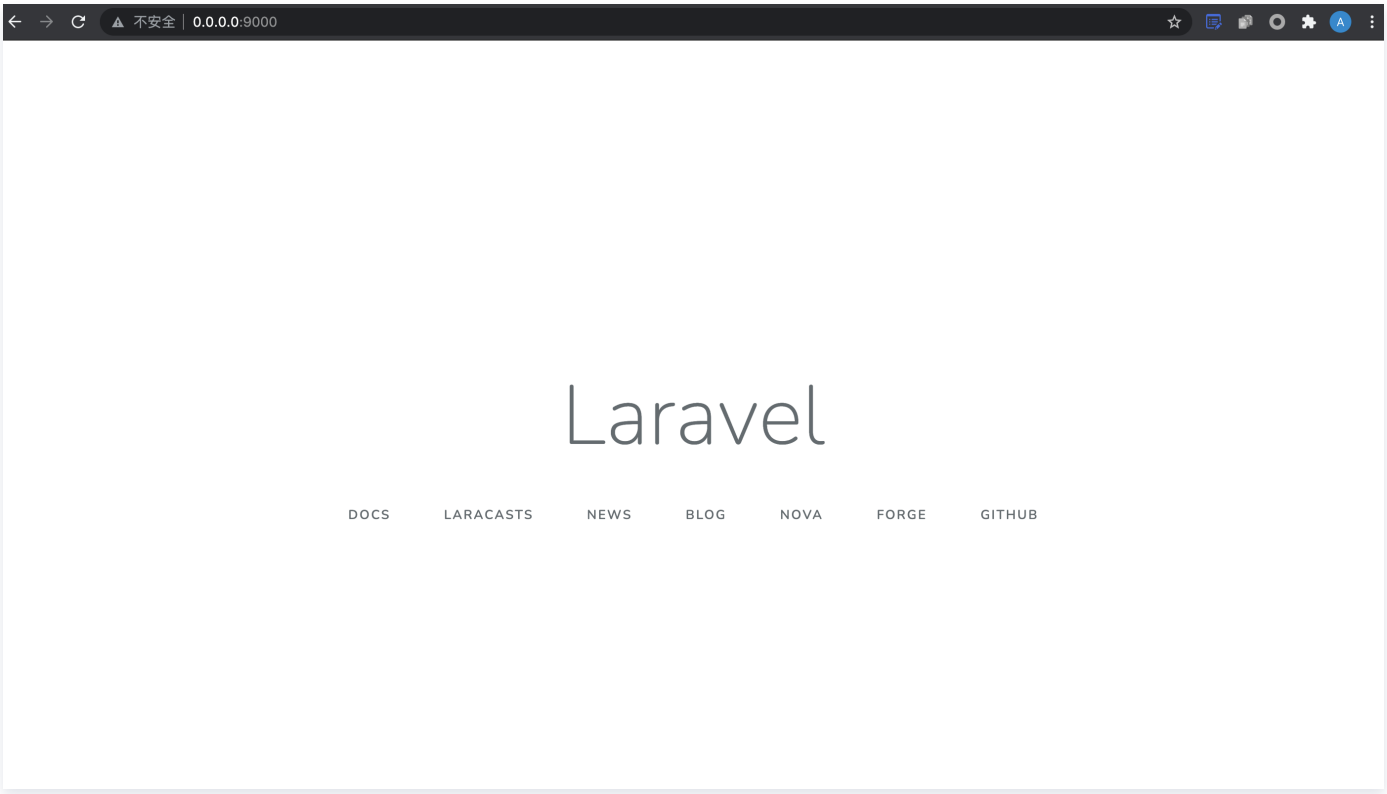
1. Refer to Getting Started on macOS to set up the Laravel development environment locally.
2. Establish a sample Laravel project locally. Navigate to the project directory and execute the following command to initialize the Laravel sample application:

```
composer create-project --prefer-dist laravel/laravel blog
```

3. Run the following command to start the sample project locally. Below is the sample code:

```
$ php artisan serve --host 0.0.0.0 --port 9000
  Laravel development server started: <http://0.0.0.0:9000>
  [Wed Jul  7 11:22:05 2021] 127.0.0.1:54350 [200]: /favicon.ico
```

4. Access `http://0.0.0.0:9000` in your browser to locally view the Laravel sample project, as illustrated below:



Deployment in cloud

Perform the following steps to make simple modifications to the initialized project, so that it can be quickly deployed through an HTTP-triggered function. The steps of project transformation are as follows:

1. **Add a new `scf_bootstrap` startup file**
   Create a `scf_bootstrap` startup file in the root directory of your project. This file should be used to configure environment variables, specify service start commands, and other custom operations to ensure that your service can be launched normally through this file.

> **Note**
> - `scf_bootstrap` must have the executable permission of `755` or `777`.
> - If you want to output environment variables in the log, you need to add the `-u` parameter before the bootstrap command, such as `python -u app.py`.

©2013-2023 Tencent Cloud. All rights reserved.

2. **Modify the File Read/Write Path**
   In the SCF environment, only the `/tmp` directory is writable. Writing to other directories will fail due to lack of permissions. Therefore, it is necessary to inject the output directory of the Laravel framework as an environment variable in the `scf_bootstrap` file:

```
#!/bin/bash

# Injecting the SERVERLESS Identifier
export SERVERLESS=1
# Modify the template compilation cache path, as only the /tmp directory is readable and writable in the
cloud function.
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# Modify the session to be stored in memory (array type)
export SESSION_DRIVER=array
# Log Output to stderr
export LOG_CHANNEL=stderr
# Modifying the Application Storage Path
export APP_STORAGE=/tmp/storage

# Initialize Template Cache Directory
mkdir -p /tmp/storage/framework/views
```

3. **Modify the Listening Address and Port**
   Within the HTTP-triggered Function, the listening port is restricted to `9000`. Therefore, it is necessary to specify the listening port in the `scf_bootstrap` file using the following command:

```
/var/lang/php7/bin/php artisan serve --host 0.0.0.0 --port 9000
```

   The complete content of the `scf_bootstrap` file is as follows:

```
#scf_bootstrap
#!/bin/bash

################################
# Inject Environment Variables in the Serverless Environment
################################
# Injecting the SERVERLESS Identifier
export SERVERLESS=1
# Modify the template compilation cache path, as only the /tmp directory is readable and writable in the
cloud function.
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# Modify the session to be stored in memory (array type)
export SESSION_DRIVER=array
# Log Output to stderr
export LOG_CHANNEL=stderr
# Modifying the Application Storage Path
export APP_STORAGE=/tmp/storage

# Initialize Template Cache Directory
mkdir -p /tmp/storage/framework/views

# The HTTP function runs based on a Docker image, so the listening address must be set to 0.0.0.0 and the
port to 9000.
```

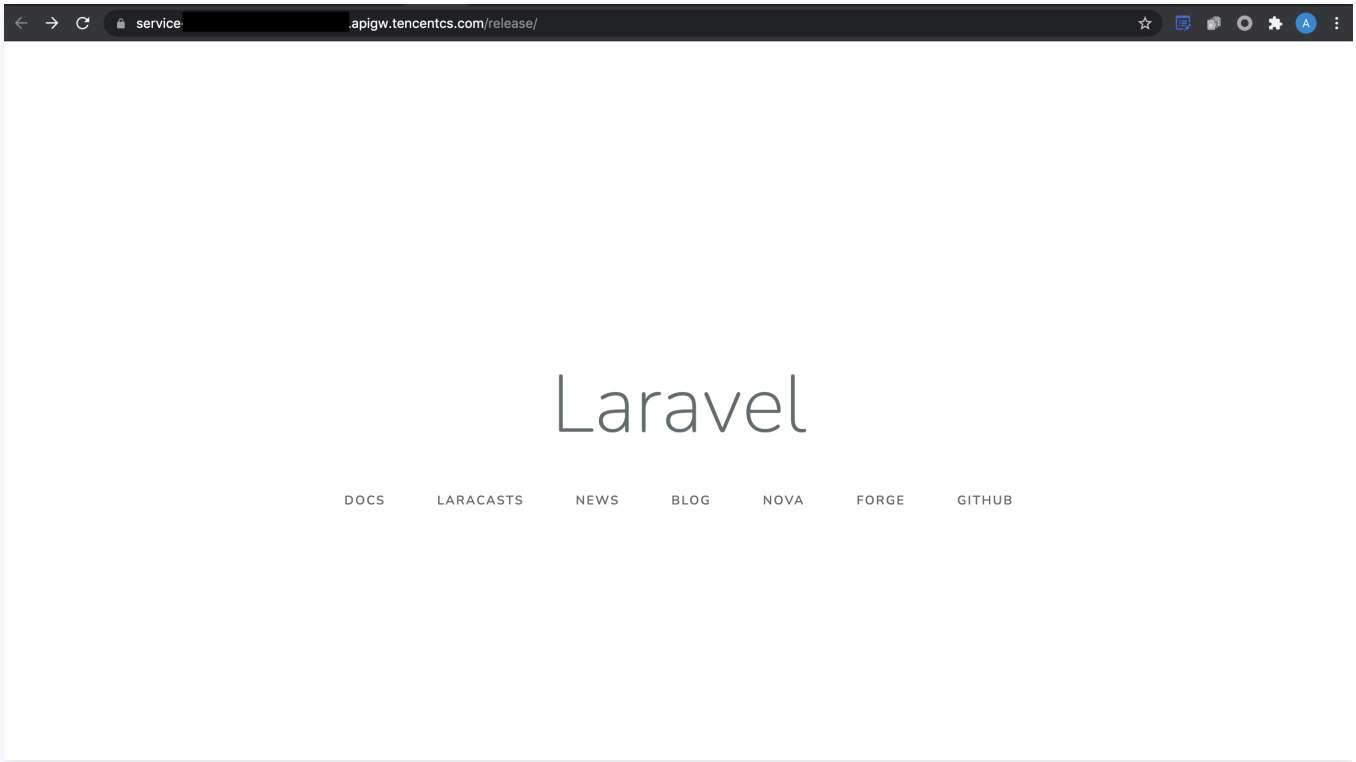
©2013-2023 Tencent Cloud. All rights reserved.

```
# The executable file path in the cloud is /var/lang/php7/bin/php.
/var/lang/php7/bin/php artisan serve --host 0.0.0.0 --port 9000
```

> **Note:**
> The example runs in the Php7.2 environment. If you are using a different language version, please refer to Standard Language Environment Absolute Path to modify the cloud executable file path in the `scf_bootstrap` file.

4. **Deploying Laravel**
   After the local configuration is completed, run the bootstrap file to ensure that your service can be started normally locally. Follow the steps below to deploy Laravel:
   4.1 Log in to the Serverless console and click on **Function Service** in the left navigation bar.
   4.2 Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
   4.3 Choose to **start from scratch** to create a new function, and configure the relevant options according to the page prompts.
   - **Function Type**: Select "HTTP-triggered Function".
   - **Function name**: Enter the name of your function.
   - **Region**: enter your function deployment region, such as Chengdu.
   - **Runtime Environment**: Select "Php7.2".
   - **Submission Method**: Select "Upload Local Folder" and upload your local project.
   - **Function Code**: select the specific local folder where the function code is.

   4.4 Upon completion of the deployment, click the generated URL to access your Laravel application, as shown below:



Development management

©2013-2023 Tencent Cloud. All rights reserved.

After the deployment is completed, you can quickly access and test your web service in the SCF console and try out various features of SCF, such as layer binding and log management. In this way, you can enjoy the advantages of low cost and flexible scaling brought by the serverless architecture.

©2013–2023 Tencent Cloud. All rights reserved.

Quickly Deploying Nest.js Framework

Last updated: 2023-09-28 17:03:55

Operational Overview

This document describes how to quickly deploy a local Nest.js project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

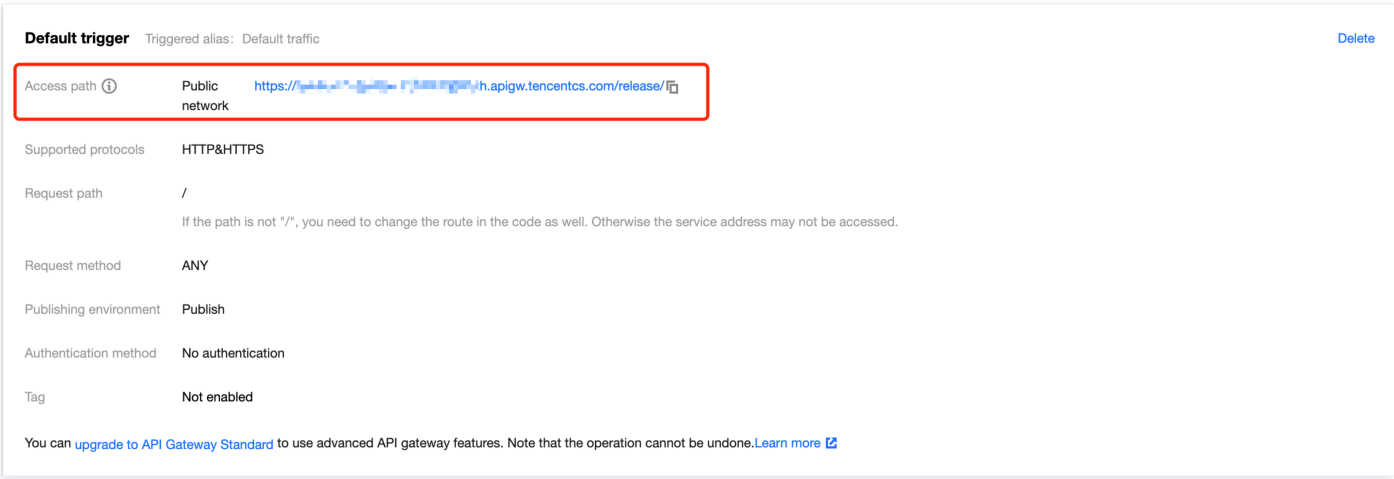
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Nest.js project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to use **Template Creation** to create a new function. Enter `nest` in the search box to filter function templates, select the **Nest Framework Template**, and click **Next**.
4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click **Trigger Management** in the left sidebar, view the access path URL, and visit your deployed Nest.js project. As shown in the figure below:



7. Click the access path URL to access the Nest.js project.

Custom deployment: Quick migration of local project to cloud

Prerequisites

©2013-2023 Tencent Cloud. All rights reserved.

The Node.js runtime environment has been installed locally.

Local development

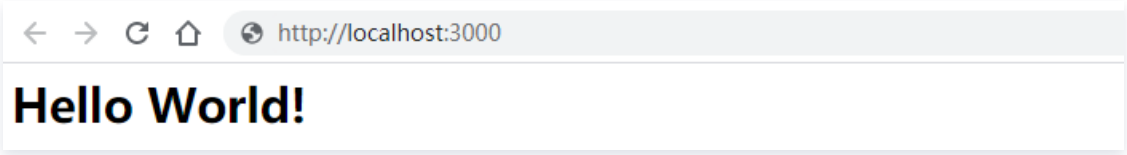
1. Refer to First steps to initialize your Nest.js project:

```
npm i -g @nestjs/cli
nest new nest-app
```

2. In the root directory, run the following command to directly start the service locally.

```
cd nest-app && npm run start
```

3. Visit `http://localhost:3000` in a browser, and you can access the sample Nest.js project locally, as shown below:



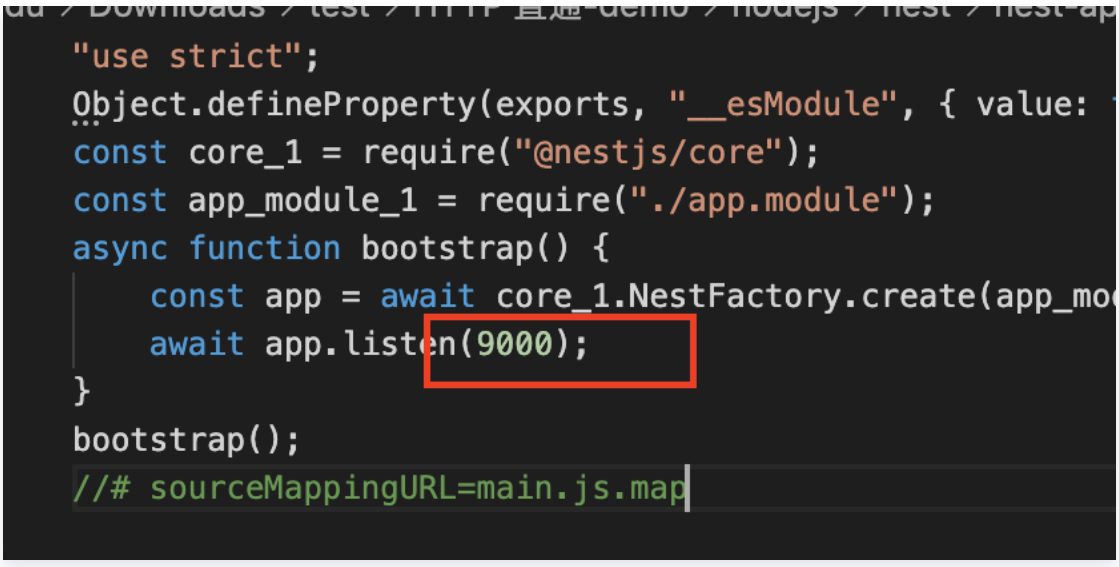
Deployment in cloud

You need to make simple modifications to the initialized project, so that the project can be quickly deployed through an HTTP-triggered function. The project transformation here is usually divided into the following two steps:

- Add the `scf_bootstrap` file.
- Change the listening address and port to `0.0.0.0:9000` .

The detailed steps are as follows:

1. Alter the startup file `./dist/main.js` to change the listening port to `9000` , as shown below:



2. Create the `scf_bootstrap` file in the root directory of the project and add the following content to it (the file is used to start the service):

```
#!/bin/bash
SERVERLESS=1 /var/lang/node12/bin/node ./dist/main.js
```

©2013-2023 Tencent Cloud. All rights reserved.

> **Note**
> - Here is only a sample bootstrap file. Please adjust the configuration according to your actual business scenario.
> - The sample uses the standard node environment path of SCF. When debugging locally, you need to change it to your local path.

3. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

4. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
5. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
6. Choose to **start from scratch** to create a function, and configure the relevant options according to the page prompts.
   - **Function Type**: Select "HTTP-triggered Function".
   - **Function name**: Enter the name of your function.
   - **Region**: Enter the region where your function is deployed, the default is Guangzhou.
   - **Runtime Environment**: Select "Nodejs 12.16".
   - **Submission Method**: Select "Upload Local Folder" and upload your local project.
   - **Function Code**: select the specific local folder where the function code is.
7. Click **Complete**.

Development management

After the deployment is completed, you can quickly access and test your web service in the SCF console and try out various features of SCF, such as layer binding and log management. In this way, you can enjoy the advantages of low cost and flexible scaling brought by the serverless architecture.

©2013-2023 Tencent Cloud. All rights reserved.

Quickly Deploying Next.js Framework

Last updated: 2023-09-28 17:04:03

Operational Overview

This document describes how to quickly deploy a local Next.js SSR project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

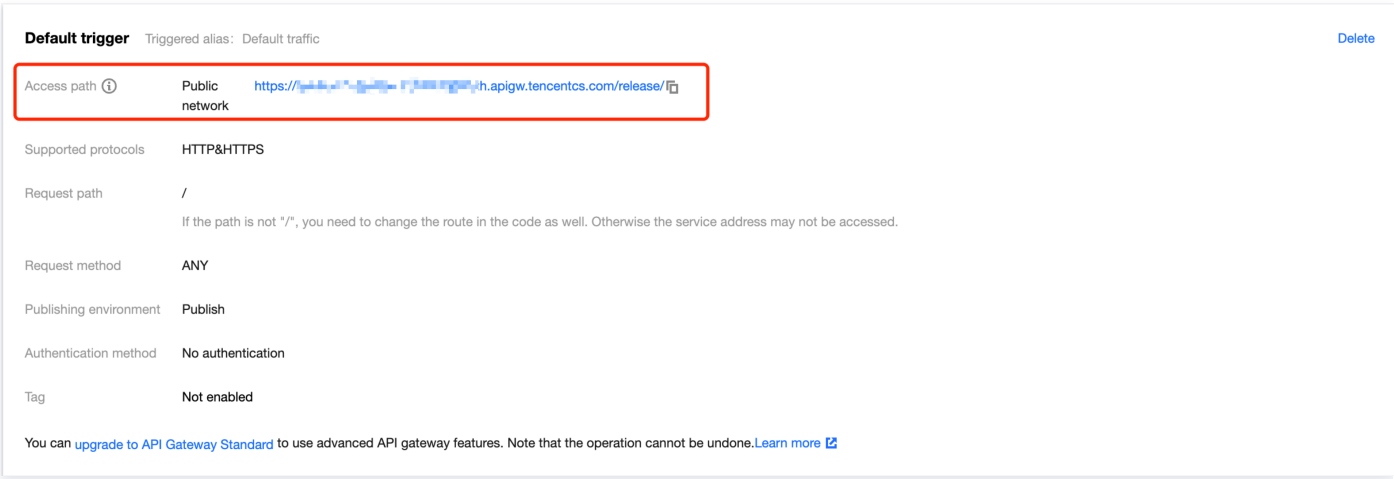
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Next.js project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to use **Template Creation** to create a new function. Enter `webfunc` in the search box to filter function templates, select the **Next.js Framework Template** and click **Next**.
4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click on "Trigger Management" in the left sidebar, check the access path URL, and visit your deployed Next.js project, as shown in the figure below:



7. Click the access path URL to access the Next.js project.

> **Explanation**

©2013-2023 Tencent Cloud. All rights reserved.

> As the Next.js framework needs to be rebuilt before each deployment, please be sure to update the code locally and run `build` again before deploying.

Custom deployment: Quick migration of local project to cloud

Prerequisites

The Node.js runtime environment has been installed locally.

Local development

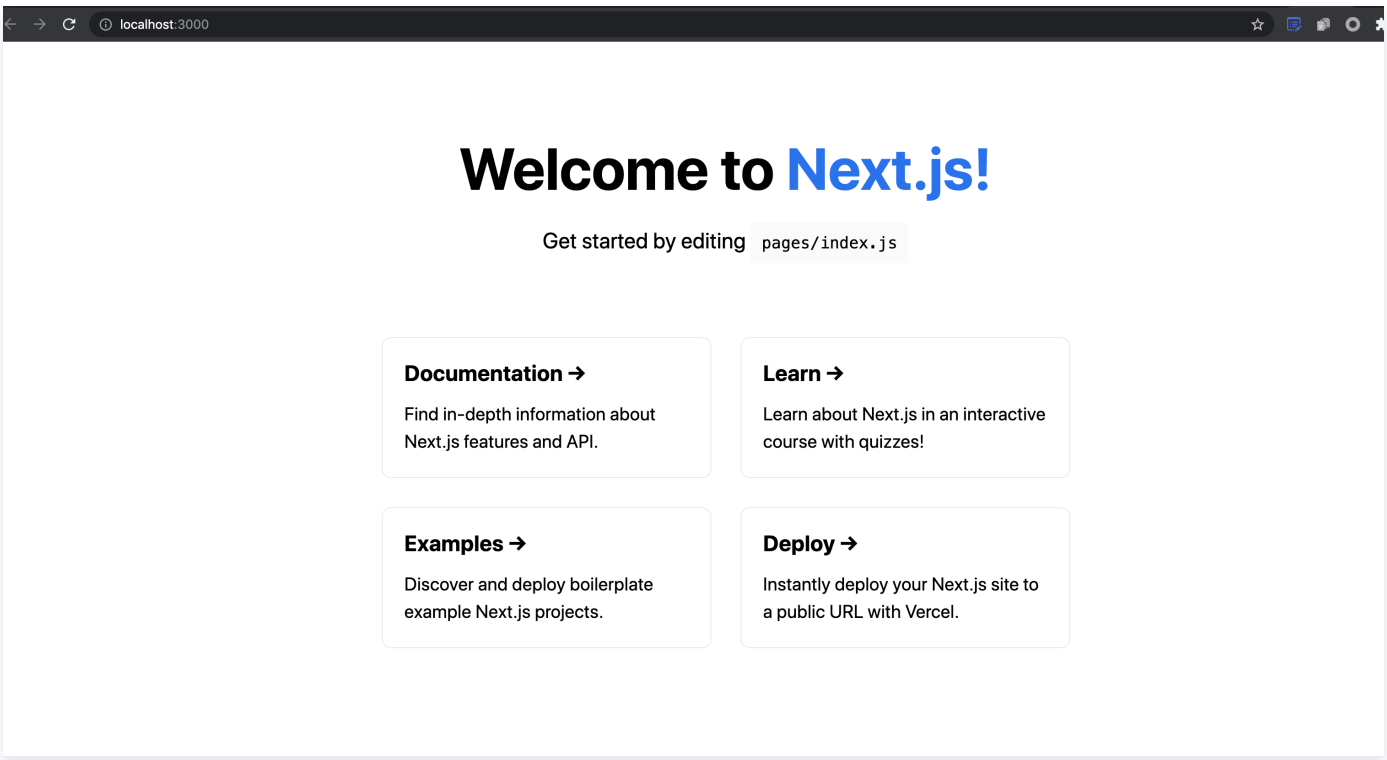
1. Refer to Getting Started to install and initialize your Next.js project:

```
npx create-next-app
```

2. In the root directory, run the following command to directly start the service locally.

```
cd my-app && npm run dev
```

3. Visit `http://localhost:3000` in a browser, and you can access the sample Next.js project locally. As shown below:



Deployment in cloud

You need to make simple modifications to the initialized project, so that the project can be quickly deployed through an HTTP-triggered function. The project transformation here is usually divided into the following two steps:

- Change the listening address and port to `0.0.0.0:9000` .
- Add the `scf_bootstrap` file.

The detailed steps are as follows:

1. Create the `scf_bootstrap` file in the root directory of the project and add the following content to it (the file is used to start the service and specify the start port):

©2013-2023 Tencent Cloud. All rights reserved.

```
#!/var/lang/node12/bin/node
const { nextStart } = require('next/dist/cli/next-start');
nextStart([ '--port', '9000', '--hostname', '0.0.0.0' ])
```

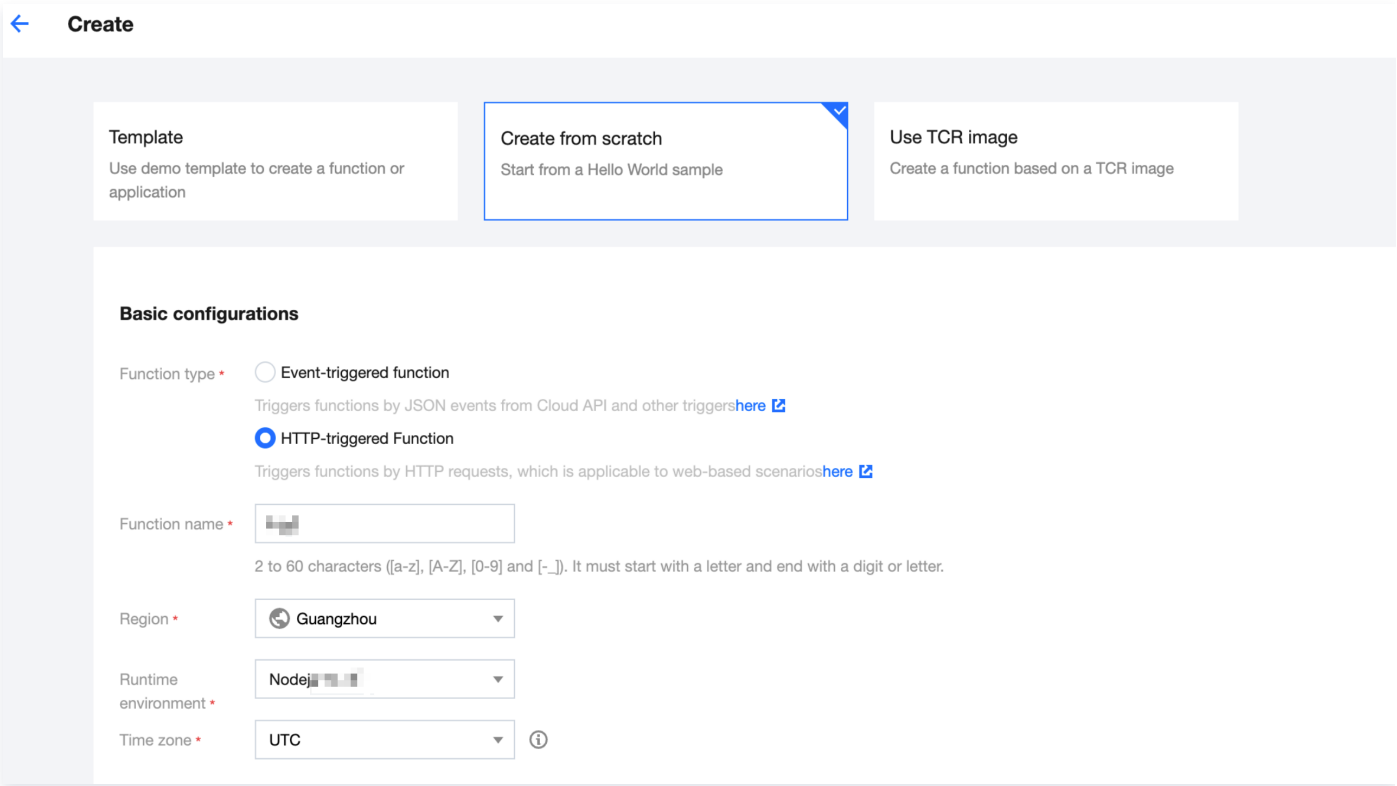
Note

- Here is only a sample bootstrap file. Please adjust the configuration according to your actual business scenario.
- The sample uses the standard node environment path of SCF. When debugging locally, you need to change it to your local path.

2. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

3. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
4. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
5. Choose to **start from scratch** to create a function, and configure the relevant options according to the page prompts. As shown in the figure below:



- **Function Type**: Select "HTTP-triggered Function".
- **Function name**: Enter the name of your function.

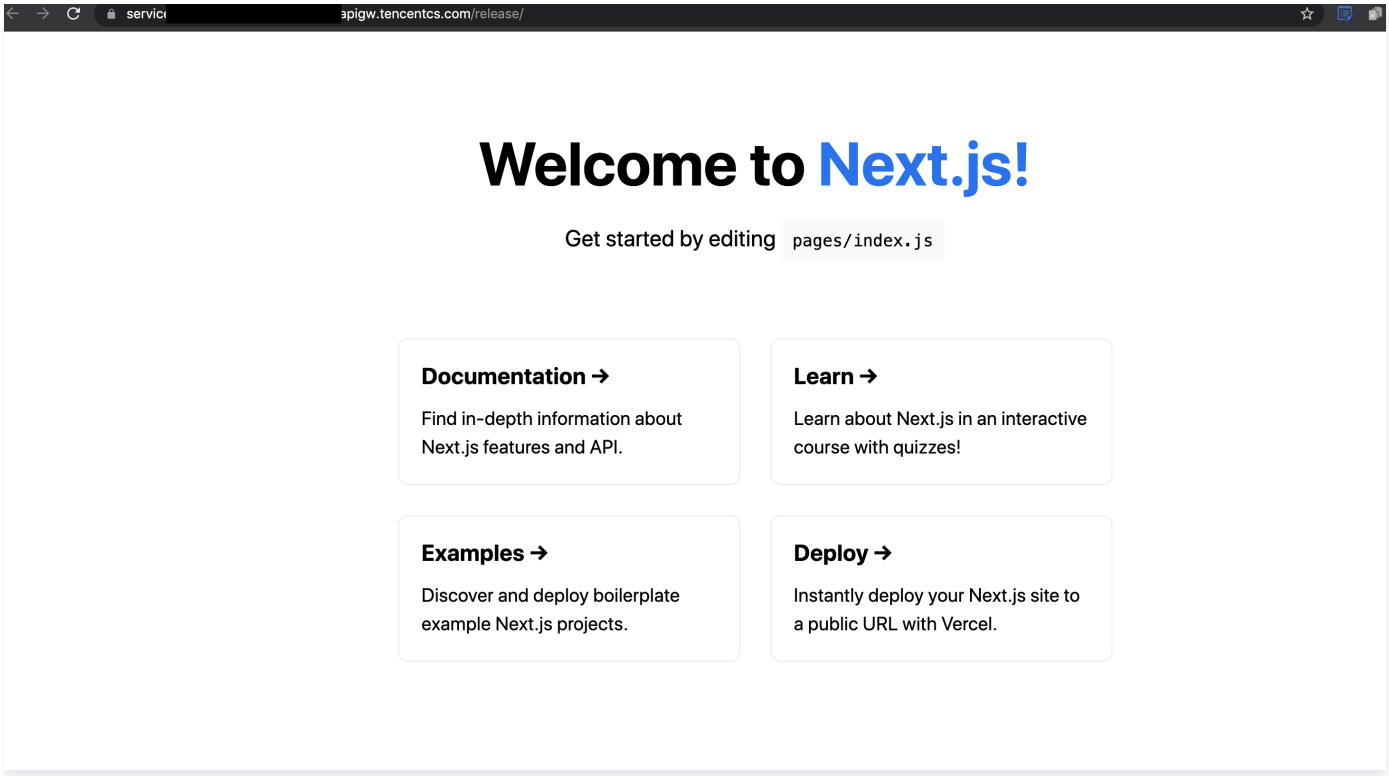
©2013-2023 Tencent Cloud. All rights reserved.

- **Region**: Enter the region where your function is deployed, the default is Guangzhou.
- **Runtime Environment**: Select "Nodejs 12.16".
- **Submission Method**: Select "Upload Local Folder" and upload your local project.
- **Function Code**: select the specific local folder where the function code is.

6. Click **Complete**.

Development management

Upon completion of the deployment, you can swiftly access and test your web services via the SCF console. Experience the various distinctive features of cloud functions, such as layer binding, log management, and more. Enjoy the benefits of a Serverless architecture, including cost-effectiveness and elastic scalability.



©2013-2023 Tencent Cloud. All rights reserved.

Quickly Deploying Nuxt.js Framework

Last updated: 2023-09-28 17:04:10

Overview

This document describes how to quickly deploy a local Nuxt.js SSR project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

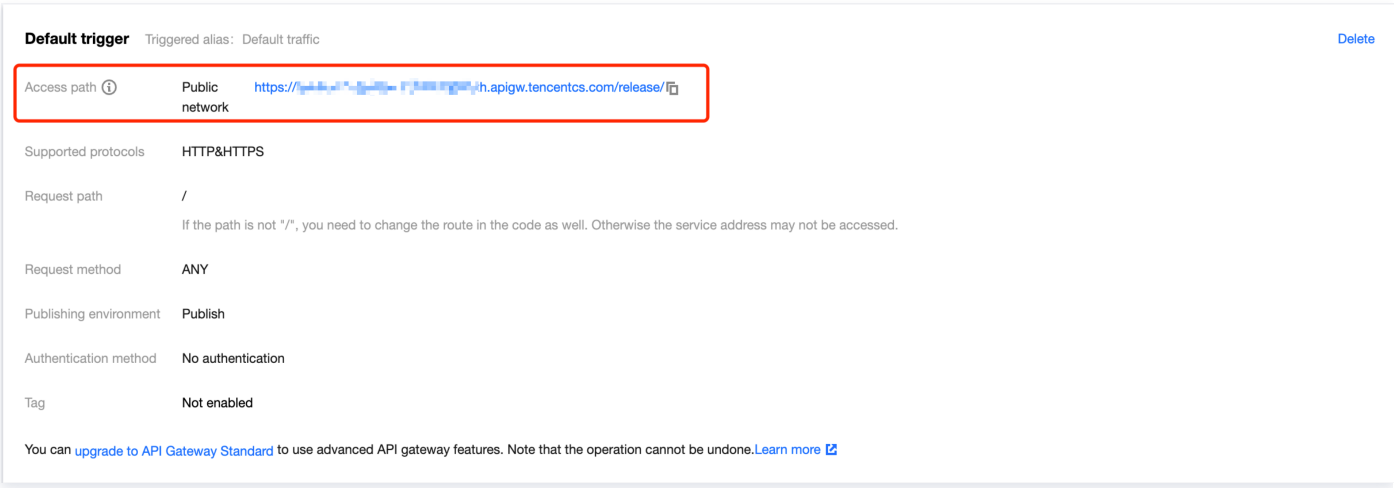
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Nuxt.js project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to use **Template Creation** to create a new function. Enter `webfunc` in the search box to filter function templates, select the **Nuxt.js Framework Template** and click **Next**.
4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function Management** page.
6. Click **Trigger Management** in the left sidebar to view the access path URL and visit your deployed Nuxt.js project, as shown in the following figure:



7. Click the access path URL to access the Nuxt.js project.

> **Explanation**

©2013-2023 Tencent Cloud. All rights reserved.

> As the Nuxt.js framework needs to be rebuilt before each deployment, please be sure to update the code locally and run `build` again before deploying.

Custom deployment: Quick migration of local project to cloud

Prerequisites

The Node.js runtime environment has been installed locally.

Local development

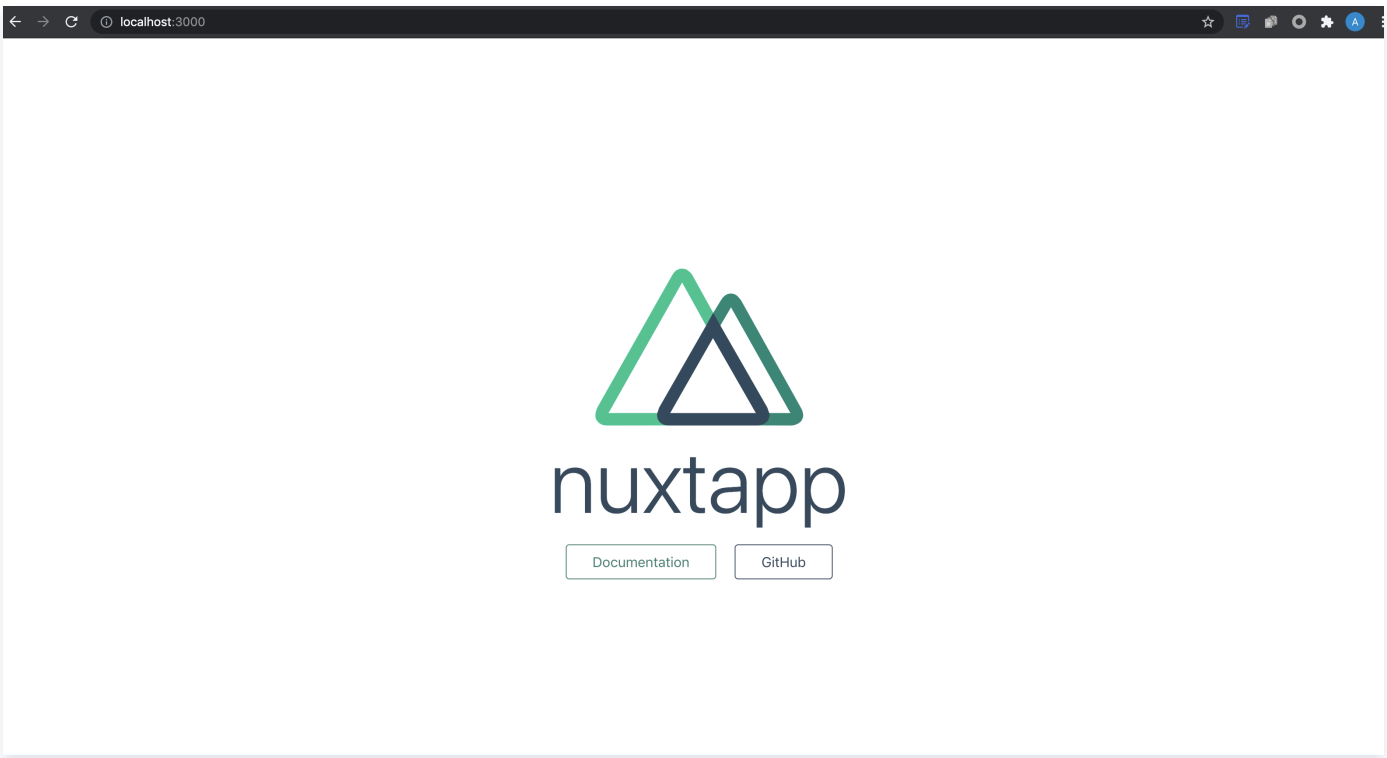
1. Refer to Installation to install and initialize your Nuxt.js project:

```
npx create-nuxt-app nuxt-app
```

2. In the root directory, run the following command to directly start the service locally.

```
cd nuxt-app && npm run dev
```

3. Open a browser and visit `http://localhost:3000` to access the sample Nuxt.js project locally, as shown below:



Deployment in cloud

You need to make simple modifications to the initialized project, so that the project can be quickly deployed through an HTTP-triggered function. The project transformation here is usually divided into the following two steps:

- Add the `scf_bootstrap` file.
- Change the listening address and port to `0.0.0.0:9000` .

The detailed steps are as follows:

1. Create the `scf_bootstrap` file in the root directory of the project and add the following content to it (the file is used to start the service and specify the start port):

©2013-2023 Tencent Cloud. All rights reserved.

```
#!/var/lang/node12/bin/node
require("@nuxt/cli")
  .run(["start", "--port", "9000", "--hostname", "0.0.0.0"])
  .catch(error => {
    require("consola").fatal(error);
    require("exit")(2);
  });
```

> **Note**
> - Here is only a sample bootstrap file. Please adjust the configuration according to your actual business scenario.
> - The sample uses the standard node environment path of SCF. When debugging locally, you need to change it to your local path.

2. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

3. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
4. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
5. Choose to **start from scratch** to create a function, and configure the relevant options according to the page prompts.
   - **Function Type**: Select "HTTP-triggered Function".
   - **Function name**: Enter the name of your function.
   - **Region**: Enter the region where your function is deployed, the default is Guangzhou.
   - **Runtime Environment**: Select "Nodejs 12.16".
   - **Submission Method**: Select "Upload Local Folder" and upload your local project.
   - **Function Code**: select the specific local folder where the function code is.
6. Click **Complete**.

> **Note**
> When you access the URL, the access may fail due to frontend routing. Therefore, you need to remove the `/release` path when accessing.

Development management

After the deployment is completed, you can quickly access and test your web service in the SCF console and try out various features of SCF, such as layer binding and log management. In this way, you can enjoy the advantages of low cost and flexible scaling brought by the serverless architecture.

©2013-2023 Tencent Cloud. All rights reserved.

Quickly Deploying Django Framework

Last updated: 2023-09-28 17:04:19

Overview

This document describes how to quickly deploy a local Django project to the cloud through a HTTP-triggered Function.

> **Explanation**
> This document mainly describes how to deploy in the console. You can also complete the deployment on the command line. For more information, please see Deploying Framework on Command Line.

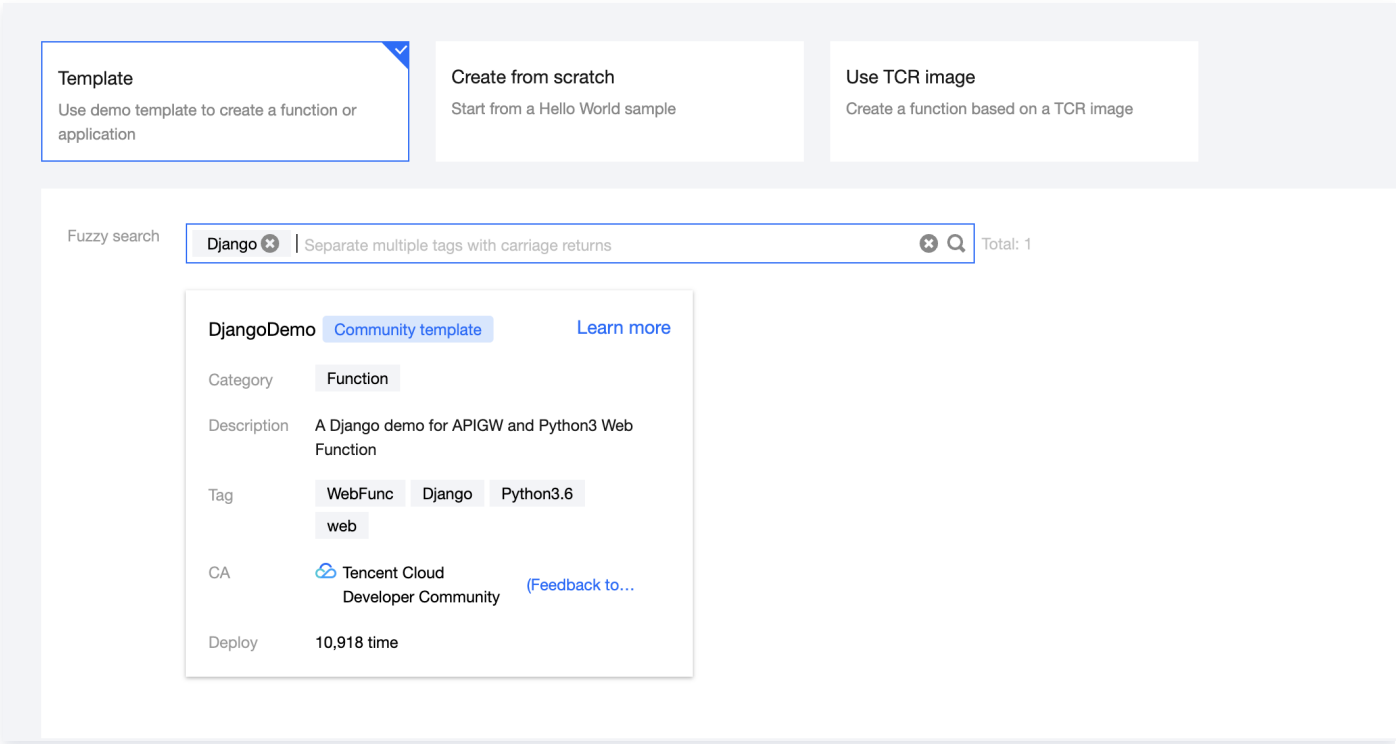
Prerequisites

Before using Tencent Cloud SCF, sign up for a Tencent Cloud account and complete identity verification.

Procedure

Template deployment: Quick deployment of Django project

1. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
2. Select the region and namespace where to create a function at the top of the page and click **Create** to enter the function creation process.
3. Choose to create a new function using **Template Creation**. In the search box, enter `Django`, select the **Django Framework Template**, and click **Next**. As shown in the figure below:

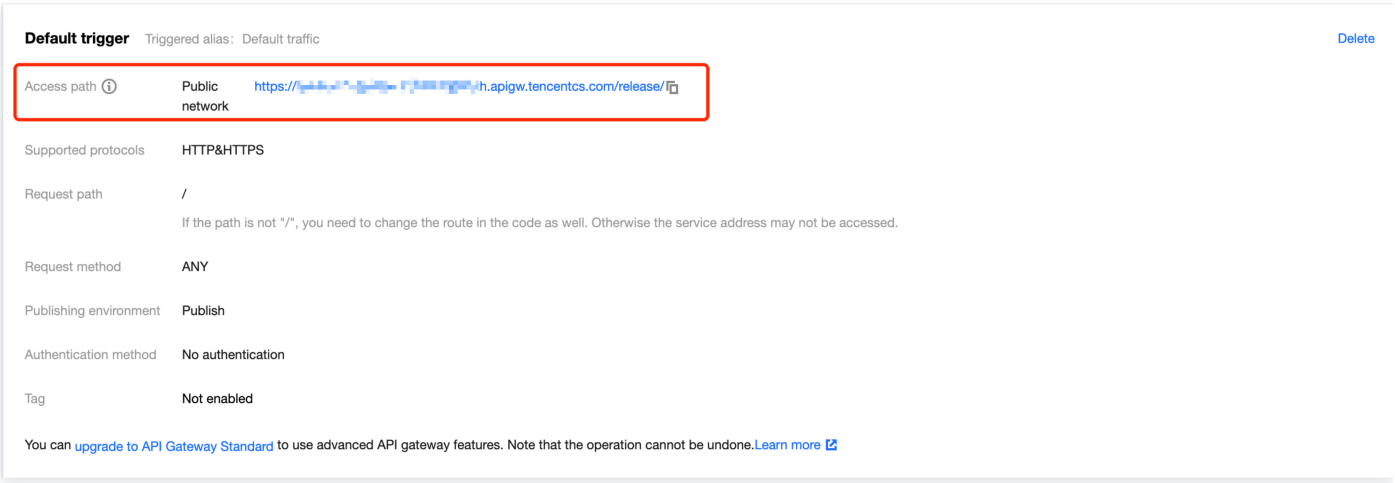


4. On the **Create** page, you can view and modify the specific configuration information of the template project.
5. Click **Complete**. After creating the HTTP-triggered Function, you can view its basic information on the **Function**

©2013-2023 Tencent Cloud. All rights reserved.

Management page.

6. Click on **Trigger Management** in the left sidebar, check the access path URL, and visit your deployed Django project. As shown in the figure below:



7. Click the access path URL to access the Django project.

Custom deployment: Quick migration of local project to cloud

Local development

1. Run the following command to confirm that Django has been installed in your local environment.

```
python -m pip install Django
```

2. Create the `Hello World` sample project locally.

```
django-admin startproject helloworld && cd helloworld
```

The directory structure is as follows:

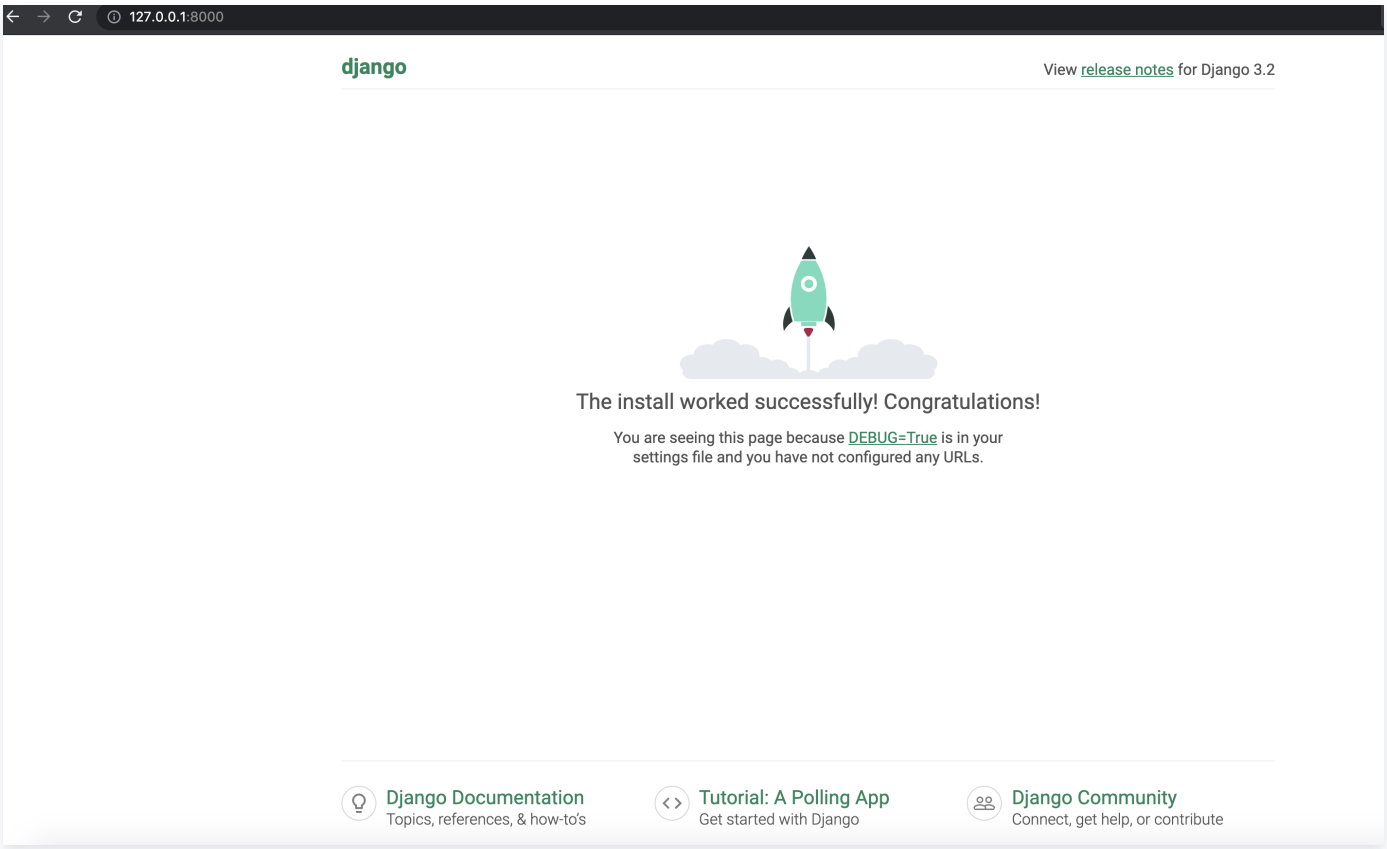
```
$ tree
. manage.py  Manager
|--***
|  |-- __init__.py  Package
|  |-- settings.py  Settings file
|  |-- urls.py  Route
|  `-- wsgi.py  Deployment
```

3. Run the `python manage.py runserver` command locally to start the bootstrap file. Below is the sample code:

```
$ python manage.py runserver
July 27, 2021 - 11:52:20
Django version 3.2.5, using settings 'helloworld.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

©2013–2023 Tencent Cloud. All rights reserved.

4. Open a browser and visit `http://127.0.0.1:8000` to access the Django sample project locally. As shown in the figure below:



Deployment in cloud

You need to make simple modifications to the locally created project, so that the project can be quickly deployed through an HTTP-triggered function. The project modification steps for Django are as follows:

1. **Install dependencies**

   1.1 As the Django dependency library is not provided in the standard cloud environment of SCF, you must install the dependencies and upload them together with the project code. Please create the `requirements.txt` file first with the following content:

   ```
   Django==3.1.3
   ```

   1.2 Run the following installation command:

   ```
   pip install -r requirements.txt -t .
   ```

   > **Explanation**
   > As the initialized default project imports the `db.sqlite3` library, please install this dependency synchronously or configure comments for the `DATABASES` field in the `setting.py` file of the project.

2. **Add `scf_bootstrap` startup file**
   In the HTTP-triggered function, the listening port must be set to **9000**. Therefore, you need to modify the listening address port. Create the `scf_bootstrap` startup file in the root directory of the project and add the following content

©2013-2023 Tencent Cloud. All rights reserved.

to it (this file is used to configure environment variables, specify service start commands, and other custom operations to ensure that your service can be started normally through this file):

```
#!/bin/bash
/var/lang/python3/bin/python3 manage.py runserver 9000
```

3. After the file is created, you need to run the following command to modify the executable permission of the file. By default, the permission `777` or `755` is required for the service to start normally. Below is the sample code:

```
chmod 777 scf_bootstrap
```

> **Note**
> - In the SCF environment, only files in the `/tmp` directory are readable/writable. We recommend you select `/tmp` when outputting files. If you select other directories, write will fail due to the lack of permissions.
> - If you want to output environment variables in logs, you need to add the `-u` parameter before the bootstrap command, such as `python -u app.py` .

4. After the local configuration is completed, run the following command to start the service (take running the command in the `scf_bootstrap` directory as an example) and make sure that your service can be normally started locally.

> **Note**
> When testing locally, be sure to change the Python path to the local path.

```
./scf_bootstrap
```

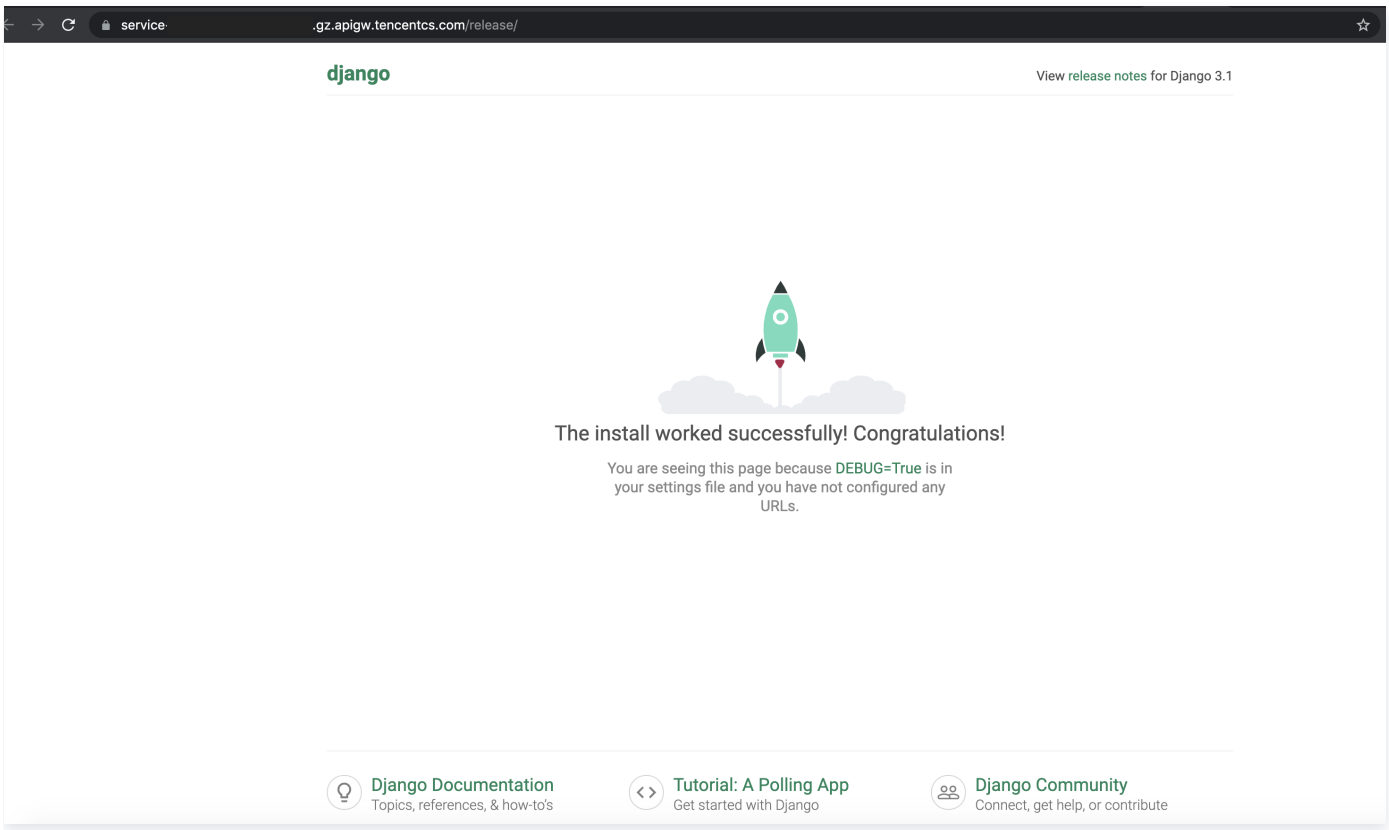
5. Log in to the Serverless console and click on **Function Service** in the left navigation bar.
6. Select the region where to create a function at the top of the page and click **Create** to enter the function creation process.
7. Choose to **start from scratch** to create a new function, and configure the relevant options according to the page prompts.
   - **Function Type**: Select "HTTP-triggered Function".
   - **Function name**: Enter the name of your function.
   - **Region**: enter your function deployment region, such as Chengdu.
   - **Runtime Environment**: Select "Python3.6".
   - **Submission Method**: Select "Upload Local Folder" to upload your local project.
   - **Function Code**: select the specific local folder where the function code is.
8. Click **Complete**.

Development management

Upon completion of the deployment, you can quickly access and test your web service through the SCF console. You can also experience various unique features of cloud functions, such as layer binding, log management, and enjoy the

©2013-2023 Tencent Cloud. All rights reserved.

advantages of a Serverless architecture, such as low cost and elastic scaling, as shown below:



©2013–2023 Tencent Cloud. All rights reserved.