云函数

Web 框架部署

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

Web 框架部署
通过命令行完成框架部署
快速部署 Egg 框架
快速部署 Express 框架
快速部署 Flask 框架
快速部署 Koa 框架
快速部署 Laravel 框架
快速部署 Nestjs 框架
快速部署 Nextjs 框架
快速部署 Nuxtjs 框架
快速部署 Django 框架

版权所有：腾讯云计算（北京）有限责任公司

Web 框架部署
通过命令行完成框架部署

最近更新时间：2022-12-02 16:43:18

除了控制台之外，您也可以通过命令行快速部署 Web 框架，本篇文档将具体为您介绍，如何通过 Serverless Cloud Framework 的 HTTP 组件，完成 Web 应用的本地部署。

前提条件

已安装 Serverless Cloud Framework，已开通服务并完成 Serverless Cloud Framework 的 权限配置。

支持框架

支持框架	相关文档
Express	快速部署 Express 框架
Koa	快速部署 Koa 框架
Egg	快速部署 Egg 框架
Next.js	快速部署 Nextjs 框架
Nuxt.js	快速部署 Nuxtjs 框架
Nest.js	快速部署 Nestjs 框架
Flask	快速部署 Flask 框架
Django	快速部署 Django 框架
Laravel	快速部署 Laravel 框架

操作步骤

1. 本地开发应用
   根据您实际业务场景，本地完成开发，详情可参考 支持框架 开发文档。
2. 配置 yml 文件
   在项目根目录下，新建 `serverless.yml` 文件，按照以下示例进行配置编写。全量配置请参考 配置文档。

```
# serverless.yml
component: http # （必选）组件名称
name: webDemo # 必选）组件实例名称.

inputs:
  region: ap-guangzhou # 云函数所在区域
  src: # 部署src下的文件代码，并打包成zip上传到bucket上
    src: ./ # 本地需要打包的文件目录
    exclude: # 被排除的文件或目录
      - .env
      - 'node_modules/**'
  faas: # 函数配置相关
    framework: express #选择框架，此处以 express 为例
    runtime: Nodejs12.16
```

```
    name: webDemo # 云函数名称
    timeout: 10 # 超时时间，单位秒
    memorySize: 512 # 内存大小，默认 512 MB
    layers:
      - name: layerName #  layer名称
        version: 1 #  版本

  apigw: #  # http 组件会默认帮忙创建一个 API 网关服务
    isDisabled: false # 是否禁用自动创建 API 网关功能
    id: service-xxx # api网关服务ID，不填则自动新建网关
    name: serverless # api网关服务ID
    api: # 创建的 API 相关配置
      cors: true #  允许跨域
      timeout: 15 # API 超时时间
      name: apiName # API 名称
      qualifier: $DEFAULT # API 关联的版本
    protocols:
      - http
      - https
    environment: test
```

3. 创建完成后，在根目录下执行 `scf deploy` 进行部署，组件会根据选择的框架类型，自动生成 `scf_bootstrap` 启动文件进行部署。

> **注意**
> 由于启动文件逻辑与用户业务逻辑强关联，默认生成的启动文件可能导致框架无法正常启动，建议您根据实际业务需求，手动配置启动文件，详情参考各框架的部署指引文档。

示例 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

/**
 * docker 中 node 路径：/var/lang/node12/bin/node
 * 由于 serverless 函数只有 /tmp 读写权限，所以在启动时需要修改两个环境变量
 * NODE_LOG_DIR 是为了改写 egg-scripts 默认 node 写入路径（~/logs）-> /tmp
 * EGG_APP_CONFIG 是为了修改 egg 应有的默认当前目录 -> /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');

// 如果通过层部署 node_modules 就需要修改 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

/*
# HTTP 直通函数由于是基于 docker 镜像运行，所以必须监听地址为 0.0.0.0，并且端口为 9000
*/
const { nextStart } = require('next/dist/cli/next-start');
nextStart(['--port', '9000', '--hostname', '0.0.0.0']);
```

- nuxtjs

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

/*
# HTTP 直通函数由于是基于 docker 镜像运行，所以必须监听地址为 0.0.0.0，并且端口为 9000
*/
require('@nuxt/cli')
  .run(['start', '--port', '9000', '--hostname', '0.0.0.0'])
  .catch((error) => {
    require('consola').fatal(error);
    require('exit')(2);
  });
```

- 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

# HTTP 直通函数由于是基于 docker 镜像运行，所以必须监听地址为 0.0.0.0，并且端口为 9000
/var/lang/python3/bin/python3 app.py
```

- django

```
#!/bin/bash

# HTTP 直通函数由于是基于 docker 镜像运行，所以必须监听地址为 0.0.0.0，并且端口为 9000
/var/lang/python3/bin/python3 manage.py runserver 0.0.0.0:9000
```

- laravel

```
#!/bin/bash

#######################################
# 注入 serverless 环境下的环境变量
#######################################
# 注入 SERVERLESS 标识
export SERVERLESS=1
# 修改模板编译缓存路径，云函数只有 /tmp 目录可读写
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# 修改 session 以内存方式（数组类型）存储
export SESSION_DRIVER=array
# 日志输出到 stderr
export LOG_CHANNEL=stderr
# 修改应用存储路径
export APP_STORAGE=/tmp/storage

# 初始化模板缓存目录
mkdir -p /tmp/storage/framework/views

# HTTP 直通函数由于是基于 docker 镜像运行，所以必须监听地址为 0.0.0.0，并且端口为 9000
# 云端可执行文件路径 /var/lang/php7/bin/php
/var/lang/php7/bin/php artisan serve --host 0.0.0.0 --port 9000
```

快速部署 Egg 框架

最近更新时间：2022-12-14 17:04:15

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Egg 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Egg 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `Egg` 筛选函数模板，选择**Egg 框架模板**并单击**下一步**。如下图所示：

4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。

6. 单击左侧菜单栏中的**触发管理**，查看访问路径。您可以通过 API 网关生成的访问路径 URL，访问您部署的 Egg 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Egg 项目。如下图所示：

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

1. 参考 Egg.js 官方文档，快速初始化示例项目。示例如下：

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

2. 在根目录下，执行以下命令在本地直接启动服务。

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

3. 打开浏览器访问 `http://localhost:7001` ，即可在本地完成 Egg 示例项目的访问。

部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下三步：

- 修改监听地址与端口为 `0.0.0.0:9000` 。

- 修改写入路径，serverless 环境下只有 `/tmp` 目录可读写。
- 新增 `scf_bootstrap` 启动文件。

具体步骤如下：

1. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于配置环境变量和启动服务，此处仅为示例，具体操作请以您实际业务场景进行调整）：

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

'use strict';

/**
 * docker 中 node 路径：/var/lang/node12/bin/node
 * 由于 serverless 函数只有 /tmp 读写权限，所以在启动时需要修改两个环境变量
 * NODE_LOG_DIR 是为了改写 egg-scripts 默认 node 写入路径（~/logs）-> /tmp
 * EGG_APP_CONFIG 是为了修改 egg 应有的默认当前目录 -> /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');

// 如果通过层部署 node_modules 就需要修改 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. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

3. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
4. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。

5. 选择**从头开始**新建函数，根据页面提示配置相关选项。如下图所示：

- **函数类型**：选择 “Web 函数”。
- **函数名称**：填写您自己的函数名称。
- **地域**：填写您的函数部署地域，默认为广州。
- **运行环境**：选择 “Nodejs 12.16”。
- **提交方法**：选择“本地上传文件夹”。
- **函数代码**：选择函数代码在本地的具体文件夹。

6. 单击**完成**完成项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。

快速部署 Express 框架

最近更新时间：2022-12-15 17:12:31

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Express 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Express 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `WebFunc` 筛选所有 Web 函数模板，选择**Express 框架模板**并单击**下一步**。如下图所示：

4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。

6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Express 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Express 项目。如下图所示：

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

1. 执行以下命令安装 Express 框架和 express-generator 脚手架，并初始化 Express 示例项目。

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

2. 执行以下命令，进入项目目录并安装依赖包。

```
cd WebApp
npm install
```

3. 安装完成后，执行以下命令在本地直接启动服务。

```
npm start
```

4. 打开浏览器访问 `http://localhost:3000` ，即可在本地完成 Express 示例项目的访问。

部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下两步：

- 修改监听地址与端口为 `0.0.0.0:9000` 。
- 新增 `scf_bootstrap` 启动文件。

具体步骤如下：

1. 在 Express 示例项目中，可通过 `./bin/www` 设置监听地址与端口，打开该文件可以发现，通过环境变量可以设置指定监听端口，否则将自动监听3000端口。如下图所示：

```
/**
 * 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. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于配置环境变量和启动服务）：

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

3. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

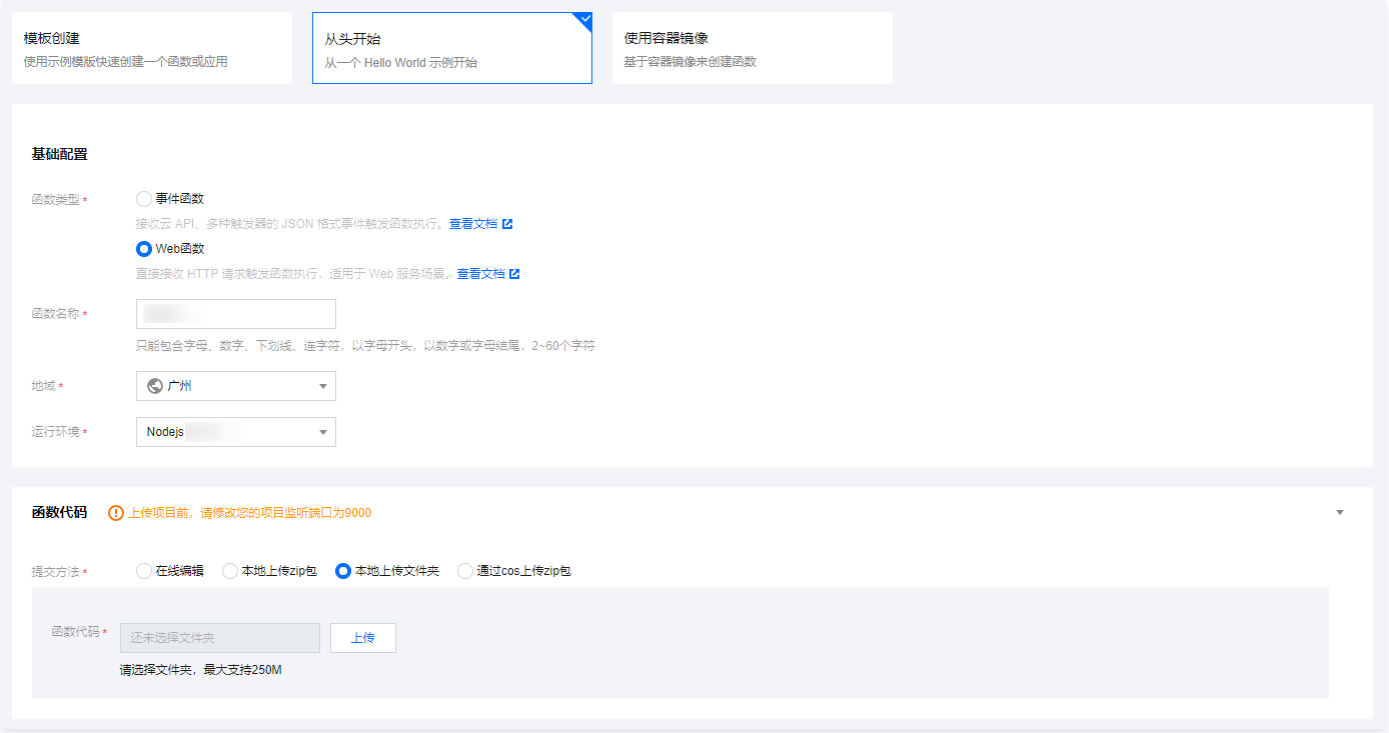
```
chmod 777 scf_bootstrap
```

```
```

4. 本地配置完成后，执行以下命令启动文件（如下命令为在该文件 `scf_bootstrap` 目录下执行时示例），确保您的服务在本地可以正常启动。

```
./scf_bootstrap
```

5. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
6. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
7. 选择**从头开始**新建函数，根据页面提示配置相关选项。如下图所示：



- **函数类型**：选择 “Web 函数”。
- **函数名称**：填写您自己的函数名称。
- **地域**：填写您的函数部署地域，默认为广州。
- **运行环境**：选择 “Nodejs 12.16”。
- **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
- **函数代码**：选择函数代码在本地的具体文件夹。

8. 单击**完成**完成 Express 项目的部署。

开发管理

部署完成后，即可在 Serverless 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。如下图所示：

访问路径 https://service- cd.apigw.tencentcs.com/release/

测试

测试模板

请求方式 GET

path /

headers

key	value
请输入key	请输入value

params

key	value
请输入key	请输入value

返回结果 说明文档

返回码 200

响应延时 780ms

响应Body

```
<!DOCTYPE html><html><head>
<title>Express</title>
<link rel="stylesheet" href="/stylesheets/style.css">
</head><body><h1>Express</h1>
<p>Welcome to Express</p></body></html>
```

响应Headers

```
X-Powered-By:Express
Content-Type:text/html; charset=utf-8
X-Api-
Requestid:ddddf292c153113e4d5425
Cache-Control:max-age=3600
Expires:Tue, 15 Jun 2021 02:21:30 GMT
Vary:Accept-Encoding
Content-Length:170
Connection:keep-alive
Etag:W/"aa-z+ebXSEdArbZ+EXIN/W
```

快速部署 Flask 框架

最近更新时间：2023-05-22 16:56:09

操作场景

本文将为您指导如何通过 SCF Web Function，快速部署您的 Flask 业务上云。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

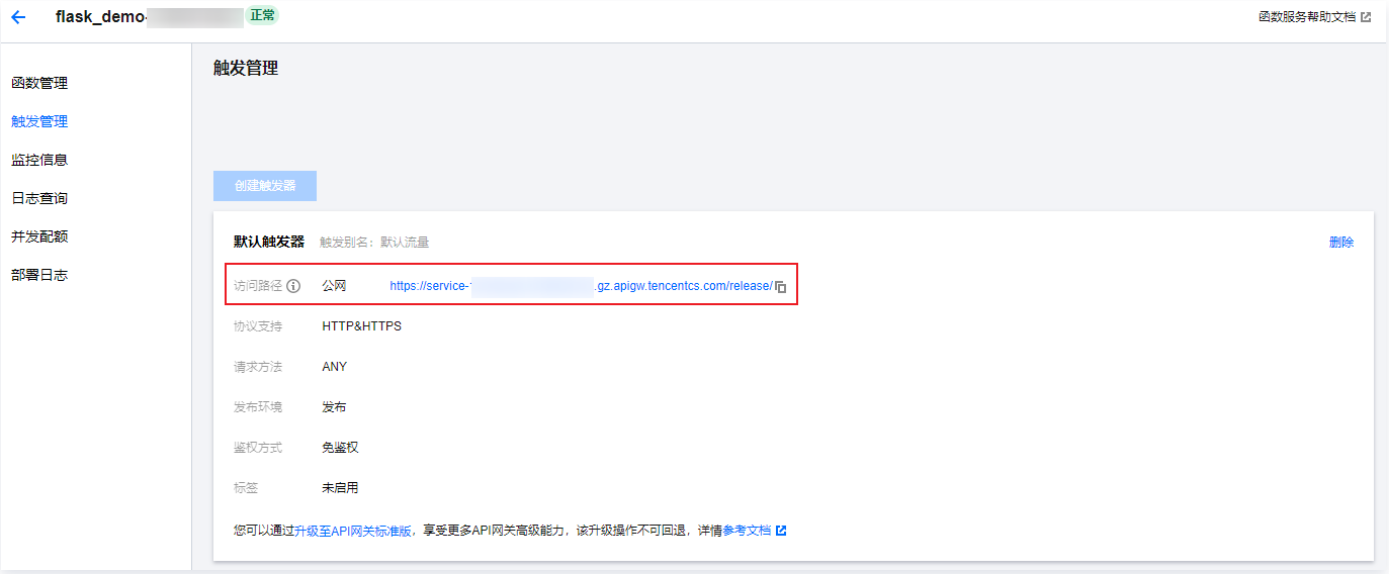
操作步骤

模板部署：一键部署 Flask 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `WebFunc` 筛选所有 Web 函数模板，选择**Flask 框架模板**并单击**下一步**。如下图所示：

4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。

6. 您可以通过自动生成的访问路径 URL，访问您部署的 Flask 项目。单击左侧菜单栏中的**触发管理**，查看访问路径。如下图所示：



7. 单击访问路径 URL，即可访问服务 Flask 项目。如下图所示：

自定义部署：快速迁移本地项目上云

本地开发

1. 执行以下命令，确认您本地的环境已安装好 Flask。

```
pip install Flask
```

2. 在本地创建 `Hello World` 示例项目。
   在项目目录下，新建 `app.py` 文件，用于实现 Hello World 应用，示例代码如下：

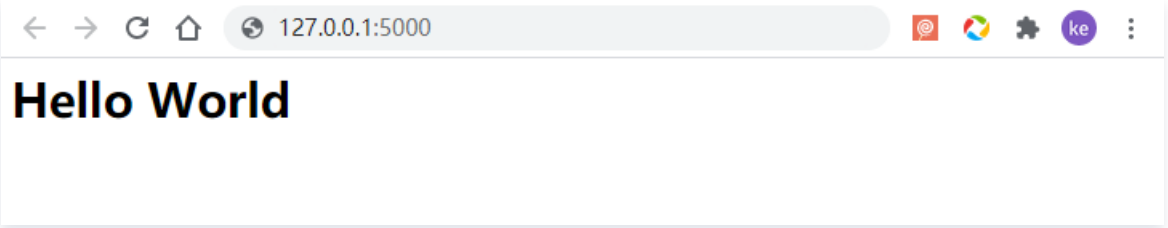
```
from flask import Flask
app = Flask(__name__)

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

if __name__ == '__main__':
```

```
    app.run()
```

3. 在本地执行 `python3 app.py` 命令运行 app.py 文件。示例如下：

```
$ 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 -
```

4. 打开浏览器访问 `http://127.0.0.1:5000` ，即可在本地完成 Flask 示例项目的访问。如下图所示：



部署上云

接下来执行以下步骤，对本地已创建完成的项目进行简单修改，使其可以通过 Web Function 快速部署，对于 Flask，具体修改步骤如下：

1. **安装依赖包**

   1.1 由于 SCF 云上标准环境内未提供 Flask 依赖库，此处您必须将依赖文件安装完成后，与项目代码一起打包上传。请先新建 `requirements.txt` 文件，文件内容如下：

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

> **注意**
> 由于 SCF 内置运行环境版本 (Python 3.6) 限制，werkzeug 只能使用低版本(<=1.0.x)，高版本可能无法正常运行，函数运行环境版本升级已在规划中，敬请期待。

   1.2 执行以下命令进行安装：

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

2. **修改监听地址与端口**

   在 Web 函数内，限制了监听端口必须为**9000**，因此需要修改监听地址端口为 `0.0.0.0:9000` ，如下图所示：

```python
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)
```

> **说明**
> 您也可以在 `scf_bootstrap` 中，通过环境变量配置监听端口。

3. **新增 `scf_bootstrap` 启动文件**

   3.1 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于完成环境变量配置，指定服务启动命令等自定义操作，确保您的服务可以通过该文件正常启动）：

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

   3.2 创建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可以正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

> **注意**
> - 在 SCF 环境内，只有 `/tmp` 文件可读写，建议输出文件时选择 `/tmp`，其他目录会由于缺少权限而写入失败。
> - 如需在日志中输出环境变量，需在启动命令前加 `-u` 参数，例如 `python -u app.py`。

4. 本地配置完成后，执行以下命令启动服务（如下命令为在 scf_bootstrap 目录下执行时示例），确保您的服务在本地可以正常启动。

```
./scf_bootstrap
```

5. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
6. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。

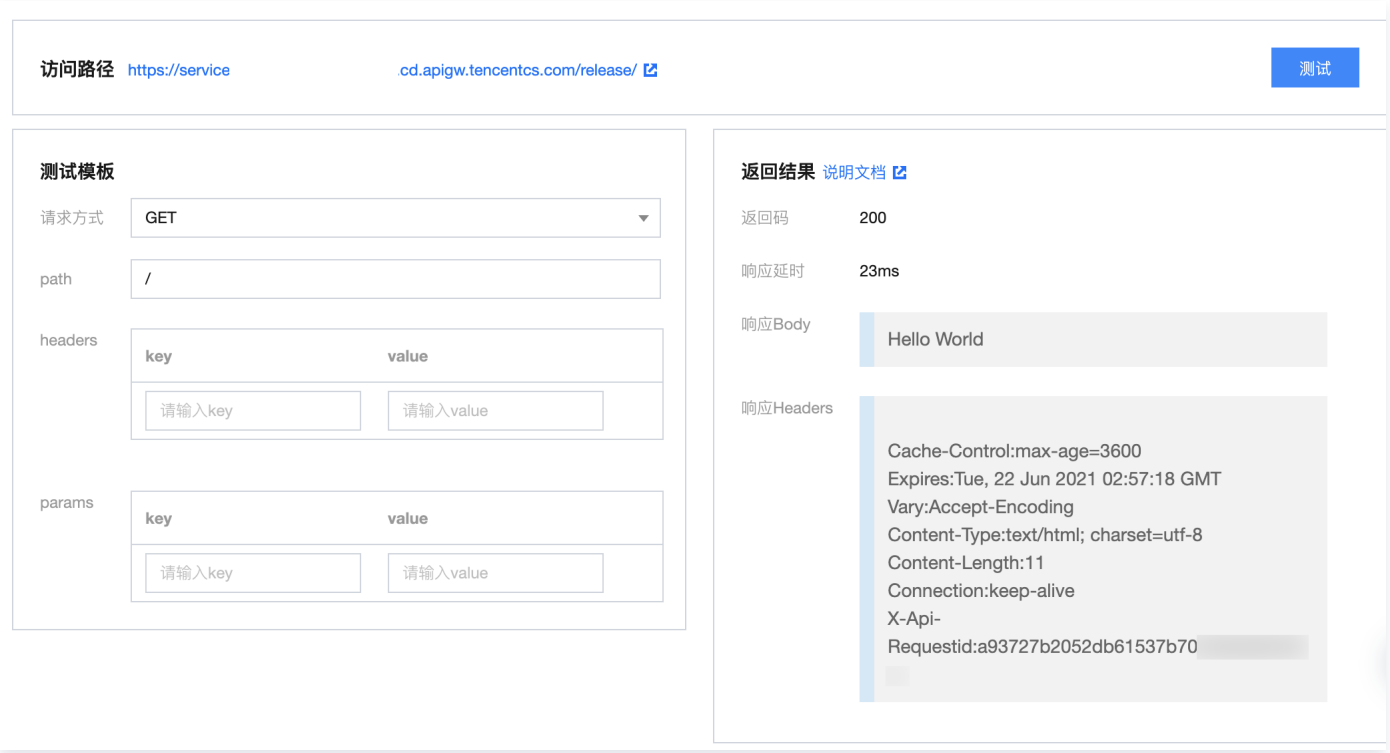
7. 选择**从头开始**新建函数，根据页面提示配置相关选项。如下图所示：

- **函数类型**：选择 “Web 函数”。
- **函数名称**：填写您自己的函数名称。
- **地域**：填写您的函数部署地域，例如成都。
- **部署方式**：选择“代码部署”，上传您的本地项目。
- **运行环境**：选择 “Python3.6”。

8. 单击**完成**完成 Flask 项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。如下图所示：


访问路径 https://service cd.apigw.tencentcs.com/release/
测试
测试模板
请求方式
GET
path
/
headers
key
value
请输入key
请输入value
params
key
value
请输入key
请输入value
返回结果 说明文档
返回码
200
响应延时
23ms
响应Body
Hello World
响应Headers
Cache-Control:max-age=3600
Expires:Tue, 22 Jun 2021 02:57:18 GMT
Vary:Accept-Encoding
Content-Type:text/html; charset=utf-8
Content-Length:11
Connection:keep-alive
X-Api-
Requestid:a93727b2052db61537b70

快速部署 Koa 框架

最近更新时间：2022-12-16 15:40:51

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Koa 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Koa 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `koa` 筛选函数模板，选择**Koa 框架模板**并单击**下一步**。
4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。
6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Koa 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Koa 项目。如下图所示：

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

1. 参考 Koa.js 官方文档，安装 Koa 环境并初始化您的 Koa 项目，此处以 `hello world` 为例， `app.js` 内容如下：

```
// 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. 在根目录下，执行以下命令在本地直接启动服务。

```
node app.js
```

3. 打开浏览器访问 `http://localhost:3000` ，即可在本地完成 Koa 示例项目的访问。

部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下两步：

- 修改监听地址与端口为 `0.0.0.0:9000` 。
- 新增 `scf_bootstrap` 启动文件。

具体步骤如下：

1. 在 Koa 示例项目中，修改监听端口到 `9000` 。如下图所示：

```
1  const Koa = require('koa');
2  const app = new Koa();
3
4  app.use(async ctx => {
5    ctx.body = 'Hello World';
6  });
7
8  app.listen(9000);
9
```

2. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于配置环境变量和启动服务）：

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

3. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

4. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
5. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
6. 选择**从头开始**新建函数，根据页面提示配置相关选项。
   - **函数类型**：选择 “Web 函数”。
   - **函数名称**：填写您自己的函数名称。
   - **地域**：填写您的函数部署地域，默认为广州。
   - **运行环境**：选择 “Nodejs 12.16”。
   - **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
   - **函数代码**：选择函数代码在本地的具体文件夹。
7. 单击**完成**完成 Koa 项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。

快速部署 Laravel 框架

最近更新时间：2023-09-06 11:18:12

操作场景

本文档指导您如何通过 Web 函数，快速迁移本地的 Laravel 服务上云。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Laravel 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里筛选 `WebFunc` ，筛选所有 Web 函数模板，选择**Laravel 框架模板**并单击**下一步**。
4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。
6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Laravel 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Laravel 项目。如下图所示：

自定义部署：快速迁移本地项目上云

本地开发

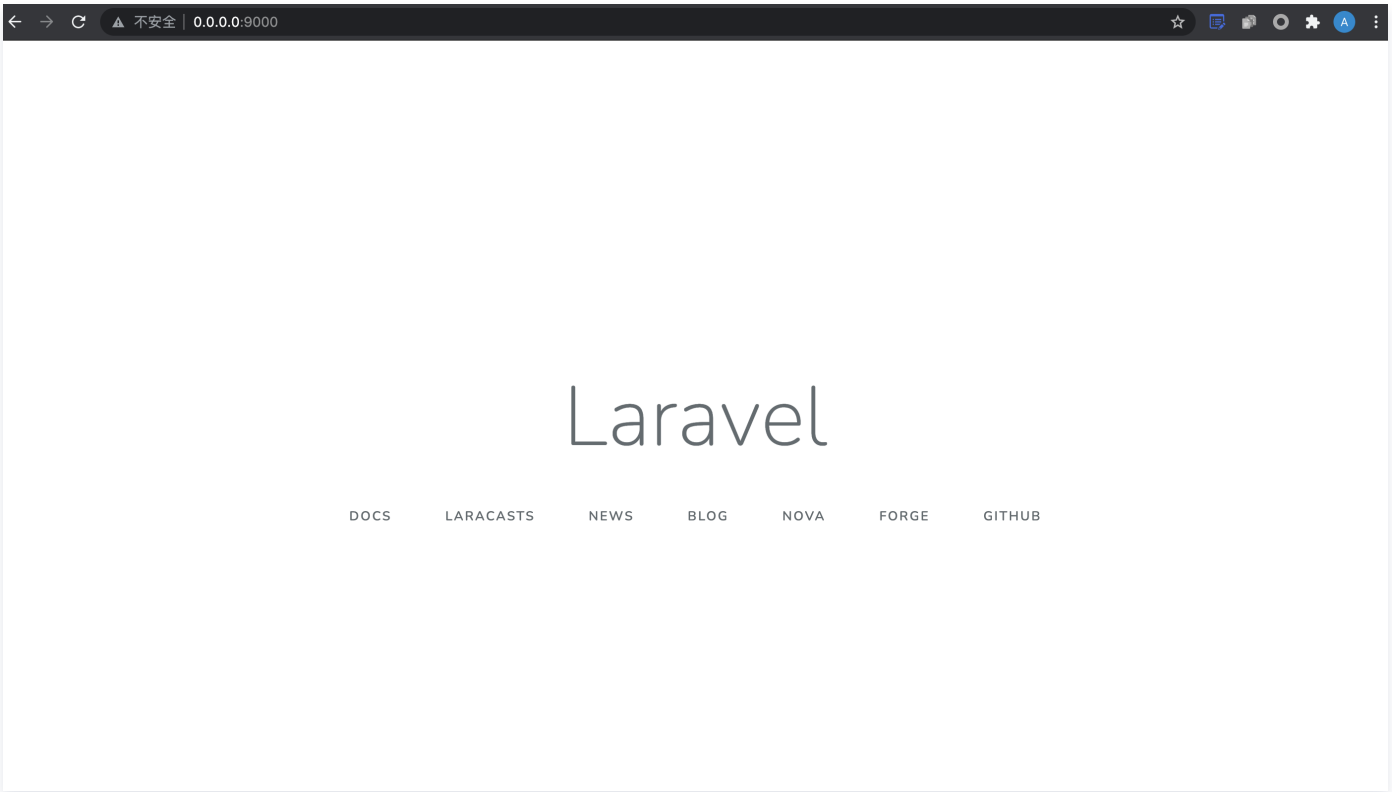
1. 参考 Laravel 官方文档，在本地环境中完成 Laravel 的开发环境搭建。
2. 在本地创建 Laravel 示例项目。进入项目目录下，执行以下命令，初始化 Laravel 示例应用：

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

3. 执行以下命令，在本地启动示例项目。示例如下：

```
$ 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. 打开浏览器访问 `http://0.0.0.0:9000` ，即可在本地完成 Laravel 示例项目的访问。如下图所示：



部署上云

执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，具体修改步骤如下：

1. **新增 `scf_bootstrap` 启动文件**
   在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件完成环境变量配置，指定服务启动命令等自定义操作，确保您的服务可以通过该文件正常启动。

   > **注意**
   > - `scf_bootstrap` 必须有 `755` 或者 `777` 的可执行权限。
   > - 如需在日志中输出环境变量，需在启动命令前需要加 `-u` 参数，例如 `python -u app.py` 。

2. **修改文件读写路径**
   由于在 SCF 环境内，只有 `/tmp` 文件可读写，其他目录会由于缺少权限而写入失败，因此需要在 `scf_bootstrap` 里，以环境变量的方式注入，调整 Laravel 框架的输出目录：

```
#!/bin/bash

# 注入 SERVERLESS 标识
export SERVERLESS=1
# 修改模板编译缓存路径，云函数只有 /tmp 目录可读写
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# 修改 session 以内存方式（数组类型）存储
export SESSION_DRIVER=array
# 日志输出到 stderr
export LOG_CHANNEL=stderr
# 修改应用存储路径
export APP_STORAGE=/tmp/storage

# 初始化模板缓存目录
mkdir -p /tmp/storage/framework/views
```

3. **修改监听地址与端口**

   在 Web 函数内，限制了监听端口必须为 `9000`，因此需要在 `scf_bootstrap` 中通过以下命令指定监听端口：

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

完整 `scf_bootstrap` 内容如下：

```
#scf_bootstrap
#!/bin/bash

##############################
# 注入 serverless 环境下的环境变量
##############################
# 注入 SERVERLESS 标识
export SERVERLESS=1
# 修改模板编译缓存路径，云函数只有 /tmp 目录可读写
export VIEW_COMPILED_PATH=/tmp/storage/framework/views
# 修改 session 以内存方式（数组类型）存储
export SESSION_DRIVER=array
# 日志输出到 stderr
export LOG_CHANNEL=stderr
# 修改应用存储路径
export APP_STORAGE=/tmp/storage

# 初始化模板缓存目录
mkdir -p /tmp/storage/framework/views

# HTTP函数是基于 docker 镜像运行，所以监听地址必须为0.0.0.0，端口为9000
# 云端可执行文件路径/var/lang/php7/bin/php
/var/lang/php7/bin/php artisan serve --host 0.0.0.0 --port 9000
```

> **注意：**
> 示例运行环境为 Php7.2，如使用其他语言版本环境，详情请参见 标准语言环境绝对路径 修改 `scf_bootstrap` 中的云端可执行文件路径。

4. **部署 Laravel**

本地配置完成后，执行启动文件，确保您的服务可以本地正常启动。执行以下步骤部署 Laravel：

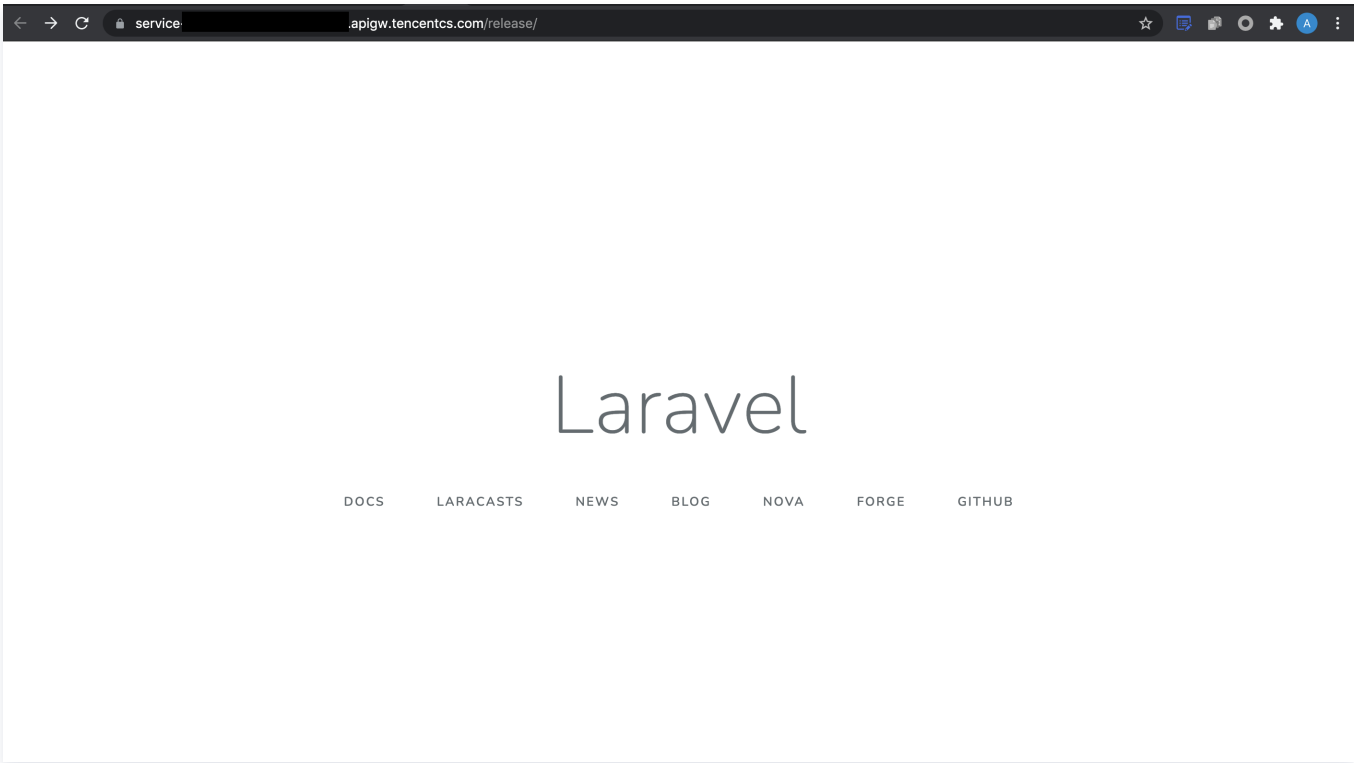
4.1 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。

4.2 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。

4.3 选择**从头开始**新建函数，根据页面提示配置相关选项。

- **函数类型**：选择 “Web 函数”。
- **函数名称**：填写您自己的函数名称。
- **地域**：填写您的函数部署地域，例如成都。
- **运行环境**：选择 “Php7.2”。
- **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
- **函数代码**：选择函数代码在本地的具体文件夹。

4.4 部署完成后，单击生成的 URL，即可访问您的 Laravel 应用。如下图所示：



开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。

快速部署 Nestjs 框架

最近更新时间：2023-12-12 14:30:22

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Nest.js 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Nest.js 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `nest` 筛选函数模板，选择**Nest框架模板**并单击**下一步**。
4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。
6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Nest.js 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Nest.js 项目。如下图所示：

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

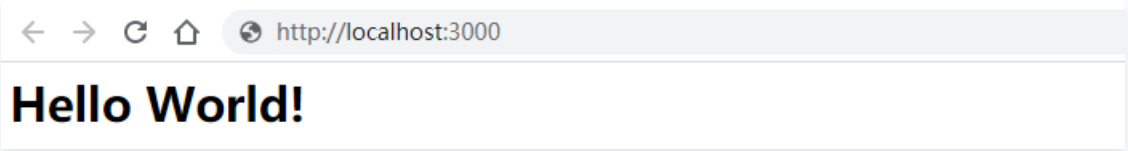
1. 参考 Nest.js 官方文档，初始化您的 Nest.js 项目：

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

2. 在根目录下，执行以下命令在本地直接启动服务。

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

3. 打开浏览器访问 `http://localhost:3000` ，即可在本地完成 Nest.js 示例项目的访问。如下图所示：



部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下两步：

- 新增 `scf_bootstrap` 启动文件。
- 修改监听地址与端口为 `0.0.0.0:9000` 。

具体步骤如下：

1. 修改启动文件 `./dist/main.js` ，监听端口改为 `9000` 。如下图所示：

```
"use strict";
Object.defineProperty(exports, "__esModule", { value:
const core_1 = require("@nestjs/core");
const app_module_1 = require("./app.module");
async function bootstrap() {
    const app = await core_1.NestFactory.create(app_mo
    await app.listen(9000);
}
bootstrap();
//# sourceMappingURL=main.js.map
```

2. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于启动服务）：

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

```
```

> **注意**
> - 此处仅为示例启动文件，具体请根据您的业务场景进行调整。
> - 示例使用的是云函数标准 Node 环境路径，本地调试时，需修改成您的本地路径。

3. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

4. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
5. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
6. 选择**从头开始**新建函数，根据页面提示配置相关选项。
   - **函数类型**：选择 “Web 函数”。
   - **函数名称**：填写您自己的函数名称。
   - **地域**：填写您的函数部署地域，默认为广州。
   - **运行环境**：选择 “Nodejs 12.16”。
   - **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
   - **函数代码**：选择函数代码在本地的具体文件夹。
7. 单击**完成**完成 Nest.js 项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。

快速部署 Nextjs 框架

最近更新时间：2022-12-16 15:22:05

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Next.js SSR 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Next.js 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `webfunc` 筛选函数模板，选择**Next.js 框架模板**并单击**下一步**。
4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。
6. 单击左侧菜单栏中的触发管理，查看访问路径 URL，访问您部署的 Next.js 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Next.js 项目。如下图所示：

> **说明**
> 由于 Nextjs 框架每次部署前需要重新构建，请确保本地更新代码并且重新 `build` 之后再进行部署。

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

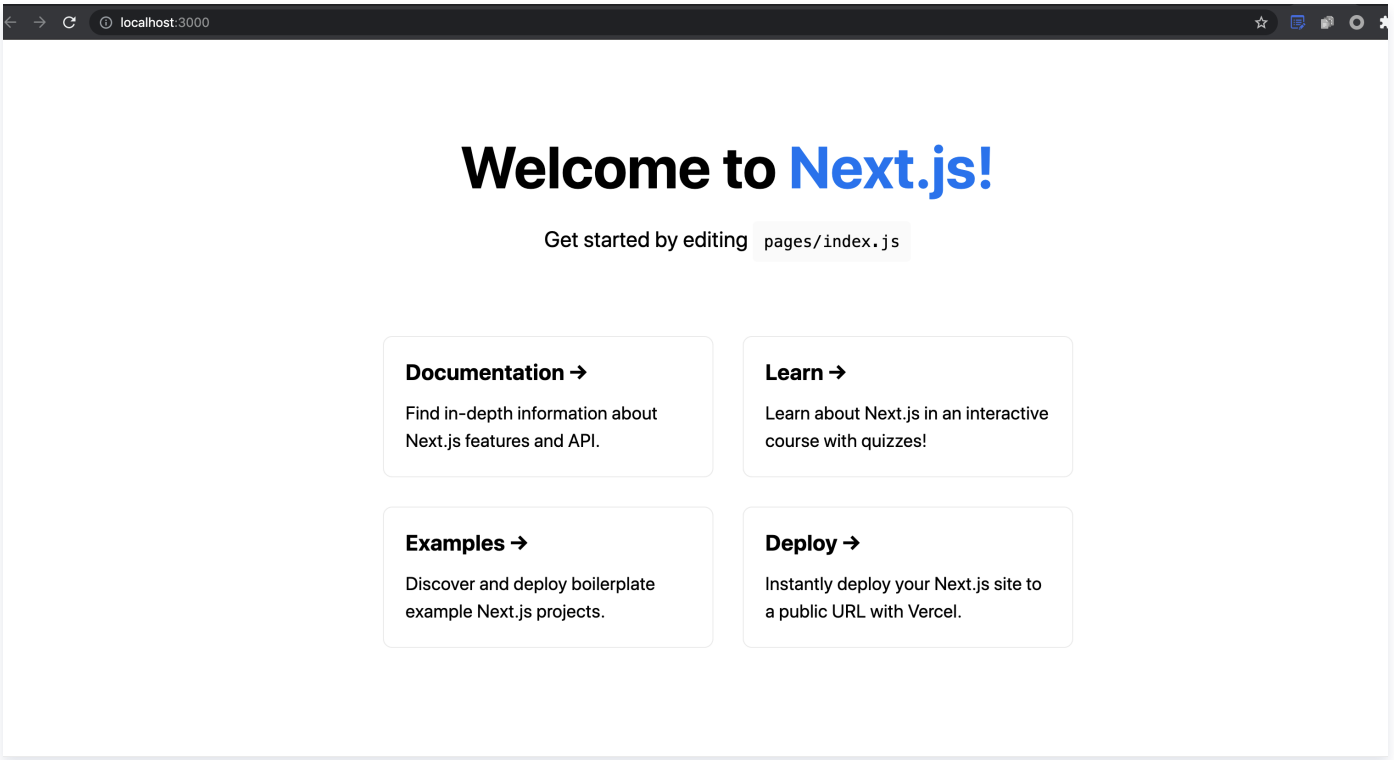
1. 参考 Next.js 官方文档，安装并初始化您的 Next.js 项目：

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

2. 在根目录下，执行以下命令在本地直接启动服务。

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

3. 打开浏览器访问 `http://localhost:3000` ，即可在本地完成 Next.js 示例项目的访问。如下图所示：



部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下两步：

- 修改监听地址与端口为 `0.0.0.0:9000` 。
- 新增 `scf_bootstrap` 启动文件。

具体步骤如下：

1. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于启动服务并指定启动端口）：

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

```
const { nextStart } = require('next/dist/cli/next-start');
nextStart([ '--port', '9000', '--hostname', '0.0.0.0' ])
```

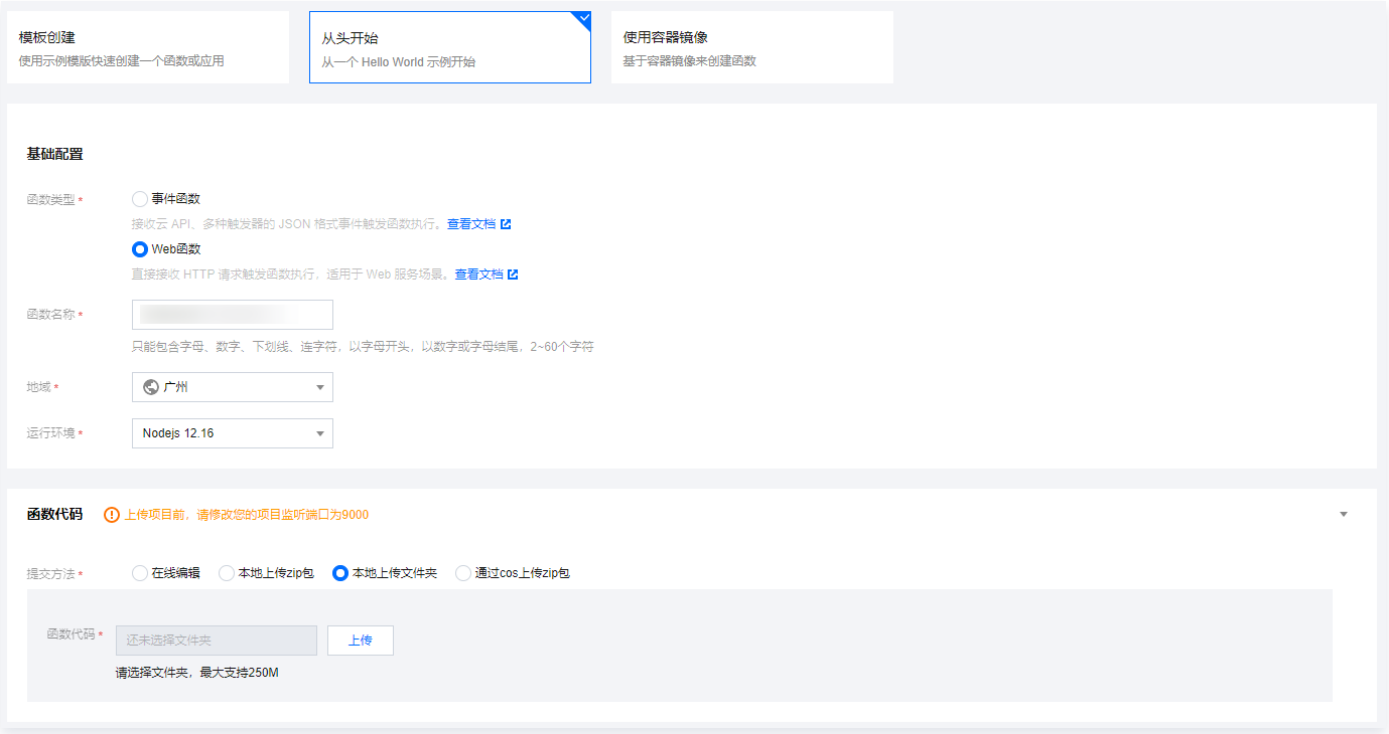
注意

- 此处仅为示例启动文件，具体请根据您的业务场景进行调整。
- 示例使用的是云函数标准 Node 环境路径，本地调试时，需修改成您的本地路径。

2. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

3. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
4. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
5. 选择**从头开始**新建函数，根据页面提示配置相关选项。如下图所示：

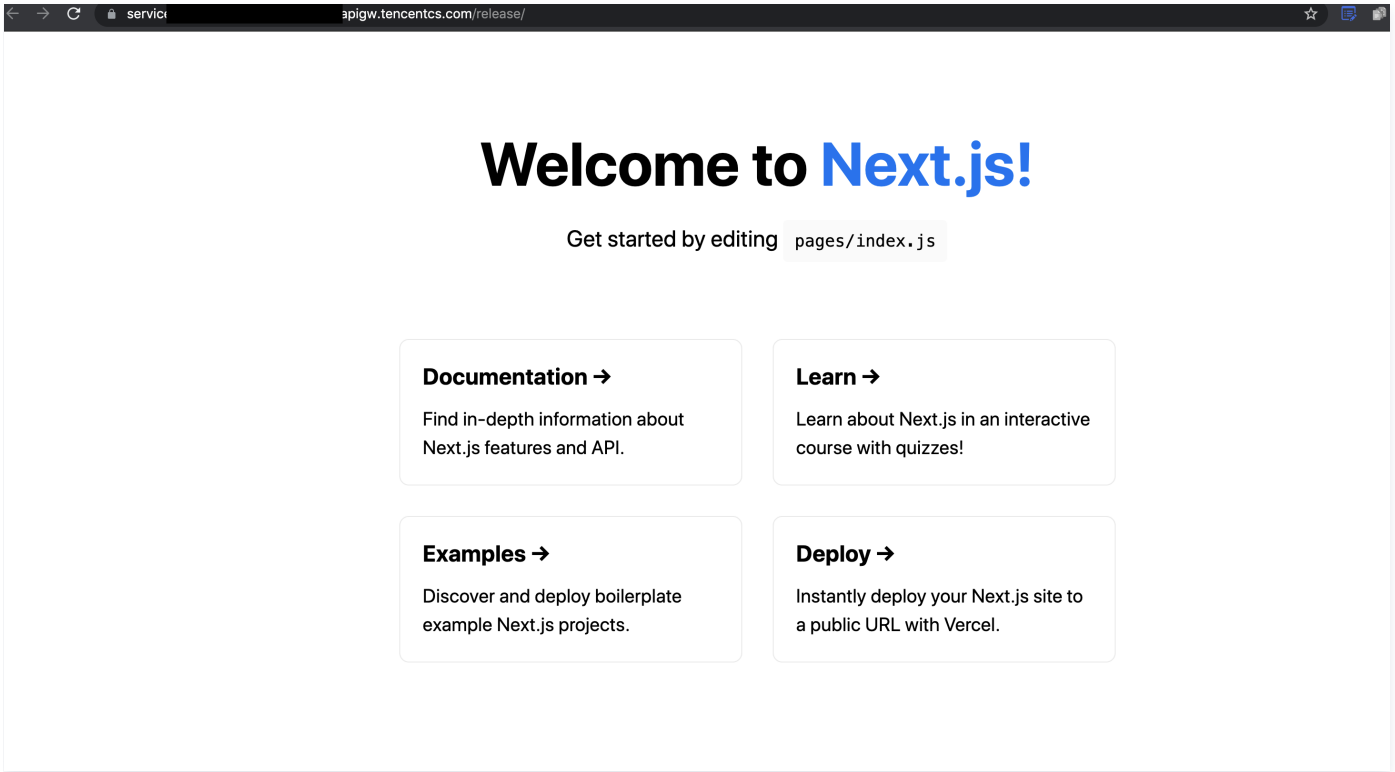


- **函数类型**：选择 “Web 函数”。
- **函数名称**：填写您自己的函数名称。
- **地域**：填写您的函数部署地域，默认为广州。
- **运行环境**：选择 “Nodejs 12.16”。
- **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
- **函数代码**：选择函数代码在本地的具体文件夹。

6. 单击**完成**完成 Next.js 项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。


servic
apigw.tencentcs.com/release/
Welcome to Next.js!
Get started by editing pages/index.js
Documentation →
Find in-depth information about Next.js features and API.
Learn →
Learn about Next.js in an interactive course with quizzes!
Examples →
Discover and deploy boilerplate example Next.js projects.
Deploy →
Instantly deploy your Next.js site to a public URL with Vercel.

快速部署 Nuxtjs 框架

最近更新时间：2022-12-16 15:10:27

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Nuxt.js SSR 项目快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Nuxt.js 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `webfunc` 筛选函数模板，选择**Nuxt.js 框架模板**并单击**下一步**。
4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。
6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Nuxt.js 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Nuxt.js 项目。如下图所示：

> **说明**
> 由于 Nuxtjs 框架每次部署前需要重新构建，请确保本地更新代码并且重新 `build` 之后再进行部署。

自定义部署：快速迁移本地项目上云

前提条件

本地已安装 Node.js 运行环境。

本地开发

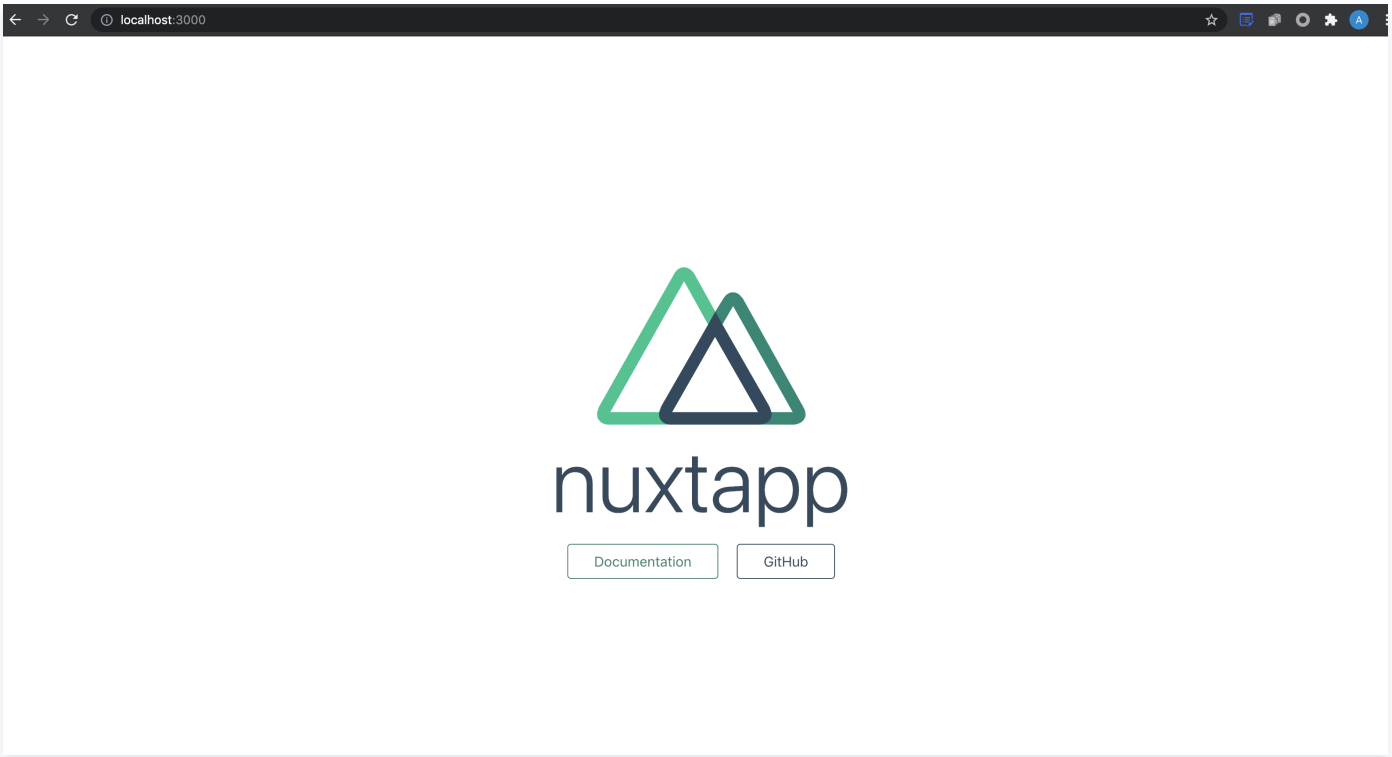
1. 参考 Nuxt.js 官方文档，安装并初始化您的 Nuxt.js 项目：

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

2. 在根目录下，执行以下命令在本地直接启动服务。

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

3. 打开浏览器访问 `http://localhost:3000` ，即可在本地完成 Nuxt.js 示例项目的访问。如下图所示：



部署上云

接下来执行以下步骤，对已初始化的项目进行简单修改，使其可以通过 Web Function 快速部署，此处项目改造通常分为以下两步：

- 新增 `scf_bootstrap` 启动文件。
- 修改监听地址与端口为 `0.0.0.0:9000` 。

具体步骤如下：

1. 在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于启动服务并指定启动端口）：

```
#!/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);
  });
```

> **注意**
> - 此处仅为示例启动文件，具体请根据您的业务场景进行调整。
> - 示例使用的是云函数标准 node 环境路径，本地调试时，需修改成您的本地路径。

2. 新建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

3. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。

4. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
5. 选择**从头开始**新建函数，根据页面提示配置相关选项。
   - **函数类型**：选择 “Web 函数”。
   - **函数名称**：填写您自己的函数名称。
   - **地域**：填写您的函数部署地域，默认为广州。
   - **运行环境**：选择 “Nodejs 12.16”。
   - **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
   - **函数代码**：选择函数代码在本地的具体文件夹。
6. 单击**完成**完成 Nuxt.js 项目的部署。

> **注意**
> 访问 URL 时，可能由于前端路由导致访问失败，访问时需去掉 `/release` 路径。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势。

快速部署 Django 框架

最近更新时间：2022-12-16 15:09:14

操作场景

本文将为您指导如何通过 Web Function，将您的本地 Django 快速部署到云端。

> **说明**
> 本文档主要介绍控制台部署方案，您也可以通过命令行完成部署，详情请参见 通过命令行完成框架部署 。

前提条件

在使用腾讯云云函数服务之前，您需要 注册腾讯云账号 并完成 实名认证 。

操作步骤

模板部署：一键部署 Django 项目

1. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
2. 在主界面上方选择期望创建函数的地域和命名空间，并单击**新建**，进入函数创建流程。
3. 选择使用**模板创建**来新建函数，在搜索框里输入 `Django` 选择 **Django 框架模板**并单击**下一步**。如下图所示：

4. 在**新建**页面，您可以查看模板项目的具体配置信息并进行修改。
5. 单击**完成**即可创建函数。函数创建完成后，您可在**函数管理**页面查看 Web 函数的基本信息。

6. 单击左侧菜单栏中的**触发管理**，查看访问路径 URL，访问您部署的 Django 项目。如下图所示：

7. 单击访问路径 URL，即可访问服务 Django 项目。如下图所示：

自定义部署：快速迁移本地项目上云

本地开发

1. 执行以下命令，确认您本地的环境已安装好 Django。

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

2. 在本地创建 `Hello World` 示例项目。

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

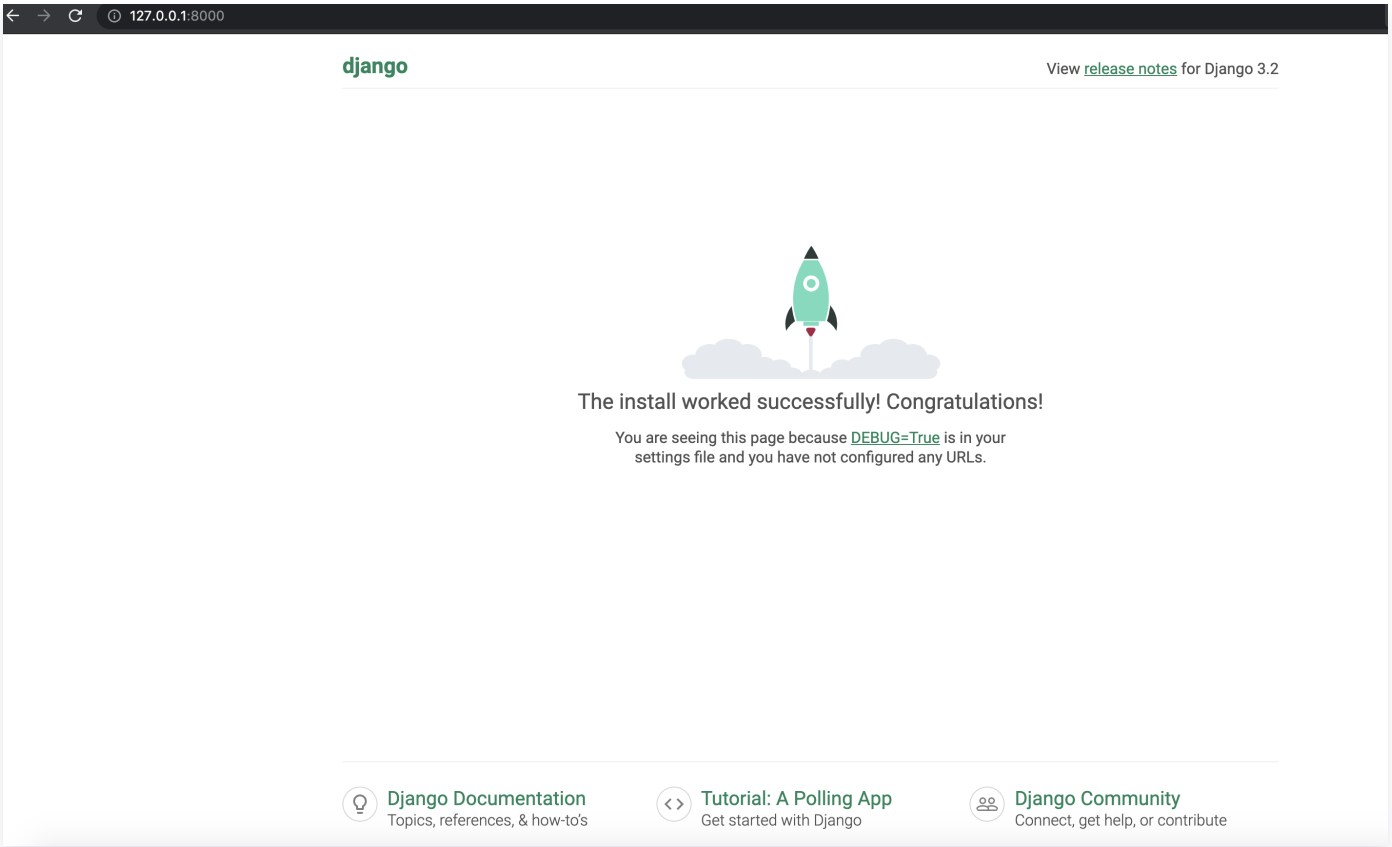
目录结构如下：

```
$ tree
. manage.py 管理器
|--***
|   |-- __init__.py 包
|   |-- settings.py  设置文件
|   |-- urls.py   路由
|   `-- wsgi.py   部署
```

3. 在本地执行 `python manage.py runserver` 命令运行启动文件。示例如下：

```
$ 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.
```

4. 打开浏览器访问 `http://127.0.0.1:8000` ，即可在本地完成 Django 示例项目的访问。如下图所示：



部署上云

接下来执行以下步骤，对本地已创建完成的项目进行简单修改，使其可以通过 Web Function 快速部署，对于 Django，具体修改步骤如下：

1. 安装依赖包

   1.1 由于 SCF 云上标准环境内未提供 Django 依赖库，此处您必须将依赖文件安装完成后，与项目代码一起打包上传。请先新建 `requirements.txt` 文件，文件内容如下：

```
Django==3.1.3
```

1.2 执行以下命令进行安装：

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

> **说明**
> 由于初始化的默认项目引用了 `db.sqlite3` 库，请同步安装该依赖，或将项目文件内 `setting.py` 里 `DATABASES` 字段部分配置注释。

2. **新增 `scf_bootstrap` 启动文件**
在 Web 函数内，限制了监听端口必须为9000，因此需要修改监听地址端口，在项目根目录下新建 `scf_bootstrap` 启动文件，在该文件添加如下内容（用于完成环境变量配置，指定服务启动命令等自定义操作，确保您的服务可以通过该文件正常启动）：

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

3. 创建完成后，还需执行以下命令修改文件可执行权限，默认需要 `777` 或 `755` 权限才可以正常启动。示例如下：

```
chmod 777 scf_bootstrap
```

> **注意**
> - 在 SCF 环境内，只有 `/tmp` 文件可读写，建议输出文件时选择 `/tmp`，其他目录会由于缺少权限而写入失败。
> - 如需在日志中输出环境变量，需在启动命令前加 `-u` 参数，例如 `python -u app.py`。

4. 本地配置完成后，执行以下命令启动服务（如下命令为在 scf_bootstrap 目录下执行时示例），确保您的服务在本地可以正常启动。

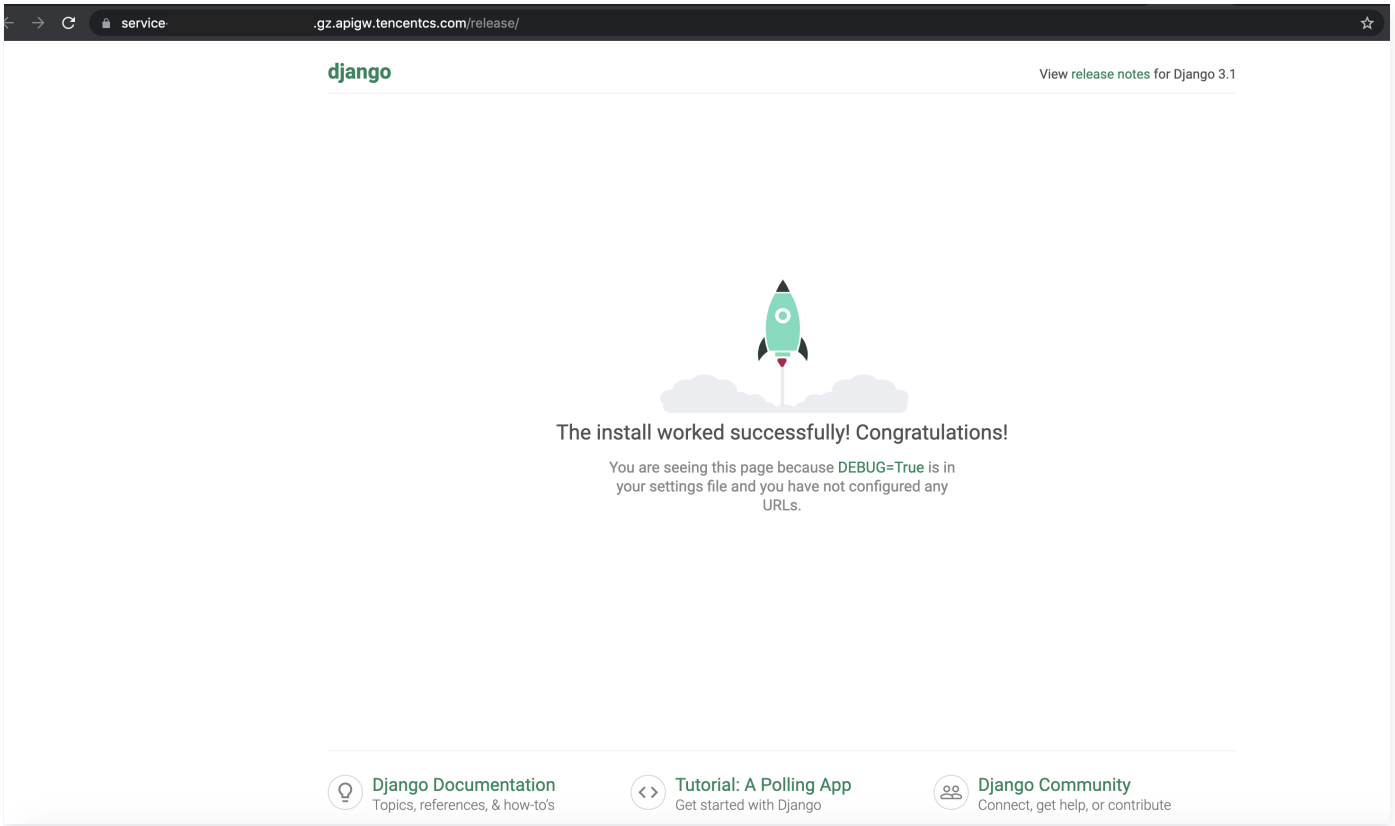
> **注意**
> 本地测试时注意将 python 路径改为本地路径。

```
./scf_bootstrap
```

5. 登录 Serverless 控制台，单击左侧导航栏的**函数服务**。
6. 在主界面上方选择期望创建函数的地域，并单击**新建**，进入函数创建流程。
7. 选择**从头开始**新建函数，根据页面提示配置相关选项。
   - **函数类型**：选择 “Web 函数”。
   - **函数名称**：填写您自己的函数名称。
   - **地域**：填写您的函数部署地域，例如成都。
   - **运行环境**：选择 “Python3.6”。
   - **提交方法**：选择“本地上传文件夹”，上传您的本地项目。
   - **函数代码**：选择函数代码在本地的具体文件夹。
8. 单击**完成**完成 Django 项目的部署。

开发管理

部署完成后，即可在 SCF 控制台快速访问并测试您的 Web 服务，并且体验云函数多项特色功能，例如层绑定、日志管理等，享受 Serverless 架构带来的低成本、弹性扩缩容等优势，如下图所示：


service
.gz.apigw.tencentcs.com/release/
django
View release notes for Django 3.1
The install worked successfully! Congratulations!
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute