Serverless 应用中心

框架支持

产品文档

腾讯云

文档目录

框架支持
通过命令行完成框架部署

最近更新时间：2021-08-20 16:21:26

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 Framework 的 HTTP 组件，完成 Web 应用的本地部署。

前提条件

已开通服务并完成 Serverless 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 `sls 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）-&gt; /tmp
* EGG_APP_CONFIG 是为了修改 egg 应有的默认当前目录 -&gt; /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', () =&gt; {
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) =&gt; {
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：2022-07-22 15:11:39

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function`，快速部署您的 Egg 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 Serverless控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

本篇文档为您介绍应用控制台的部署方案，您也可以通过命令行完成部署，具体操作请参考 产品文档。

模板部署 -- 部署 Egg 示例代码

1. 登录 Serverless 控制台。
2. 单击**新建应用**，选择**Web 应用>Egg 框架**。
3. 单击“下一步”，完成基础配置选择。
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Egg 项目。

自定义部署 -- 快速部署 Web 应用

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：

```
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，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 Egg 框架，具体改造说明如下：

- 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。

1. (可选)配置 scf_bootstrap 启动文件

> 说明：
> 您也可以在控制台完成该模块配置。

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来调整）：

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

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
```

2. 控制台上传

登录 Serverless 控制台，选择**Web 应用>Egg 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Egg 项目。

高级配置管理

您可在“高级配置”里进行更多应用管理操作，如创建层、绑定自定义域名、配置环境变量等。

快速部署 Koa 框架

最近更新时间：2022-07-22 15:09:05

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function` ，快速部署您的 Koa 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 Serverless控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

前提条件

- 在使用腾讯云Serverless 应用中心之前，您需要 注册腾讯云账号 并完成 实名认证。

> 注意：
> 本篇文档为您介绍应用控制台的部署方案，您也可以通过命令行完成部署，具体操作请参考 产品文档。

操作步骤

模板部署 -- 部署 Koa 示例代码

1. 登录 Serverless 控制台。
2. 单击**新建应用**，选择**Web 应用>Koa 框架**。
3. 单击“下一步”，完成基础配置选择。
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Koa 项目

自定义部署 -- 快速部署 Web 应用

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。

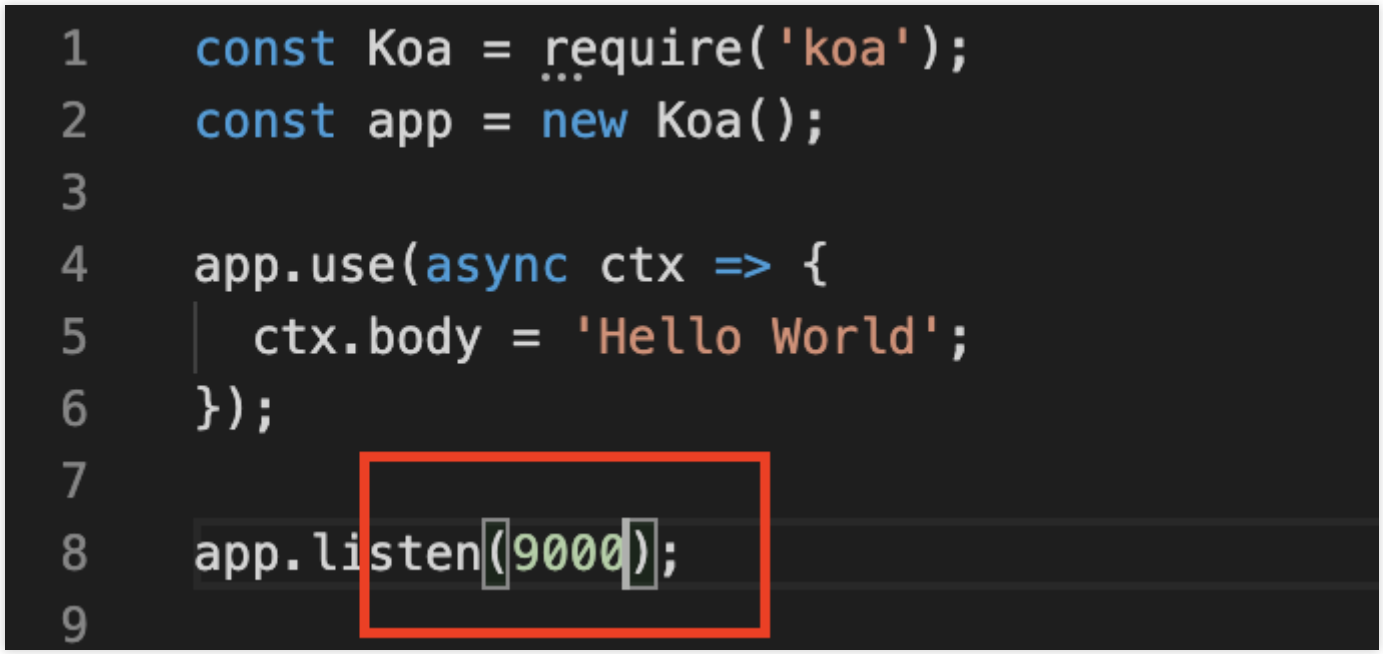
部署上云

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`



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）：

> 说明：
> 您也可以在控制台完成该模块配置。

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

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. 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，接下来，登录 Serverless 控制台，选择**Web 应用>Koa 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Koa 项目。

快速部署 Express 框架

最近更新时间：2023-05-05 15:03:36

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function`，快速部署您的 Express 业务上云。

注意

应用控制台部署与函数直接部署有什么区别？

通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

如果您只关注代码逻辑开发，无需额外资源创建，可以通过 Serverless 控制台，完成快速部署。

如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

本篇文档为您介绍应用控制台的部署方案，您也可以通过命令行完成部署，具体操作请参考 产品文档。

模板部署：部署 Express 示例代码

1. 登录 Serverless 控制台。
2. 单击**新建应用**，选择**Web 应用 > Express 框架。**
3. 单击“下一步”，完成基础配置选择。在上传方式中，选择**示例代码**直接部署。
4. 单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用列表页面，单击示例应用名称，进入应用详情页。
6. 在应用的资源列表页面，单击示例应用的 API 网关生成的访问路径 URL，查看您部署的 Express 项目。

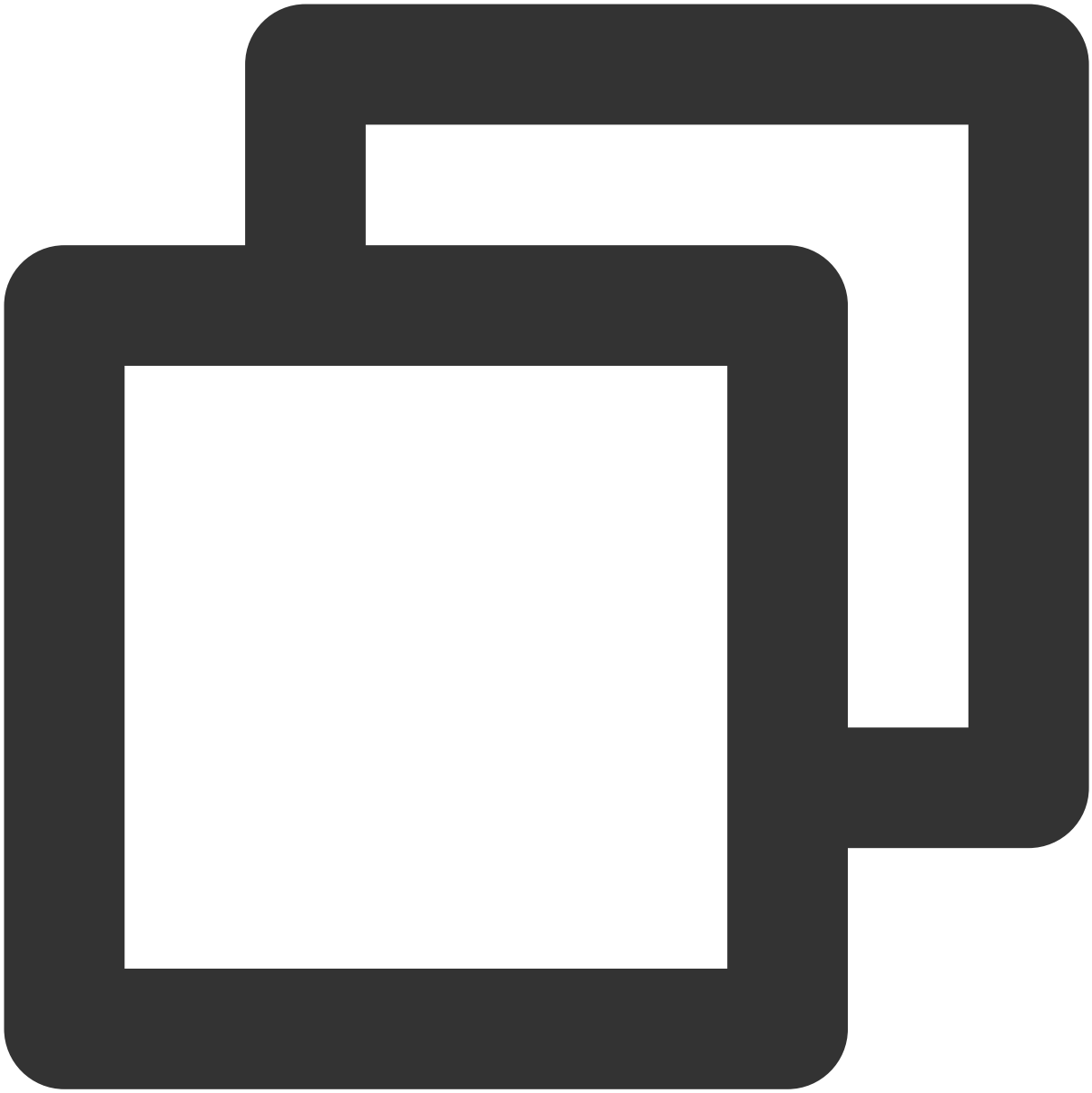
自定义部署：快速部署 Web 应用

前提条件

本地已安装 Node.js 运行环境。

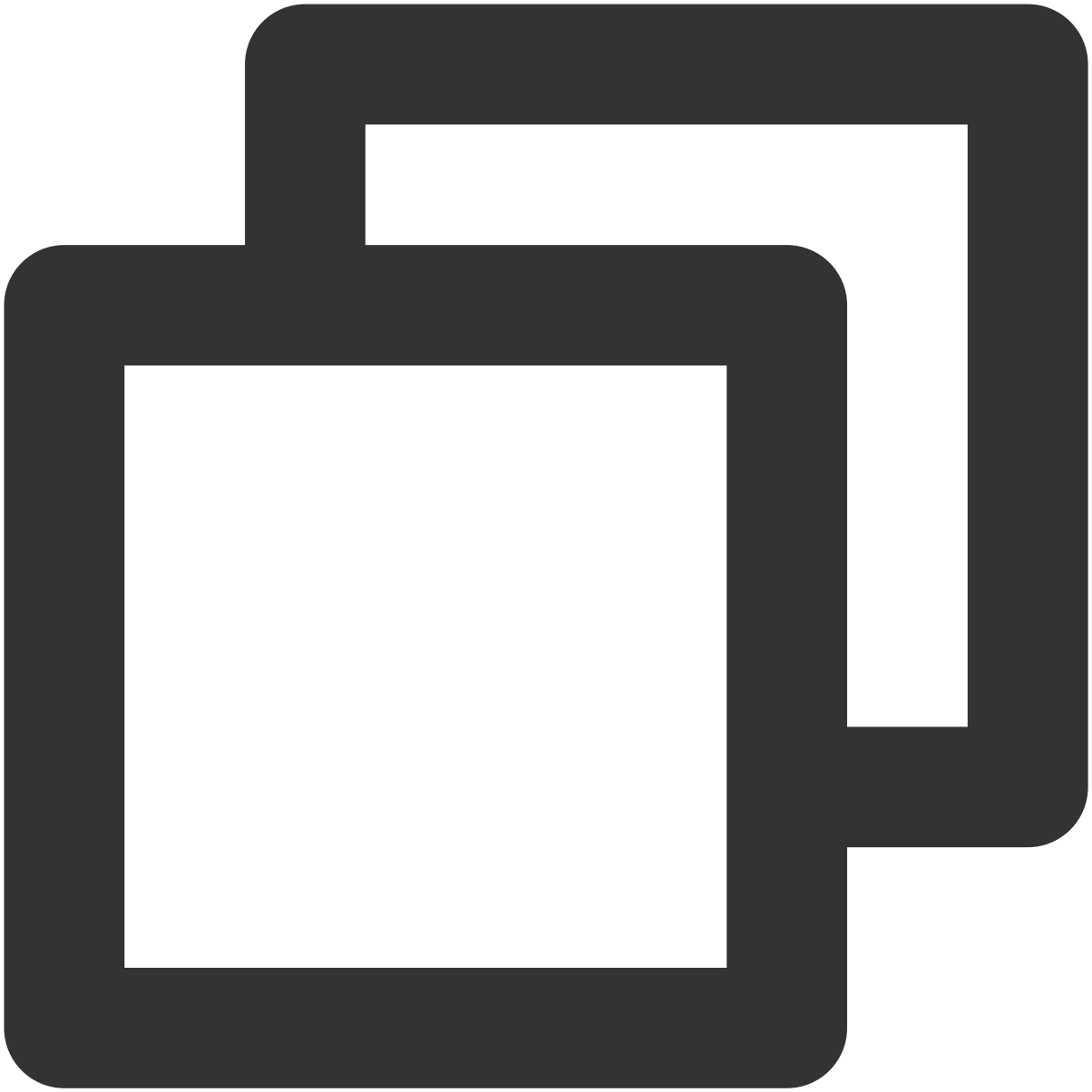
本地开发

1. 首先，在确保您的本地已安装 Node.js 运行环境后，安装 Express 框架和 express-generator 脚手架，初始化您的 Express 示例项目。



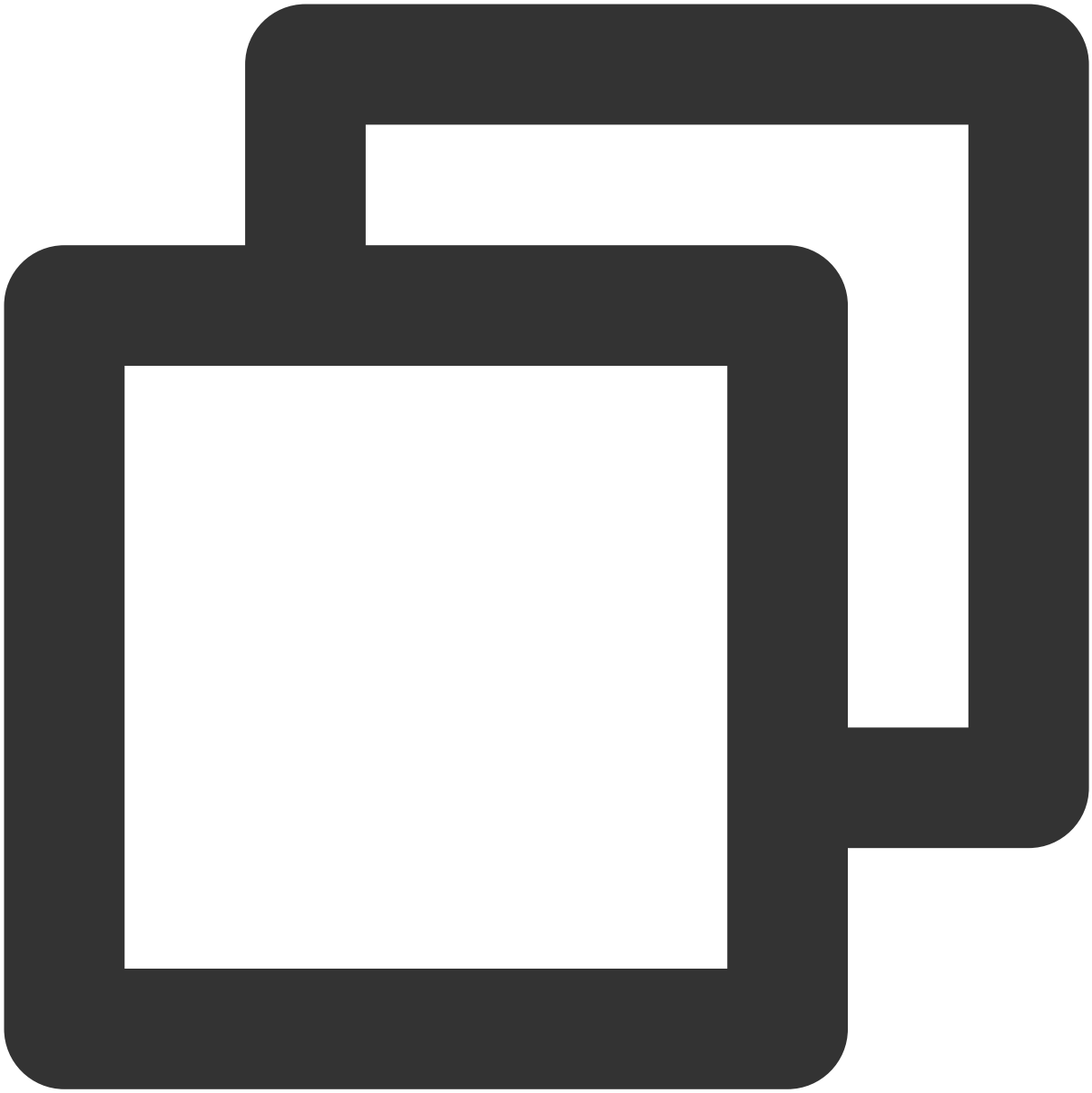
```
npm install express --save
npm install express-generator --save
express WebApp
```

2. 进入项目目录，安装依赖包。



```
cd WebApp
npm install
```

3. 安装完成后，本地直接启动，在浏览器里访问 `http://localhost:3000` ，即可在本地完成 Express 示例项目的访问。



```
npm start
```

部署上云

接下来，我们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两步：

修改监听地址与端口，改为 `0.0.0.0:9000`

新增 `scf_bootstrap` 启动文件

具体步骤如下：

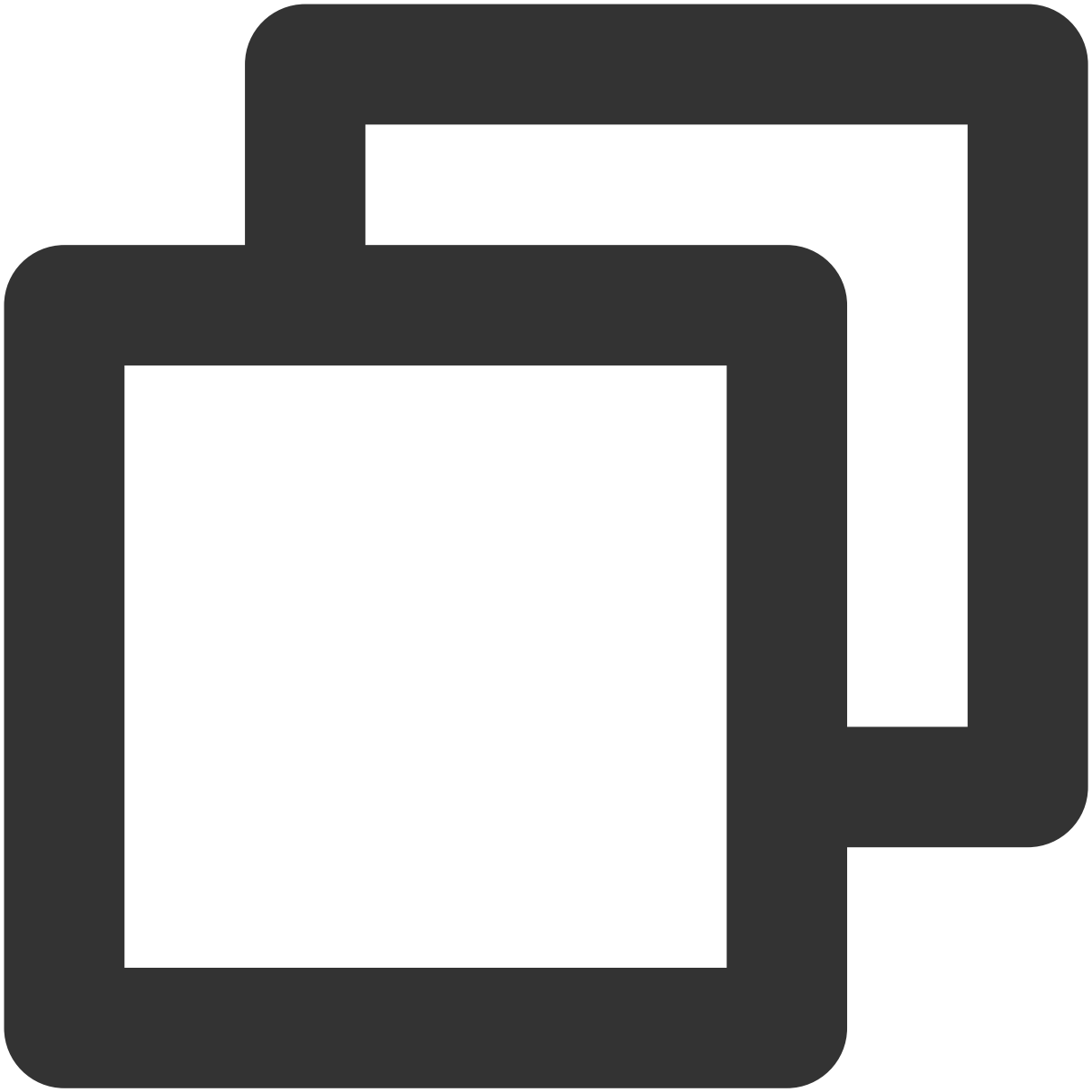
1. 已知在 Express 示例项目中，通过 `./bin/www` 设置监听地址与端口，打开该文件可以发现，我们可以通过环境变量，设置指定监听端口，否则将自动监听 `3000` 。

```
/**
 * 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. 接下来，在项目根目录下新建 `scf_bootstrap` 启动文件，在里面配置环境变量，并指定服务启动命令。

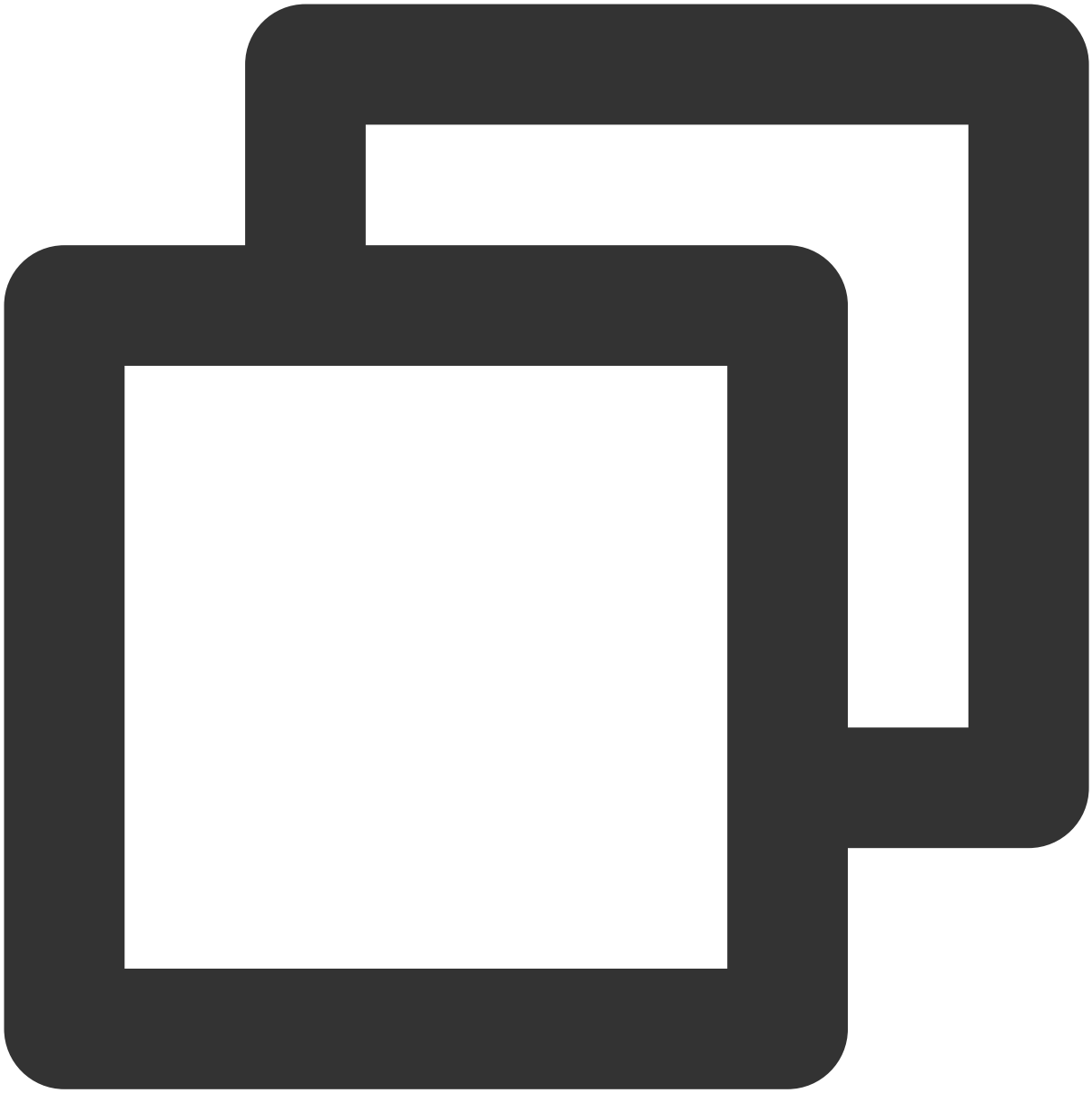
说明

您也可以在控制台完成该模块配置。



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

创建完成后，注意修改您的可执行文件权限，默认需要 `777` 或 `755` 权限才可以正常启动。



```
chmod 777 scf_bootstrap
```

3. 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，接下来，登录 Serverless 控制台，选择 **Web 应用 > Express 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

注意

启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Express 项目。

快速部署 Nextjs 框架

最近更新时间：2022-07-22 16:11:30

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function` ，快速部署您的 Next.js 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 Serverless控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

前提条件

在使用腾讯云Serverless 应用中心之前，您需要 注册腾讯云账号 并完成 实名认证。

> 注意：
> 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，请参考具体操作请参考 产品文档。

操作步骤

模板部署 -- 部署 Next.js 示例代码

1. 登录 Serverless 控制台。
2. 单击**新建应用**，选择**Web 应用>Next.js 框架**。
3. 单击“下一步”，完成基础配置选择
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Next.js 项目

自定义部署 -- 快速部署 Web 应用

前提条件

本地已安装 Node.js 运行环境。

本地开发

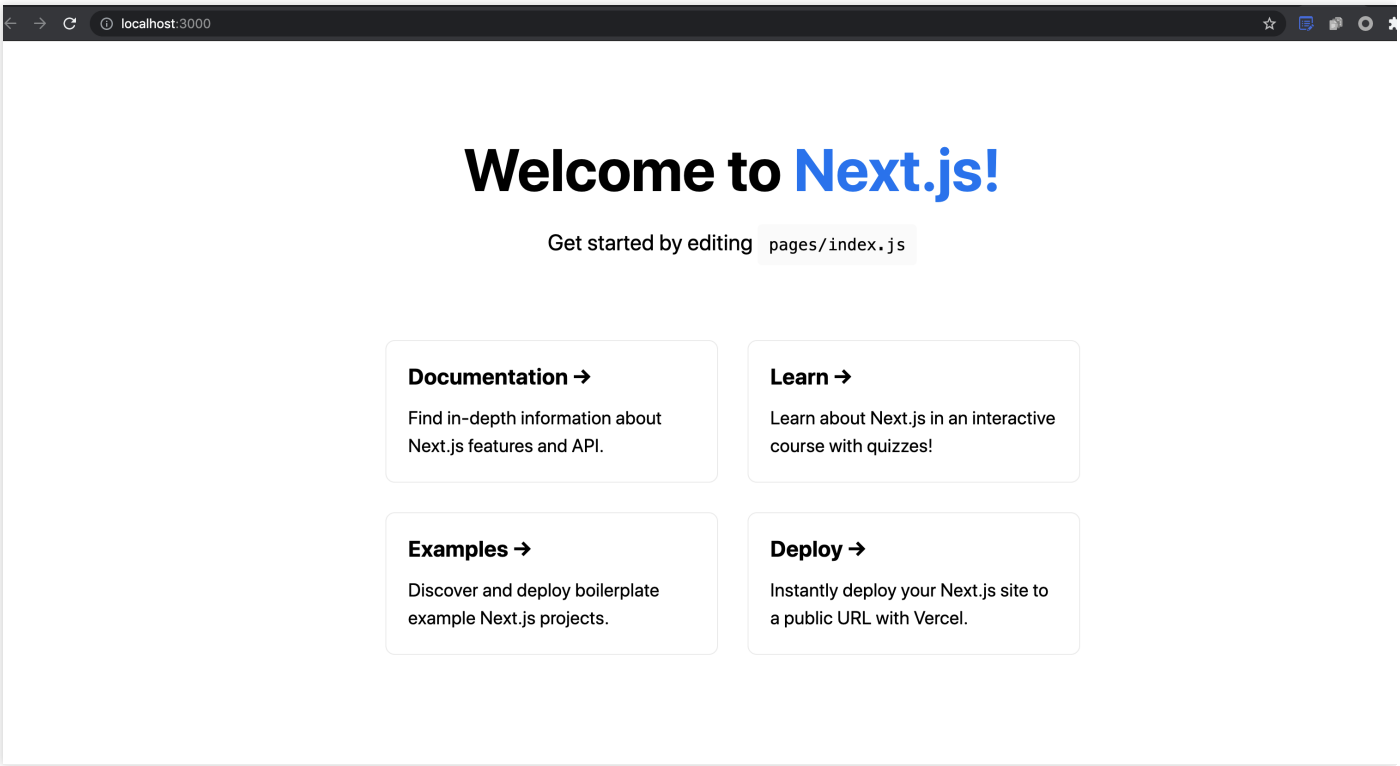
1. 参考 Next.js 官方文档，安装并初始化您的 Next.js 项目：

```
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。



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）：

> 说明：
> 您也可以在控制台完成该模块配置。

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

> 注意
> 1. 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
> 2. 示例使用的是云函数标准 node 环境路径，本地调试时，注意修改成您的本地路径

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
```

2. 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，接下来，登录 Serverless 控制台，选择**Web 应用>Next.js 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Next.js 项目。

快速部署 Nuxtjs 框架

最近更新时间：2022-07-22 17:45:14

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function` ，快速部署您的 Nuxt.js 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 Serverless控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

前提条件

- 在使用腾讯云Serverless 应用中心之前，您需要 注册腾讯云账号 并完成 实名认证。

> 注意：
> 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，请参考具体操作请参考 产品文档。

操作步骤

模板部署 -- 部署 Nuxt.js 示例代码

1. 登录 Serverless 控制台。
2. 单击**新建应用**，选择**Web 应用>Nuxt.js 框架**。
3. 单击“下一步”，完成基础配置选择。
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Nuxt.js 项目。

自定义部署 -- 快速部署 Web 应用

前提条件

本地已安装 Node.js 运行环境。

本地开发

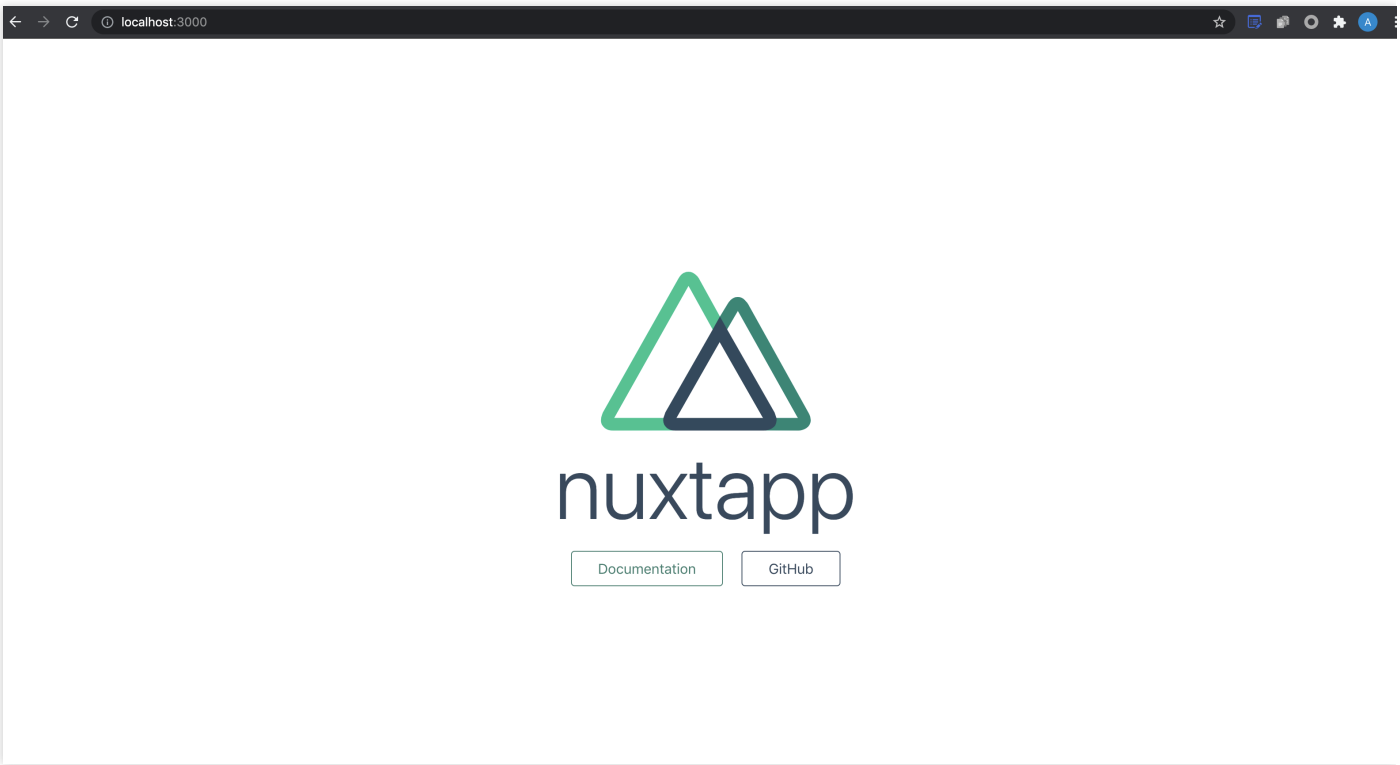
1. 参考 Nuxt.js 官方文档，安装并初始化您的 Nuxt.js 项目：

```
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。



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）：

> 说明：
>
> - 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，注意修改成您的本地路径

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

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
```

2. 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，接下来，登录 Serverless 控制台，选择**Web 应用>Nuxt.js 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Nuxt.js 项目。

快速部署 Flask 框架

最近更新时间：2022-06-13 15:10:22

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function`，快速部署您的 Flask 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 SCF 云函数控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

本篇文档为您介绍应用控制台的部署方案，您也可以通过命令行完成部署，具体操作请参考 产品文档。

模板部署 -- 部署 Flask 示例代码

1. 登录 Serverless 应用控制台。
2. 单击**新建应用**，选择**Web 应用>Flask 框架**。
3. 单击“下一步”，完成基础配置选择。
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Flask 项目。

自定义部署 -- 快速部署 Web 应用

本地开发

1. 首先需要确认您本地的环境内已经安装好 Flask。

```
pip install Flask
```

2. 本地创建 `Hello World` 示例项目
   在项目目录下，新建 `app.py` 项目，实现最简单的 Hello World 应用，示例代码如下：

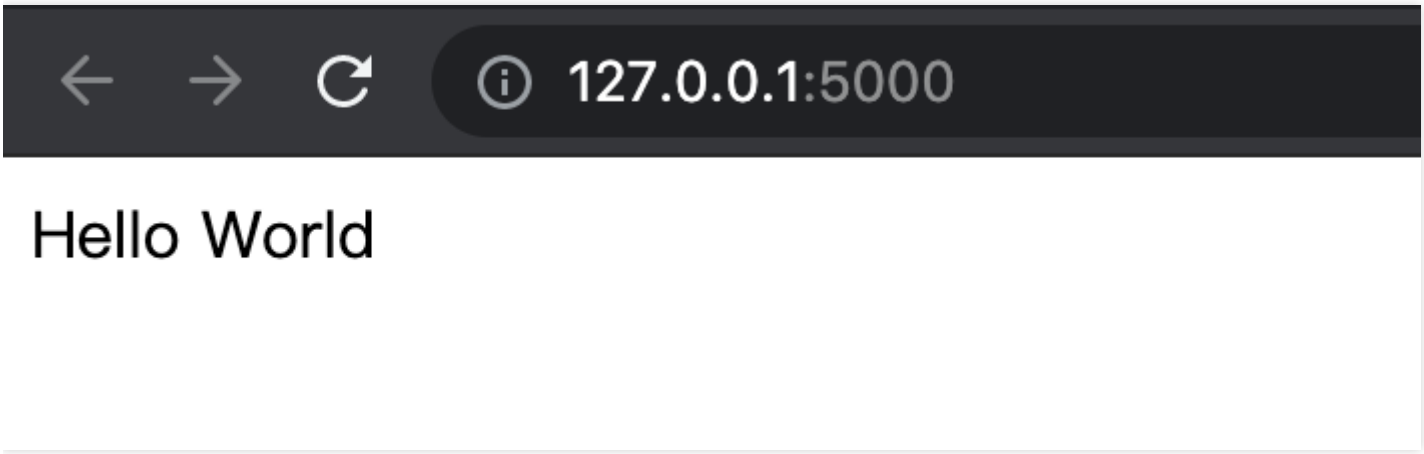
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
return 'Hello World'
if __name__ == '__main__':
app.run()
```

3. 本地运行 app.py 文件，在浏览器里访问 `http://127.0.0.1:5000` ，即可在本地完成Express 示例项目的访问。

```
$ 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 -
```



部署上云

接下来，我们对本地已经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，具体改造步骤如下：

1. 安装依赖包

由于 SCF 云上标准环境内没有 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，首先新建 `requirements.txt` 文件：

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

接下来执行安装：

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

> 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。

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`，因此需要对监听地址端口进行修改，改为 `0.0.0.0:9000`。

```
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)
```

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

3. (可选)配置 scf_bootstrap 启动文件

在项目根目录下新建 `scf_bootstrap` 启动文件，在里面完成环境变量配置，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
```

创建完成后，注意修改您的可执行文件权限，默认需要 `777` 或 `755` 权限。

```
chmod 777 scf_bootstrap
```

> 注意：
> - 您也可以在控制台完成该模块配置
> - 在 SCF 环境内，只有 `/tmp` 文件可读写，建议输出文件时选择 `/tmp`，其它目录会由于缺少权限而写入失败
> - 如果想要在日志中输出环境变量，启动命令前需要加 `-u` 参数，示例：`python -u app.py`

4. 控制台上传

登录 Serverless 应用控制台，选择**Web 应用>Flask 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Flask 项目。

高级配置管理

您可在“高级配置”里进行更多应用管理操作，如创建层、绑定自定义域名、配置环境变量等。

快速部署 Laravel 框架

最近更新时间：2021-08-20 16:21:27

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。

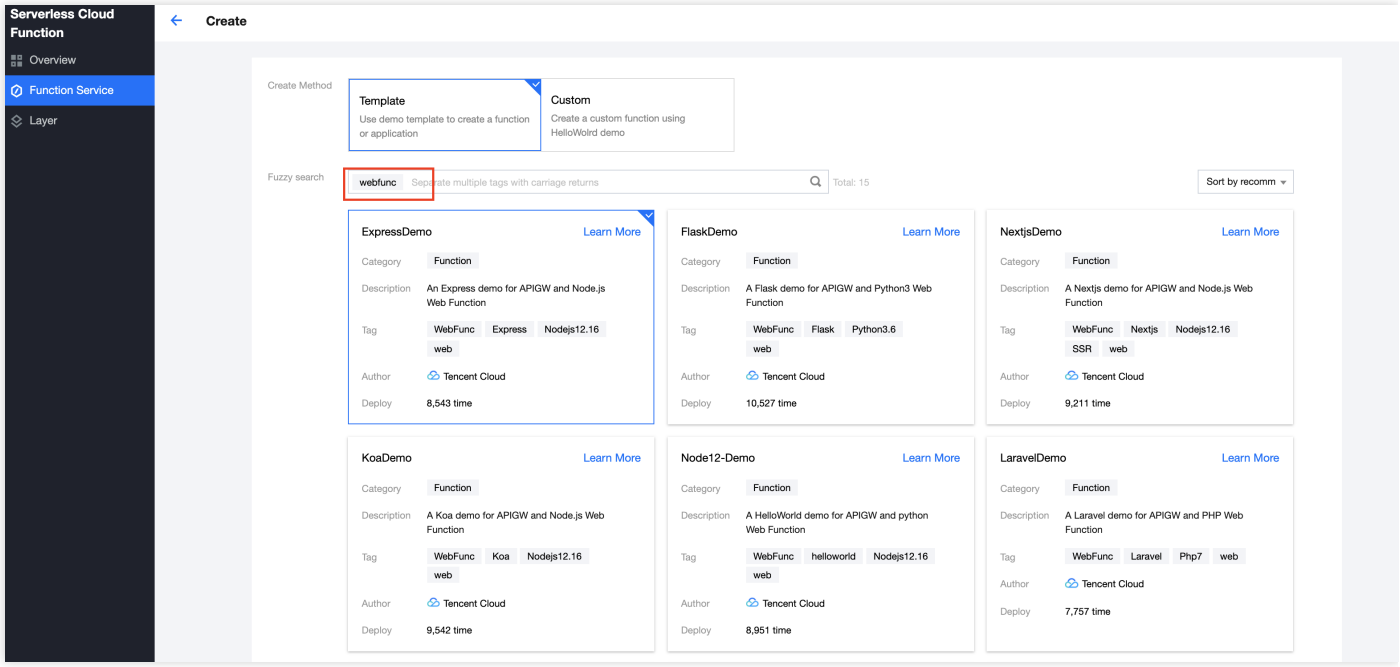
前提条件

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。

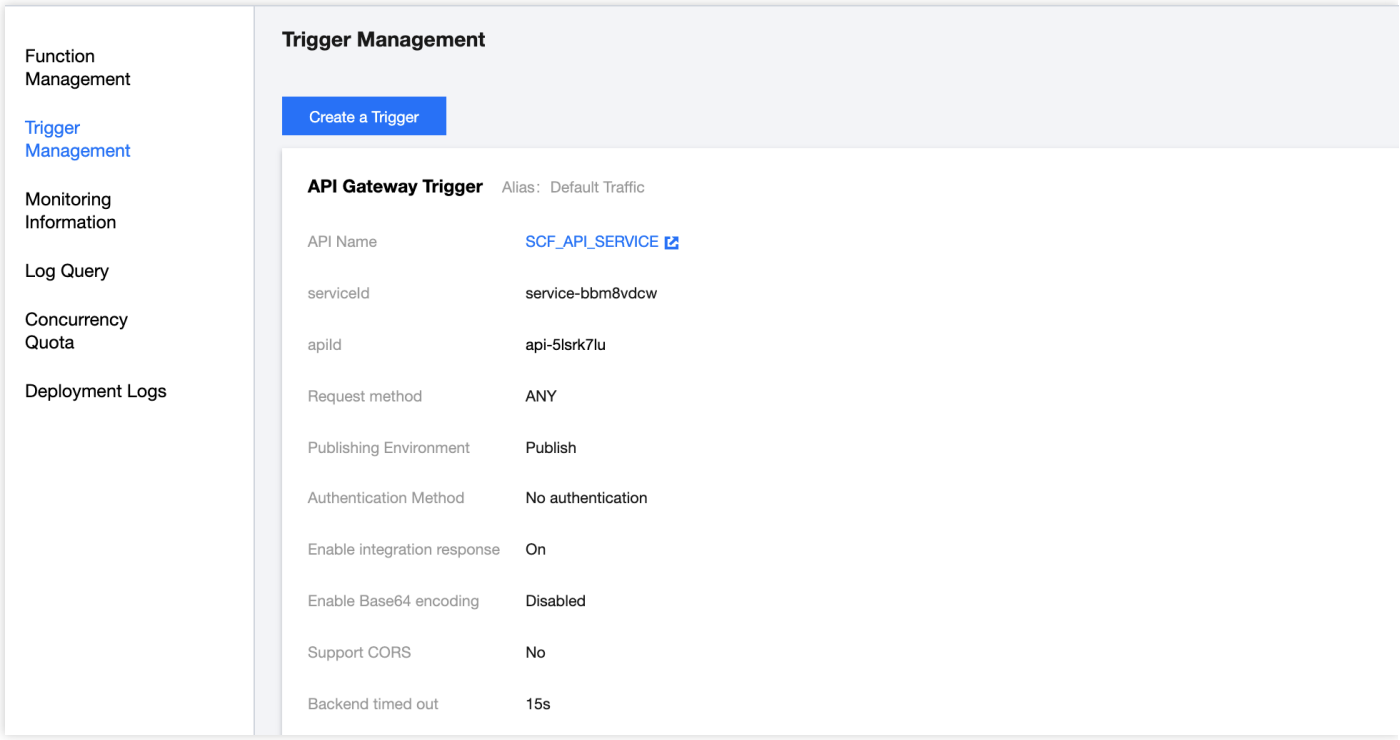
操作步骤

模版部署 -- 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，并单击【新建】，进入函数创建流程。
3. 选择使用【模版创建】来新建函数，在搜索框里筛选 `WebFunc`，筛选所有 Web 函数模版，选择【Laravel 框架模版】并单击【下一步】。如下图所示：



4. 在“配置”页面，您可以查看模版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 API 网关生成的访问路径 URL，访问您部署的 Laravel 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 Laravel 项目。

自定义部署 -- 快速迁移本地项目上云

本地开发

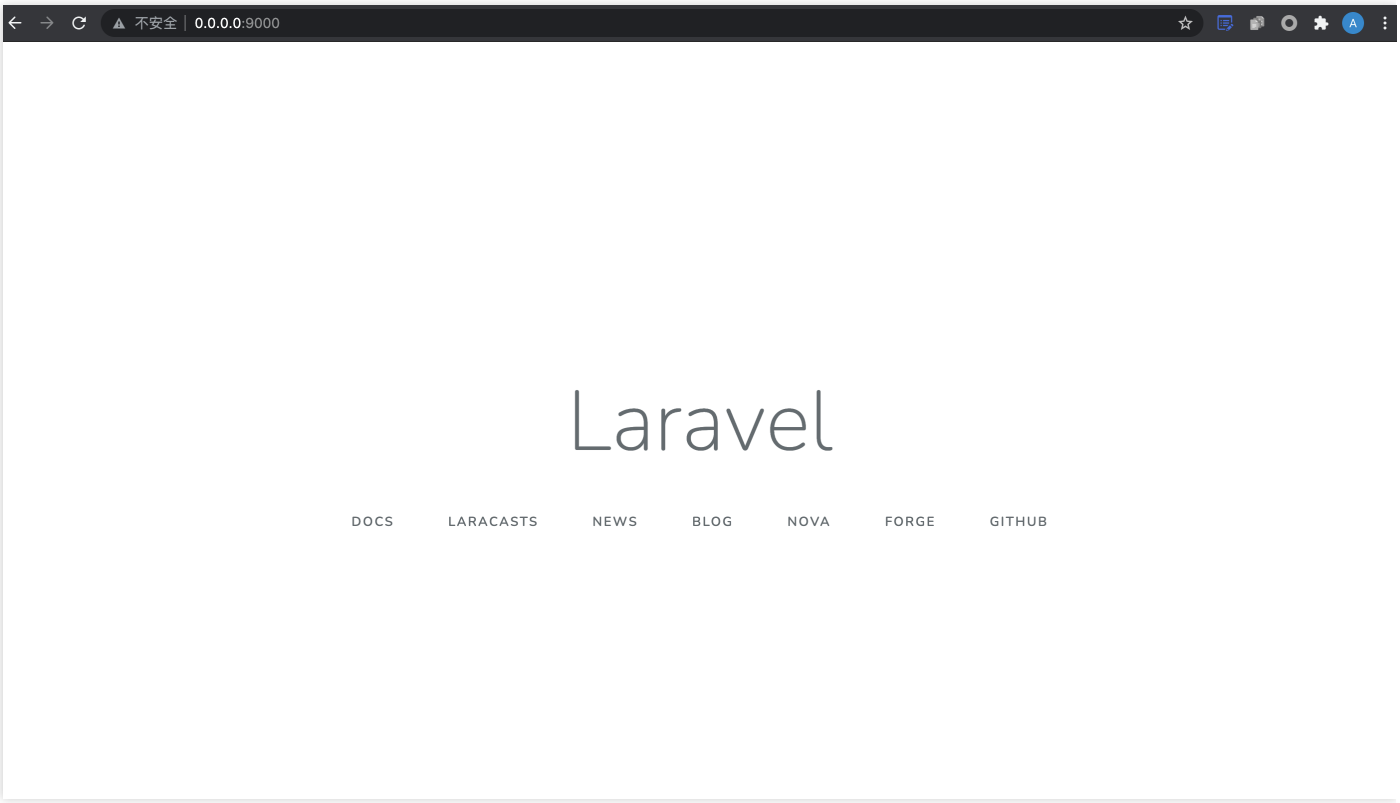
1. 参考 Laravel 官方文档，在本地环境中完成 Laravel 的开发环境搭建。
2. 在本地创建 Laravel 示例项目。进入项目目录下，执行以下命令，初始化 Laravel 示例应用：

```
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

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来调整）：

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，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：

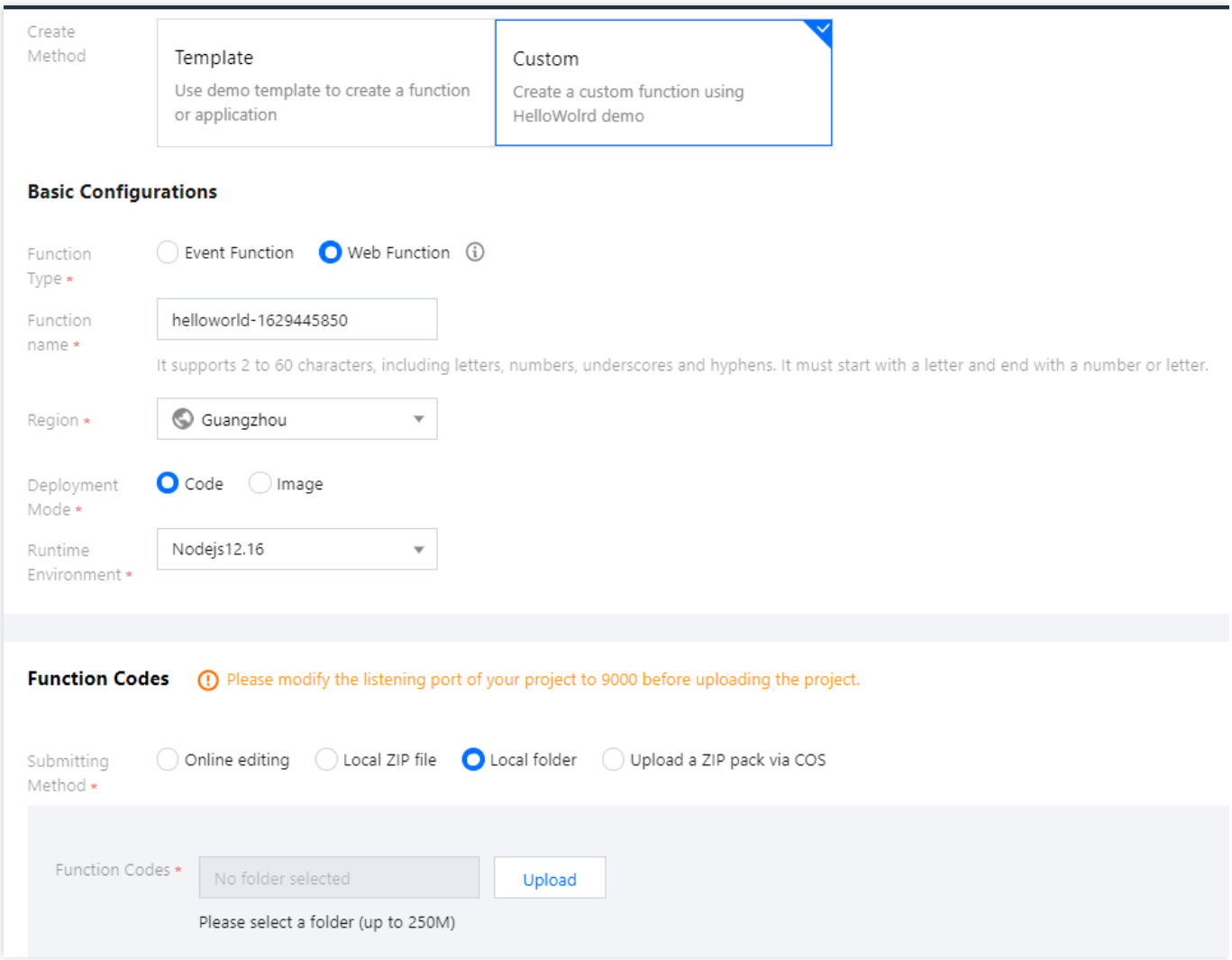
4. **部署上云**

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：

i. 登录 Serverless 控制台，单击左侧导航栏的【函数服务】。

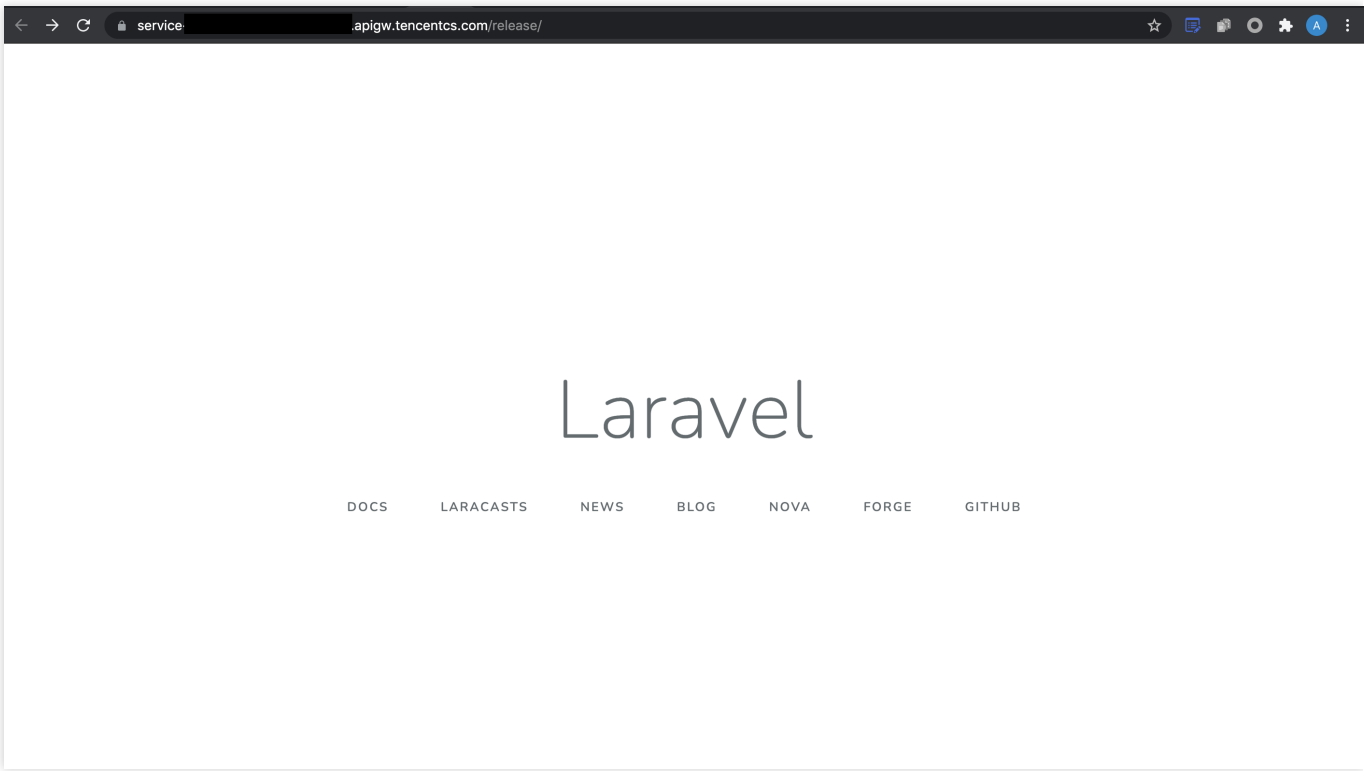
ii. 在主界面上方选择期望创建函数的地域，并单击【新建】，进入函数创建流程。

iii. 选择【自定义创建】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 “Php7”。

iv. 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：2022-06-13 15:10:22

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function` ，快速部署您的 Nest.js 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 SCF 云函数控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

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。

> 注意：
> 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，请参考具体操作请参考 产品文档。

操作步骤

模板部署 -- 部署 Nest.js 示例代码

1. 登录 Serverless 应用控制台。
2. 单击**新建应用**，选择**Web 应用>Nest.js 框架**。
3. 单击“下一步”，完成基础配置选择
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Nest.js 项目

自定义部署 -- 快速部署 Web 应用

前提条件

本地已安装 Node.js 运行环境。

本地开发

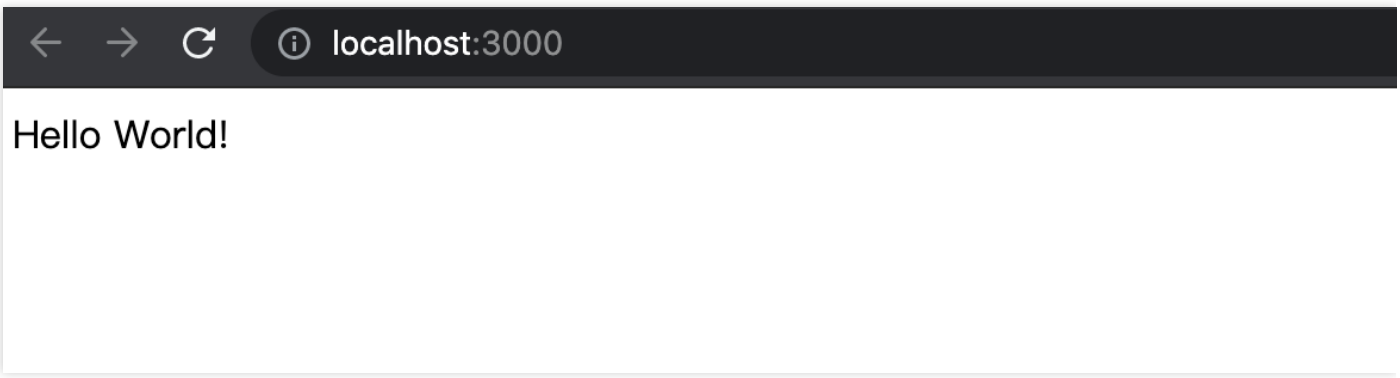
1. 参考 Nest.js 官方文档，初始化您的 Nest.js 项目：

```
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。



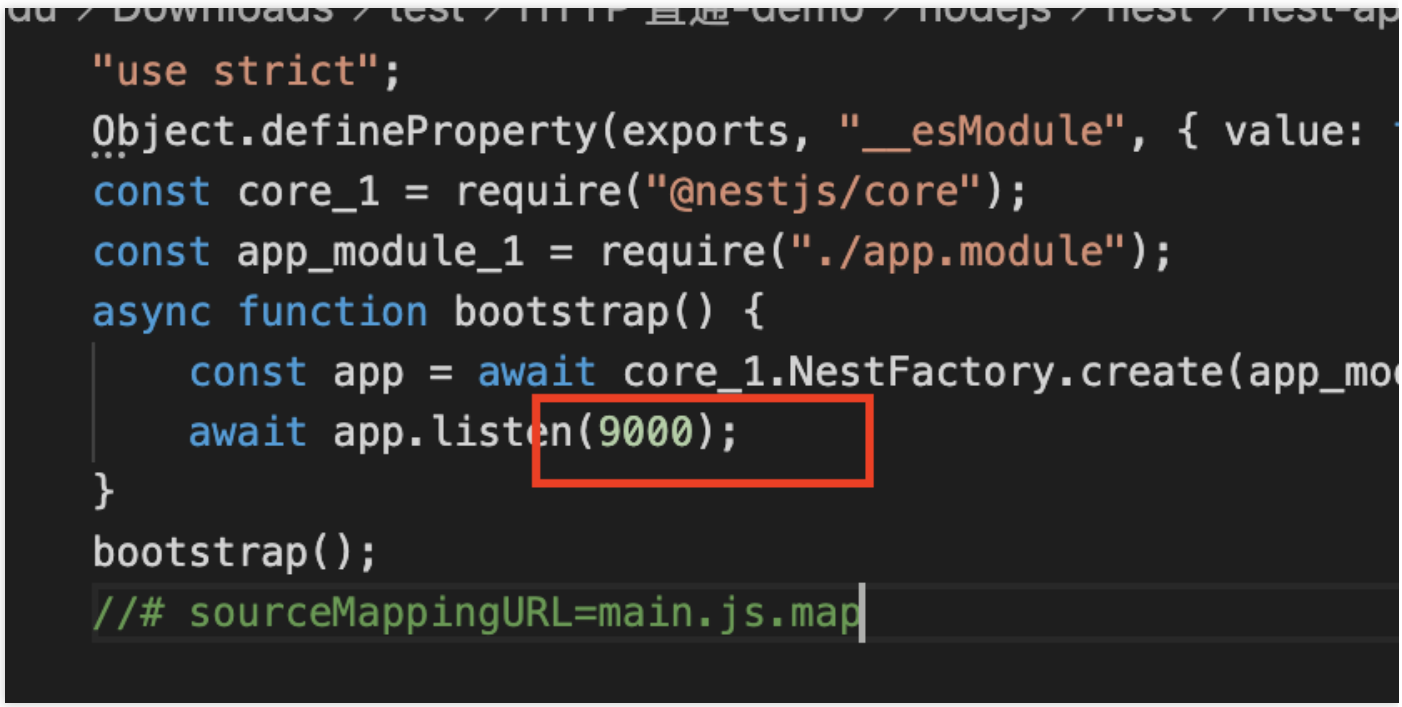
部署上云

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`：



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）：

> 说明：
> 您也可以在控制台完成该模块配置。

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

> 注意：

1. 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。
2. 示例使用的是云函数标准 node 环境路径，本地调试时，注意修改成您的本地路径。

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. 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，接下来，登录 Serverless 应用控制台，选择 **Web 应用>Nest.js 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**。

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

> 注意：
> 启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Nest.js 项目。

快速部署 Django 框架

最近更新时间：2022-06-13 15:10:22

应用中心框架部署方案已经全新升级，您可以通过 `SCF Web Function`，快速部署您的 Django 业务上云。

> 注意：
> **应用控制台部署与函数直接部署有什么区别？**
> 通过应用部署或函数部署，均可以基于 Web 函数，快速部署常见 Web 框架。

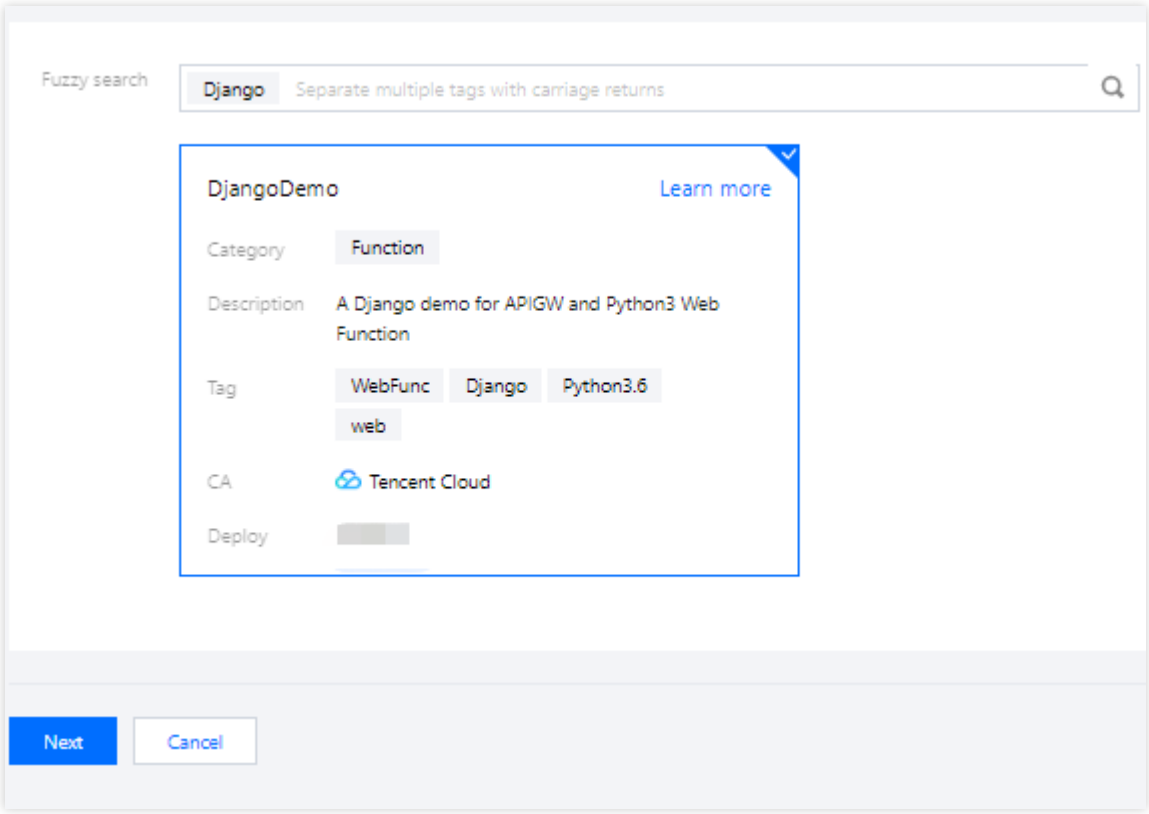
- 如果您只关注代码逻辑开发，无需额外资源创建，可以通过 SCF 云函数控制台，完成快速部署。
- 如果除了代码部署外，您还需要更多能力或资源创建，如自动创建层托管依赖、一键实现静态资源分离、支持代码仓库直接拉取等，可以通过应用控制台，完成 Web 应用的创建工作。

本篇文档为您介绍应用控制台的部署方案，您也可以通过命令行完成部署，具体操作请参考 产品文档。

模板部署 -- 部署 Django 示例代码

1. 登录 Serverless 应用控制台。

2. 单击**新建应用**，选择**Web 应用>Django 框架**，如下图所示：



3. 单击“下一步”，完成基础配置选择。
4. 上传方式，选择**示例代码**直接部署，单击**完成**，即可开始应用的部署。
5. 部署完成后，您可在应用详情页面，查看示例应用的基本信息，并通过 API 网关生成的访问路径 URL 进行访问，查看您部署的 Django 项目。

自定义部署 -- 快速部署 Web 应用

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
```

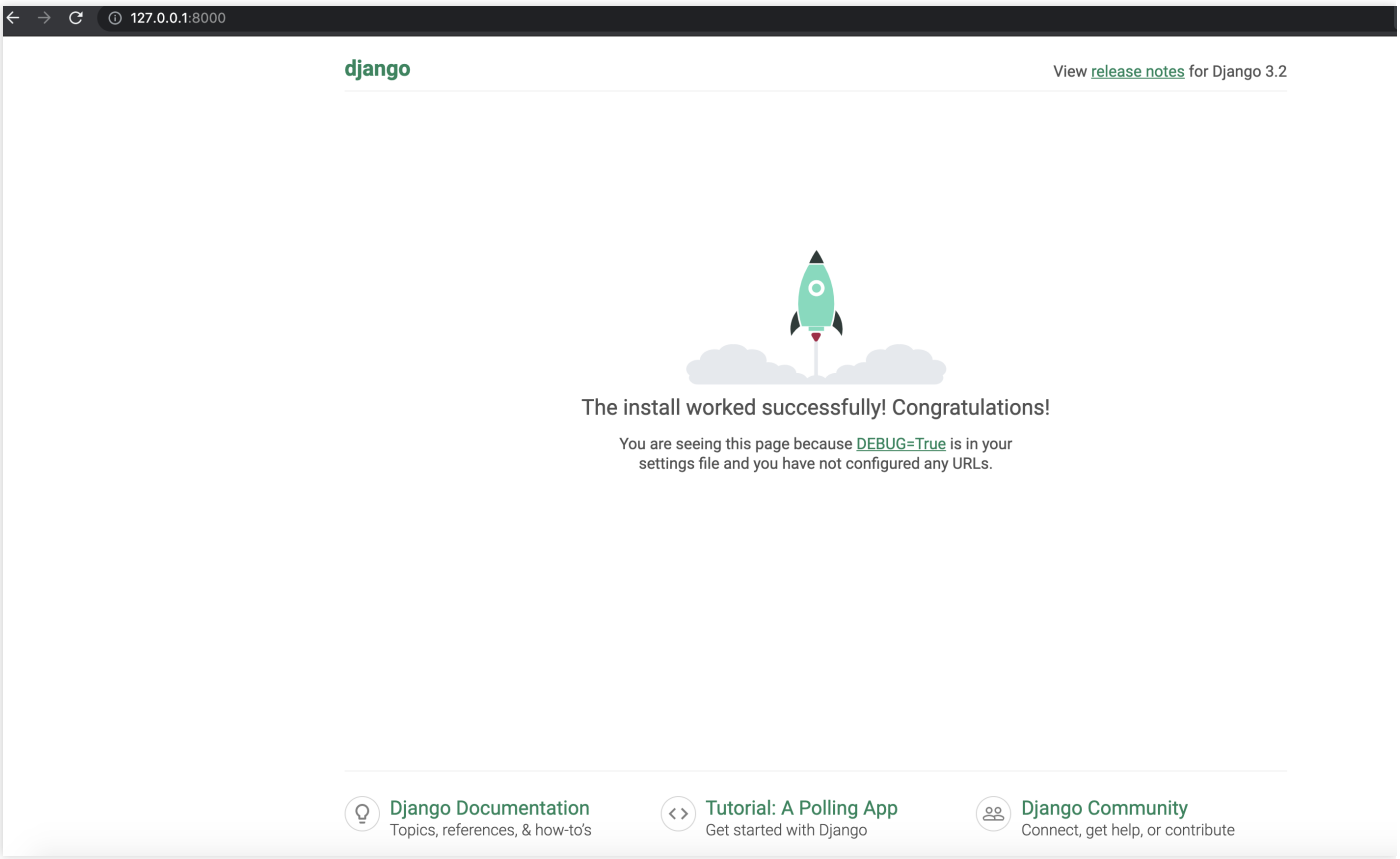
目录结构如下：

```
$ 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

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
```

执行以下命令进行安装：

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

> 注意：
> 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` 字段部分配置注释掉。

2. (可选)配置 scf_bootstrap 启动文件

> 说明：
> 您也可以在控制台完成该模块配置。

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
```

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`。

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
```

3. 控制台上传

登录 Serverless 应用控制台，选择**Web 应用>Django 框架**，上传方式可以选择**本地上传**或**代码仓库拉取**

您可以在控制台完成启动文件 `scf_bootstrap` 内容配置，配置完成后，控制台将为您自动生成 启动文件，和项目代码一起打包部署。

注意：
启动文件以项目内文件为准，如果您的项目里已经包含 `scf_bootstrap` 文件，将不会覆盖该内容。

配置完成后，单击**完成**，部署您的 Django 项目。

高级配置管理

您可在“高级配置”里进行更多应用管理操作，如创建层、绑定自定义域名、配置环境变量等。

快速部署 Wordpress 原生应用

最近更新时间：2021-08-20 16:21:27

腾讯云 Serverless 提供了基于 Serverless 架构的 Wordpress 全新部署方式，通过 Serverless Framework Wordpress 组件，仅需几步，就可以快速部署一个 Wordpress 项目。

架构简介

该组件主要为您创建以下资源：

模块	说明
云函数 SCF	负责 Serverless Wordpress 的接入层实现，从而运行 WordPress
API 网关	WordPress 的对外入口，实现了 RESTful API
CFS	WordPress 的 Serverless 存储仓库
TDSQL-C Serverless	通过创建 TDSQL-C Serverless (原 CynosDB) 的 MySQL 类型数据库，实现数据库按量计费，自动扩缩容
VPC	内网打通 SCF 云函数、CFS、TDSQL-C Serverless 之间的网络，保障网络隔离

功能优势

- **支持 Wordpress 原生框架**
  使用 Serverless Wordpress 组件，您无需对原生 Wordpress 项目进行任何改造，即可直接完成部署，做到对框架无入侵，也支持后续的版本升级。
- **降低使用成本**
  从接入层到计算层到存储层，全部使用 Serverless 资源，真正做到按量计费，弹性伸缩，极大节省成本。
- **部署步骤简单**
  通过 Serverless Wordpress 组件，只需几行 YAML 文件配置，即可快速完成 Wordpress 应用部署，极大降低部署门槛。

部署步骤

您可以通过**命令行**或**控制台**完成 Serverless Wordpress 部署，步骤如下：

前提条件

- 已开通 云函数 SCF 服务。
- 已开通 文件存储 CFS 服务。
- （可选）准备好的自定义域名。

控制台部署

> 注意：
> 目前只支持北京、广州、南京、上海四个区域。

1. 登录 Serverless 应用控制台, 单击【新建应用】。
2. 根据指引，填入应用名称，选择【应用模版】>【Wordpress 应用】，单击【创建】既可以完成应用新建。
3. 在 Serverless 应用页，单击【访问应用】，即可访问您的 Wordpress 项目，您也可以在应用详情页完成自定义域名的配置。

命令行部署

> 注意：
> 目前只支持 `ap-guangzhou-4` 、 `ap-shanghai-2` 、 `ap-beijing-3` 、 `ap-nanjing-1` 四个可用区。

1. 本地创建 `wordpress-demo` 文件夹，在 Wordpress 官网 下载应用到该文件夹内。
2. 在文件夹内创建 `serverless.yml` 配置文件，完成应用信息配置，参考如下（更多配置内容，请参见 全量配置文档）：

```
app: wordpress
stage: dev
component: wordpress
name: wordpressDemo
inputs:
region: ap-shanghai # 项目所在区域
zone: ap-shanghai-2
src: # 项目路径，选择您的 wordpress 路径
src: ./wordpress
exclude:
- .env
apigw: # api网关配置
customDomains: # （可选）自定义域名绑定
```

```
- domain: abc.com # 待绑定的自定义的域名
certId: abcdefg # 待绑定自定义域名的证书唯一 ID
customMap: true # 是否自定义路径
pathMap:
- path: /
environment: release
protocols: # 绑定自定义域名的协议类型，默认与服务的前端协议一致。
- http
- https
```

完成后，您的项目结构如下：

```
.wordpress-demo
├── wordpress # wordpress 源文件
├── serverless.yml # 配置文件
└── .env # 环境变量文件
```

3. 在根目录下，执行 `sls deploy`，即可完成部署。示例如下：

```
$ sls deploy
serverless ⚡framework
Action: "deploy" - Stage: "dev" - App: "appDemo" - Instance: "wordpressDemo"
region: ap-shanghai
zone: ap-shanghai-2
vpc:
...
cfs:
...
db:
...
apigw:
created: true
url: https://service-xxxxx.sh.apigw.tencentcs.com/release/
...
layer:
...
wpInitFaas:
...
wpServerFaas:
...
```

4. 部署成功后，单击 `apigw` 部分输出的 URL，根据指引完成账号密码配置，即可开始使用您的 Wordpress 应用。

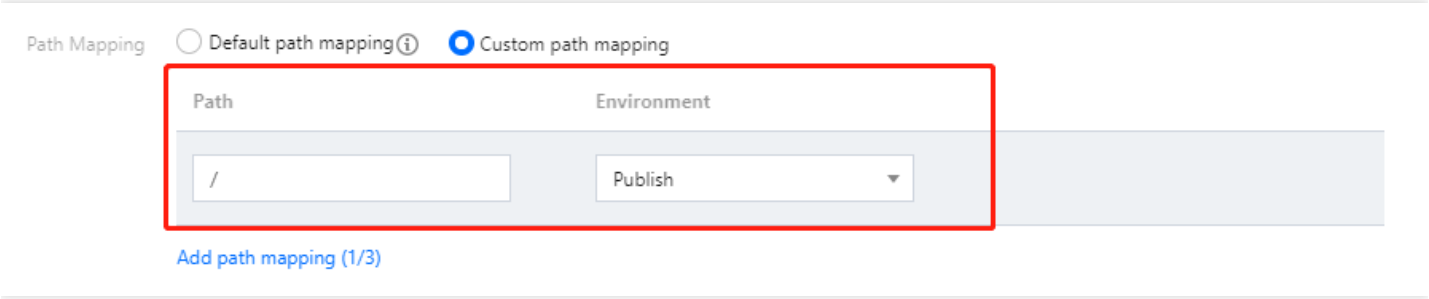
常见问题

权限问题导致部署失败该如何处理？

- 主账号/子账号需确认是否有以下权限：
  - 确认角色：**SCF_QcsRole、SLS_QcsRole、CODING_QcsRole**
  - 确认权限：
    - SCF_QcsRole 须拥有 **CFSFullAccess** 权限
    - CODING_QCSRole 须拥有 **QcloudSLSFullAccess、QcloudSSLFullAccess、QcloudAccessForCODINGRole** 权限
- 子账号还需确认以下权限：
  账号本身有 **SLS、SCF、CFS、CynosDB、CODING** 使用权限。

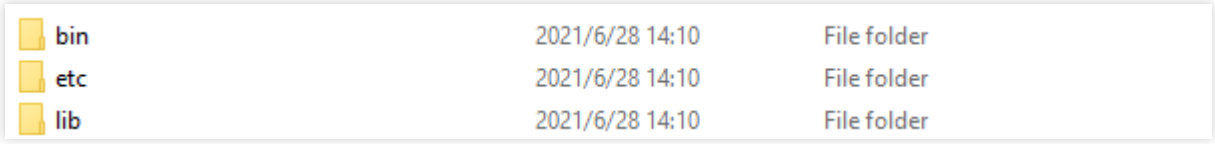
绑定自定义域名后，显示报错 {"message":"There is no api match env_mapping '/'"}？

在 API 网关控制台 修改自定义映射，如下图所示：



如何通过修改 php.ini 修改上传文件大小限制？

1. 修改 layer 代码。将 etc 文件夹中的 php.ini 文件移到 etc/php.d 文件夹下，您也可以直接使用我们提供的 压缩包。重新打包上传 layer 时，注意打包层级结构，只打包父文件夹下的文件，否则会出现函数初始化失败：



2. 按照如下修改 wp-server-xxx 函数的 bootstrap 代码：

```
#!/bin/bash
export PATH="/opt/bin:$PATH"
export LD_LIBRARY_PATH=/opt/lib/:$LD_LIBRARY_PATH
export PHP_INI_SCAN_DIR=/opt/etc/php.d
php -d extension_dir=/opt/lib/php/modules/ sl_handler.php 1>&2
```

如何处理报错 "event too large"？

函数目前只支持最大**6MB**的事件上传，超过该大小文件不支持上传。

目前 API 网关 base 64转码会将用户本身代码大小扩大1.5倍左右，因此上传文件时，建议文件大小控制在**3.5MB**以内。

如何修改 Wordpress 根目录文件？

目前文件挂载在文件存储 CFS 上，无法直接修改，建议通过安装 File Manager 插件管理根目录文件。